nanoT5: A PyTorch Framework for Pre-training and Fine-tuning T5-style Models with Limited Resources

Piotr Nawrot
University of Edinburgh
https://github.com/PiotrNawrot/nanoT5

Abstract

State-of-the-art language models like T5 have revolutionized the NLP landscape, but their computational demands hinder a large portion of the research community. To address this challenge, we present nanoT5, a specially-optimized PyTorch framework for efficient pre-training and fine-tuning of T5 models. Drawing on insights from optimizer differences and prioritizing efficiency, nanoT5 allows a T5-Base model to be pre-trained on a single GPU in just 16 hours, without any loss in performance. With the introduction of this open-source framework, we hope to widen the accessibility to language modelling research and cater to the community's demand for more user-friendly T5 (Encoder-Decoder) implementations. We make our contributions, including configurations, codebase, pre-training insights, and pre-trained models, available to the public.

1 Introduction

The transformative power of large pre-trained language models such as GPT-3 (Brown et al., 2020), T5 (Raffel et al., 2019), and PaLM (Chowdhery et al., 2022) is undeniable. However, their massive computational requirements remain a barrier for many researchers. Notably, models like T5 require extensive datasets and significant computational resources for their pre-training (Raffel et al., 2019). Furthermore, many open-source implementations lean heavily on TPU accelerators (Shazeer, 2020), which are not as available to the academic community as GPUs.

Recognizing this gap, we introduce nanoT5, a resource-efficient, open-source PyTorch framework designed for the pre-training and fine-tuning of T5 models. Inspired by pioneering efforts such as nanoGPT (Karpathy, 2021) and Cramming (Geiping and Goldstein, 2022), nanoT5 uniquely concentrates on enhancing the training pipeline specifically for T5 encoder-decoder models. Our framework includes optimized configurations and scripts, enabling researchers to pre-train a T5-Base model with 248M parameters on a single GPU in just 16 hours. Every facet, from data preprocessing and model architecture to the learning rate schedule, has been tuned for both efficiency and adaptability. With nanoT5, users can seamlessly initiate model pre-training within minutes of accessing our GitHub repository.

This paper underscores two main innovations: First, we delve into the nuances between the Adam and Adafactor optimizer performances as detailed in (Havinga), suggesting a version of AdamW (Loshchilov and Hutter, 2017), augmented with matrix-wise learning rate scaling based on root mean square. This variant showcases better speed and robustness compared to the default Adafactor (Shazeer and Stern, 2018). Second, we demonstrate that T5 models trained with nanoT5, housing around 250M parameters, can achieve performance akin to the publicly-available checkpoints while requiring 150x less pre-training data.

Our primary motivation stems from the growing demand for reproducible and tuned baselines (Kaddour et al., 2023), enabling fast and small-scale hypothesis validation in the evolving realm of large pre-trained Transformers. With nanoT5, we address a gap highlighted by community requests [1][2][3], providing an approachable platform for working with T5 (Encoder-Decoder) architecture. To our understanding, nanoT5 pioneers the effort to reproduce T5 v1.1 pre-training using PyTorch, deviating from prior Jax/Flax implementations. We invite the community to explore our training configurations, codebase, and pre-trained models, all of which are available at https://github.com/PiotrNawrot/nanoT5.

[1] https://github.com/google-research/text-to-text-transfer-transformer/issues/172
[2] https://github.com/huggingface/transformers/issues/18030
[3] https://github.com/huggingface/transformers/issues/5079

2 Related Work

The landscape of open-source repositories tailored for efficient pre-training of Transformer language models is vast. Notably, nanoGPT (Karpathy, 2021) sheds light on decoder-only models, while Cramming (Geiping and Goldstein, 2022) homes in on the optimal pre-training of the encoder-only BERT architecture (Devlin et al., 2019). Contrastingly, with nanoT5, we sought to bridge the existing gap by providing a standalone research template tailored for the T5-style (Encoder-Decoder) models.

