VaultGemma: A Differentially Private Gemma Model

VaultGemma Team[1, 2]
[1]Google Research, [2]Google DeepMind

We introduce VaultGemma 1B, a 1 billion parameter model within the Gemma family, fully trained with differential privacy. Pretrained on the identical data mixture used for the Gemma 2 series, VaultGemma 1B represents a significant step forward in privacy-preserving large language models. We openly release this model to the community.

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks, yet a significant challenge in their development and deployment is the inherent privacy risk. Trained on vast, web-scale corpora, LLMs have been shown to be susceptible to verbatim memorization and extraction of training data (Biderman et al., 2023; Carlini et al., 2021, 2023; Ippolito et al., 2023; Lukas et al., 2023; Prashanth et al., 2025). This can lead to the inadvertent disclosure of sensitive or personally identifiable information (PII) that was present in the pretraining dataset.

To address these challenges, Differential Privacy (DP) (Dwork et al., 2006) has emerged as the gold standard, providing a rigorous, mathematical framework to limit the influence of any single example in the training data on the resulting model. A model trained with DP provably bounds the reconstruction or leakage of information tied to individual data points.

An LLM encounters the vast majority of its training data during the initial pretraining phase (Abdin et al., 2024; Gemma Team et al., 2024a,b). This stage relies on massive, heterogeneous datasets that, despite filtering efforts, can contain sensitive information. Thus, it is important to consider pretraining an LLM fully with DP. This approach provides an end-to-end privacy guarantee from the ground up, ensuring the foundational model is built in a way that prevents the memorization of specific, sensitive details. It allows the model to learn general patterns and knowledge about the world without being overly influenced by any single document or user's data, fundamentally mitigating the risk of privacy leaks from the original training corpus.

A common alternative to full private pretraining is to apply DP exclusively during the fine-tuning phase. However, this approach leaves the foundational model and its vast pretraining data unprotected, leading to several significant pitfalls. First, the model may have already memorized sensitive PII from the pretraining corpus before DP fine-tuning begins, and this process does not retroactively erase the memorized information. This leaves the model vulnerable to extraction attacks that can reveal verbatim training data—a risk that is significantly amplified for open-weight models, where adversaries have full access to model weights to probe for and reconstruct sensitive information. Consequently, this practice can create a false sense of security, as labeling the model "private" applies only to the fine-tuning data, while the core pretraining data remains at risk (Tramèr et al., 2024).

VaultGemma represents a significant step forward in the journey toward building AI that is both powerful and private by design. By developing and applying a new, robust understanding of the scaling laws for DP (McKenna et al., 2025), we have successfully trained and released the largest open-weight, privately trained language model to date. Our primary motivation for releasing VaultGemma is to accelerate research and development in private AI. By providing the community with a powerful, high-utility private model and a clear methodology, we aim to lower the barrier to entry for building privacy-preserving

Corresponding author(s): amersinha@google.com

technologies. Furthermore, this model can serve as a valuable foundation for applications where privacy of the training data when using the model is paramount.

While a utility gap still exists between private and non-private models, our work demonstrates that this gap can be systematically narrowed. We hope that VaultGemma and our accompanying research will empower the community to build the next generation of safe, responsible, and private AI for everyone.

2. Model Architecture

Similar to previous Gemma models (Gemma Team et al., 2024a,b), VaultGemma 1B, is a decoder-only transformer model, with most architecture elements similar to other Gemma versions.

Table 1 | Overview of the main parameters and design choices for the 1B model. See section 2 for more details.

Parameters	1B
d_model	1152
Layers	26
Pre-norm	yes
Post-norm	no
Non-linearity	GeGLU
Feedforward dim	13,824
Head type	MQA
Num heads	4
Num KV heads	1
Head size	256
Global att. span	1024
Vocab size	256,128
Tied embedding	yes

2.1. Sequence Length

We choose to decrease the sequence length to 1024 for pretraining. We find that using a smaller sequence length significantly reduces compute requirements, which in turn allows us to train using larger batch sizes—a necessity for good performance in private training.

2.2. Global Attention

Given our use of a small sequence length, we choose to use global attention on all layers rather than alternating with sliding window attention.

2.3. Pre-norm with RMSNorm

To stabilize training, we use RMSNorm (Zhang and Sennrich, 2019) to normalize the input of each transformer sub-layer, the attention layer, and the feedforward layer.

3. Dataset

We train using the same pretraining dataset as Gemma 2 27B. This dataset contains 13T tokens of primarily-English data. These tokens come from a variety of data sources, including web documents, code, and science articles and only contain text data. (Gemma Team et al., 2024b)

3.1. Filtering

We use the same data filtering techniques as (Gemma Team et al., 2024b). Specifically, we filter the pretraining dataset to reduce the risk of unwanted or unsafe utterances, filter out certain personal information or other sensitive data, decontaminate evaluation sets from our pretraining data mixture, and reduce the risk of recitation by minimizing the proliferation of sensitive outputs.

3.2. Tokenizer

We use the same non-private tokenizer as Gemma 1, Gemma 2, and Gemini: a SentencePiece tokenizer (Kudo and Richardson, 2018) with split digits, preserved whitespace, and byte-level encodings. The resulting vocabulary has 256K entries.

4. Evaluations

We show the performance of our final pretrained model across a variety of benchmarks and compare it against its non-private counterpart across a range of standard academic benchmarks in Table 2. To put this performance in perspective

and quantify the current impact of DP on performance, we also include a comparison to an older similar-sized GPT-2 model, which performs similarly on these benchmarks. This comparison illustrates that today's private training methods produce models with utility comparable to that of non-private models from roughly five years ago, highlighting the important gap our work will help the community systematically close.

5. DP Implementation

5.1. Private Training

We implemented DP-SGD (Abadi et al., 2016) on top of the Gemma pretraining pipeline using clipping and noise addition components provided by JAX Privacy (Balle et al., 2025). Our implementation uses vectorized per-example clipping for maximum parallelism, and gradient accumulation to simulate large batch sizes. Gradient accumulation steps are independent and each adds properly calibrated Gaussian noise to the partial gradients so that when these partial noisy gradients are averaged we obtain the target gradient required for DP-SGD model updates.

5.2. Batch Construction

Repeated Documents. Our training mixture is composed of a diverse set of documents sampled from numerous source datasets. We sample documents from these source documents into the mixture with probability proportional to its weight. As these datasets are of different quality, we assign different weights to sample documents from them into our mixture. At worst, we can sample a single document from these sources up to seven times in our mixture. However, for most of the source datasets, we sample documents fewer than three times into our final mixture.

Packing. To increase training efficiency, we pack documents into fixed size sequences of 1024 tokens. Due to the diversity of the type of documents we train on, our packing can pack multiple documents into a single example or break up a single document into multiple sequences.

Privacy Guarantee. VaultGemma was trained with a $(\epsilon \leq 2.0, \delta \leq 1.1e^{-10})$-sequence-level DP guarantee, where sequence consists of 1024 tokens after sampling and packing. It is important to note that if two repeated sequences occurred due to the sampling of repeated documents, they are treated as separate privacy units for the purpose of this guarantee.

Truncated Poisson Subsampling. We employ Truncated Poisson Subsampling (Chua et al., 2024) for sampling our mini-batches. This method provides a computationally efficient approximation of Poisson subsampling, where each example in the dataset is included in a batch with a fixed probability. Poisson subsampling results in batches of variable size which can result in slower training throughput. Truncated Poisson Subsampling allows us to use a fixed batch size by padding the batch when it is too small and truncating it when it is too large.

A key distinction in our methodology lies in the implementation of the data pipeline. The original work suggests a implementing truncated Poisson sub-sampling using MapReduce. This approach is designed to minimize the overhead during training by front-loading the sampling computation. However, we found that a pre-generation step was not necessary for our use case and introduced complexities in data handling and storage.