To expedite the training process of nanoT5 we incorporated various optimizations. These encompass mixed precision training (Micikevicius et al., 2017), compiled runtimes (Narang et al., 2021), and more. Additionally, we delved into the potential of efficient training methodologies such as recent optimizers (Chen et al., 2023; Liu et al., 2023), and fast attention mechanism (Dao et al., 2022), which are elaborated further in Section 4.3. It's crucial to note that while we evaluated various efficient algorithms, we consciously opted against those, such as (Nawrot et al., 2022; Shazeer et al., 2017), that would modify the core model structure. Instead, our intent with nanoT5 was to cultivate a straightforward baseline for further research endeavors. The standout contribution of our work in terms of efficient training algorithms is the AdamW variant, with the RMS matrix scaling, which improves T5 pre-training convergence.

3 Methodology

Our validation strategy seeks to replicate the T5-base pre-training outcomes detailed in (Shazeer, 2020) and the fine-tuning results of Tk-Instruct on the Super Natural-Instructions (SNI) meta-dataset (Wang et al., 2022).

3.1 Training pipeline

We've devised a comprehensive training pipeline prioritizing efficient data management, low-level optimizations, and coding simplicity, all while preserving the core model and training logic:

- **Dataset Handling:** Given the extensive volume of the C4 dataset, which exceeds 300GB, our repository implements concurrent data downloading with model training. This optimization speeds up the commencement of T5 model pre-training to a few minutes.
- **Exposure and Simplicity:** Our methodology aims to strike a balance between adaptability and abstraction. With tools such as the HuggingFace Accelerator (Sylvain Gugger, 2022), we abstract tasks like checkpoint management and tensor operations. Experiment tracking is realized via neptune.ai (Neptune team, 2019), and we've employed hydra (Yadan, 2019) for coordinated hyperparameter handling.
- **Efficiency:** We've leveraged the optimizations of PyTorch 2.0 (Paszke et al., 2019), and employed mixed-precision training in line with established optimization guidelines [4][5].
- **Flexibility:** Our repository is designed with adaptability in mind, offering support for multi-GPU training, gradient accumulation, and gradient checkpointing. This ensures users can reproduce our results on a variety of GPUs beyond the A100 and can experiment with configurations larger than the T5-base size emphasized in this study. Additionally, we provide support for both CPUs and Apple's ARM M1 chips.

3.2 Pre-training

Our experiments strictly follow the T5-v1.1-base training configuration (Shazeer, 2020), where the model itself comprises of roughly 248M parameters. The C4 dataset (Raffel et al., 2019), sourced directly from Huggingface, undergoes tokenization via the Wordpiece tokenizer (Schuster and Nakajima, 2012), with the original model's vocabulary. During pre-processing, 15% of input data is masked using sentinel tokens, setting the neural network's target as the prediction of these tokens, leveraging its decoder. Consistent with the original study, we've set the batch size at 128 examples, with inputs of 512 tokens and outputs of 114 tokens. Optimization is facilitated through the Adafactor optimizer (Shazeer and Stern, 2018), combined with the Inverse-Square-Root (ISR) learning rate schedule. The model is trained for 2^{16} steps. For more details please refer to the original work.

3.3 Fine-tuning

Our fine-tuning employs the Super Natural-Instructions (SNI) meta-dataset (Wang et al., 2022), which has been previously used for fine-tuning

[4] https://huggingface.co/docs/transformers/perf_train_gpu_one

[5] https://pytorch.org/tutorials/recipes/recipes/tuning_guide.html

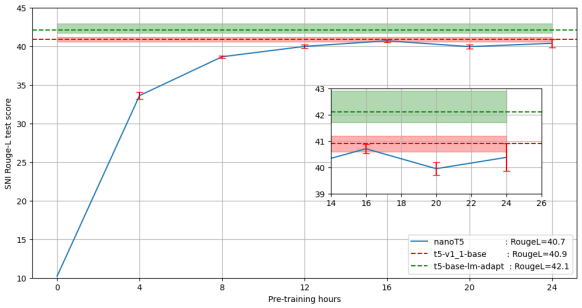

Figure 1: Downstream performance of models across various pre-training durations, including existing T5-base variants accessible through Huggingface Hub.