Instead, we implement Truncated Poisson Sampling directly within our data loading pipeline using `pygrain` (Ritter et al., 2023). Our implementation performs on-the-fly sampling and batching of the data as it is fed to the model. We found that this approach does not introduce a significant computational overhead, and we observe data throughput speeds comparable to a standard data pipeline at the same physical batch size. We also find that overhead of padding is small at large batch sizes that we use for training with the padding accounting for less than 2% of the total batch size.

5.3. Privacy Accounting

Our privacy accounting methodology is based on the ABLQ$_P$ method under the "zeroing-out" ad-

Table 2 | A comparison of DP and standard model performance and training configurations.

Model	ARC-C 0 shot	ARC-E 0 shot	HellaSwag 0 shot	PIQA 0 shot	SIQA 0 shot	BoolQ 0 shot	TriviaQA 5 shot
VaultGemma 1B	26.45	51.78	39.09	68.00	46.16	62.04	11.24
Gemma3 1B (PT)	38.31	71.34	61.04	77.37	49.28	68.75	39.75
GPT-2-1.5B	39.78	51.10	47.91	70.51	-	61.80	6.00

jacency notion (Kairouz et al., 2021) as detailed in (Chua et al., 2024). The accounting is implemented using the PLD accountant (Doroshenko et al., 2022) from the Google DP accounting library (Google DP Team, 2022).

6. Scaling Law

6.1. Methodology

Our methodology for deriving scaling laws for DP language models builds upon the framework established in (McKenna et al., 2025), while introducing three key modifications to enhance the modeling of the optimal learning rate, the estimation of loss values, and the final scaling law formulation. This approach allows for a more granular and robust understanding of the interplay between model size, training iterations, and the noise-to-batch ratio under DP constraints.

Explicit Modeling of the Optimal Learning Rate. A primary departure from previous work is the explicit modeling of the optimal learning rate. Instead of treating the learning rate as a hyperparameter to be optimized through a grid search for each configuration, we model its optimal value as a function of the training setup. For each experimental configuration, defined by a specific model size and noise-to-batch ratio, we conduct seven training runs, each with a different learning rate.

The resulting final loss values from these seven runs are then used to fit a quadratic function. The vertex of this parabola provides an estimate of the optimal learning rate for that configuration. This approach is based on the observation that the relationship between the learning rate and the final loss is often convex. The final scaling laws are subsequently derived using the loss values obtained from training each configuration at its modeled optimal learning rate. This two-step process allows for a more precise determination of the best achievable performance for each configuration, reducing the risk of suboptimal learning rate selection influencing the final scaling law.

Parametric Extrapolation of Loss Values. To efficiently estimate the loss across a continuous range of training iterations without relying on intermediate loss values, we adopt a parametric fitting approach. For each model configuration, we execute five separate training runs, each with a different, predetermined number of training iterations. The final loss from each of these five runs is then recorded.

We use a simple parametric form inspired by (Hoffmann et al., 2022), namely $L = E + \frac{A}{T^{\alpha}}$ where L is the loss, T is the number of training iterations and E, A, and α are the fitted parameters. We fit this function using `scipy.optimize.curve_fit` to these five data points. This function allows for both interpolation and extrapolation of the loss to any number of iterations within a reasonable range. We find that this reduces overestimating the loss when training iterations are smaller than the experimental configuration.

Semi-Parametric Fit. Our final scaling law is constructed through a two-stage fitting process, which separately models the loss as a function of model size and training iterations as a parametric function described above.

We utilize the parametric extrapolation to gen-

erate a dense grid of loss values across a wide range of model sizes, training iterations, and noise-batch ratios. These generated data points are then used to fit separate non-parametric model following the methodology in (McKenna et al., 2025) to predict the loss value for any value of model size, training iterations, and noise-batch ratios.

7. Training Configuration

7.1. Compute Infrastructure

We train on a $2 \times 16 \times 16$ configuration of TPUv6e totaling 2048 chips, with 2048-way data replication and 1-way model sharding. Using a vectorized implementation of per-example-clipping, we are able to process four examples per core in parallel and aggregate gradients computed across 64 independent iterations to produce each model update. As in other Gemma variants, we use the GSPMD partitioner for training step computation and the MegaScale XLA compiler.

7.2. Batch Size and Number of Iterations

Based on our scaling law and compute budget, we find a variety of configurations produce comparable loss similar to (McKenna et al., 2025). Our final training configuration is in Table 3. After training, we find that our scaling law prediction for loss was within 1% of the true value achieved.

Table 3 | Overview of the training hyperparameters.

Parameters	**1B**
Iterations	100,000
Expected Batch Size	517,989
Noise Multiplier	0.6143481
ϵ	2.0
δ	$1.1e^{-10}$

8. Empirical Privacy and Memorization

One major benefit of DP is that it provably bounds the impact of an example on the trained model. In this section, we empirically assess these benefits on training-example memorization rates (Carlini et al., 2023; Gemini Team, 2024; Gemma Team, 2024; Nasr et al., 2023). This "memorization rate"[1] is defined as the ratio of generations from the model that match its training data compared to all model generations using the following setup. Thus, this rate assesses a model's likelihood of producing near-copies of text used in training (Biderman et al., 2023; Carlini et al., 2021; Ippolito et al., 2023).

We follow the methodology described in (Gemma 3 Team, 2025). We subsample roughly 1M training data samples distributed uniformly across different corpora and test for discoverable extraction (Nasr et al., 2023) of this content using a prefix of length 50 and a suffix of length 50. If all tokens in the continuation match the source suffix, we denote the text as "exactly memorized". If the continuations matches up to an edit distance of 10%, we denote this as "approximately memorized".

Figure 1 compares the memorization rates across Gemma models ordered in reverse chronological order starting from the left. All non-DP versions of Gemma had detectable levels of memorization, that are decreasing in time due to changes in the training architecture and recipe. We were not able to detect any memorization from DP Gemma, despite using the older training architecture and recipe from Gemma 2.

9. Conclusion

In this work, we introduced VaultGemma, the largest open-weight language model trained from its inception with a formal DP guarantee. The development of this model was guided by our formulation of novel scaling laws for DP training, which provide a principled and quantitative framework for navigating the inherent trade-offs between model utility, privacy, and computational cost. We are releasing the model weights and our training methodology to facilitate reproducibility and

[1] We do not state or imply [here] that a model "contains" its training data in the sense that there is a copy of that data in the model. Rather, a model memorizes attributes of its training data such that in certain cases it is statistically able to generate such training data when following rules and using information about features of its training data that it does contain.

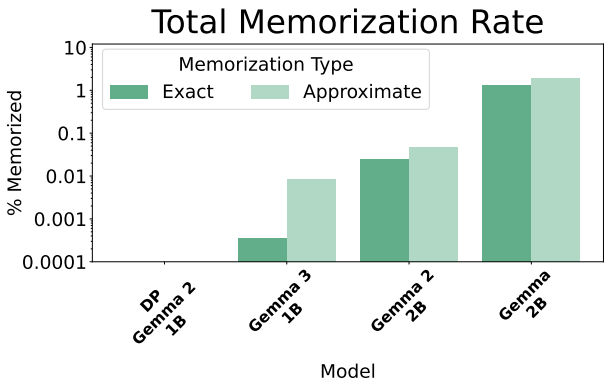

Figure 1 | Total memorization rates for both exact and approximate memorization. No memorization was detected for DP Gemma.

encourage further research in privacy-preserving machine learning.

While our results are promising, a utility gap persists between privately and non-privately trained models. The scaling laws we present offer a clear roadmap for future research aimed at improving the performance of private models. Future work could focus on developing more efficient algorithms, exploring novel data curation strategies, and applying these principles to even larger models. VaultGemma serves as a powerful baseline and a key step toward making large-scale, provably private AI a practical reality.

10. Contributions and Acknowledgments

Core Contributors
Amer Sinha
Thomas Mesnard
Ryan McKenna
Daogao Liu
Christopher A. Choquette-Choo
Yangsibo Huang
Da Yu
George Kaissis
Zachary Charles
Ruibo Liu
Lynn Chua
Pritish Kamath
Pasin Manurangsi
Steve He

Tech Leads
Chiyuan Zhang
Badih Ghazi
Borja De Balle Pigem

Product
Prem Eruvbetine
Tris Warkentin

Leads
Armand Joulin
Ravi Kumar

Acknowledgements
Peter Kairouz
Brendan McMahan
Dan Ramage
Andreas Terzis

References

M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318, 2016.

M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv*, 2404.14219, 2024.

B. Balle, L. Berrada, Z. Charles, C. A. Choquette-Choo, S. De, V. Doroshenko, D. Dvijotham, A. Galen, A. Ganesh, S. Ghalebikesabi, J. Hayes, P. Kairouz, R. McKenna, B. McMahan, A. Pappu, N. Ponomareva, M. Pravilov, K. Rush, S. L. Smith, and R. Stanforth. JAX-Privacy: Algorithms for privacy-preserving machine learning in JAX, 2025. URL http://github.com/google-deepmind/jax_privacy.

S. Biderman, U. Prashanth, L. Sutawika, H. Schoelkopf, Q. Anthony, S. Purohit, and E. Raff. Emergent and predictable memorization in large language models. In *NeurIPS*, pages 28072–28090, 2023.

N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *USENIX Security*, pages 2633–2650, 2021.

N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramèr, and C. Zhang. Quantifying memorization across neural language models. In *ICLR*, 2023.

L. Chua, B. Ghazi, P. Kamath, R. Kumar, P. Manurangsi, A. Sinha, and C. Zhang. Scalable DP-SGD: Shuffling vs. Poisson subsampling. In *NeurIPS*, 2024.

V. Doroshenko, B. Ghazi, P. Kamath, R. Kumar, and P. Manurangsi. Connect the dots: Tighter discrete approximations of privacy loss distributions. *PETS*, 2022(4):552–570, 2022.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv*, 2403.05530, 2024.

Gemma 3 Team. Gemma 3 technical report. *arXiv:2503.19786*, 2025.

Gemma Team. Gemma: Open models based on Gemini research and technology. *arXiv*, 2403.08295, 2024.

Gemma Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on Gemini research and technology. *arXiv*, 2403.08295, 2024a.

Gemma Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv*, 2408.00118, 2024b.

Google DP Team. Google's differential privacy libraries., 2022. https://github.com/google/differential-privacy.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv*, 2203.15556, 2022.

D. Ippolito, F. Tramèr, M. Nasr, C. Zhang, M. Jagielski, K. Lee, C. A. Choquette-Choo, and N. Carlini. Preventing verbatim memorization in language models gives a false sense of privacy. In *INLG-SIGDIAL*, 2023.

P. Kairouz, B. Mcmahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. Practical and private (deep) learning without sampling or shuffling. In *ICML*, pages 5213–5225, 2021.

T. Kudo and J. Richardson. SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv*, 1808.06226, 2018.

N. Lukas, A. Salem, R. Sim, S. Tople, L. Wutschitz, and S. Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *S & P*, 2023.

R. McKenna, Y. Huang, A. Sinha, B. Balle, Z. Charles, C. A. Choquette-Choo, B. Ghazi, G. Kaissis, R. Kumar, R. Liu, D. Yu, and C. Zhang. Scaling laws for differentially private language models. In *ICML*, 2025.

M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee. Scalable extraction of training data from (production) language models. *arXiv*, 2311.17035, 2023.

U. S. Prashanth, A. Deng, K. O'Brien, J. SV, M. A. Khan, J. Borkar, C. A. Choquette-Choo, J. R. Fuehne, S. Biderman, T. Ke, et al. Recite, reconstruct, recollect: Memorization in LMs as a multifaceted phenomenon. In *ICLR*, 2025.

M. Ritter, I. Indyk, A. Singh, A. Audibert, A. Seelam, C. Hanes, E. Lau, J. Olesiak, J. Kang, and X. Wu. Grain - feeding JAX models, 2023. URL http://github.com/google/grain.

F. Tramèr, G. Kamath, and N. Carlini. Position: Considerations for differentially private learning with large-scale public pretraining. In *ICML*, pages 48453–48467, 2024.

B. Zhang and R. Sennrich. Root mean square layer normalization. In *NIPS*, 2019.