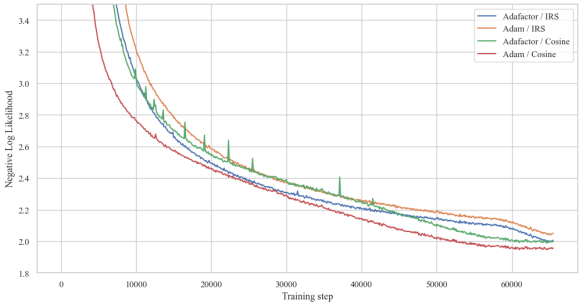

Figure 2: Training loss curves contrasting different optimizers and learning rate schedules.

models like FlanT5 (Chung et al., 2022), BLOOM (Scao et al., 2022), and Tk-Instruct (Wang et al., 2022). To assess the correctness of our fine-tuning setup, and the efficiency of our pre-training, we decided to reproduce the Tk-Instruct methodology.

3.4 Reproducibility

Ensuring that our work can be reliably replicated is a core focus of our methodology. To facilitate this, we have taken the following measures:

- **Model Weights:** We make the model's weights available on the HuggingFace Hub. These can be downloaded and used for fine-tuning on the SNI dataset with nanoT5.
- **Loss Curves:** We openly share both the pre-training and fine-tuning loss curves to provide insight into the model's learning dynamics.
- **Hyperparameters:** All hyperparameters used in our experiments have been released.
- **Environment and Hardware:** In our repository we offer comprehensive instructions on how to recreate our environment, including detailed information about our hardware. This encompasses specifications of our CPU and GPU, as well as the relevant driver versions.
- **Statistical Robustness:** To ensure the validity of our results, each experiment was conducted three times with different random seeds.

4 Results

4.1 Reproducing Pre-Training

By following the original experimental setup described above, we achieved a negative log-likelihood of 1.995 on the held-out set, which is slightly inferior to the reference.

In exploring alternative optimization methods, we tested the AdamW optimizer as a potential replacement for the original Adafactor. While AdamW theoretically promises greater training stability by directly estimating the second moment of the gradients (as opposed to Adafactor's low-rank approximation), our training with AdamW diverged. This behavior mirrors findings from a study on T5 pre-training (Havinga). Upon further investigation, we identified that matrix-wise learning rate (LR) scaling using its root mean square (RMS) [6] was the crucial element ensuring Adafactor's convergence. After augmenting AdamW with this extra LR scaling, which we will refer to as RMS scaling, it not only converged but also exhibited improved stability during pre-training and operated slightly faster, thanks to the direct retrieval of the second moment from memory instead of approximating it.

Nonetheless, when combined with the Inverse-Square-Root LR schedule, AdamW's performance was still outpaced by Adafactor. By replacing the ISR schedule with a Cosine LR Schedule, we achieved a superior negative log-likelihood of 1.953 on the held-out set, significantly surpassing Adafactor with the ISR schedule. The specific results of these experiments can be found in Table 2. A comparison of the training loss curves using different optimizers (Adafactor vs. AdamW) and schedules (ISR vs. Cosine) is provided in Figure 2.

4.2 Fine-Tuning Performance Across Different Pre-Training Durations

Our fine-tuning configuration strictly aligns with that of Tk-Instruct. However, there remains some ambiguity regarding whether Tk-Instruct was initialized from a regular checkpoint (google/t5-v1_1-base) or from a version specifically tailored for Lan-

[6] For more details please refer to (Shazeer and Stern, 2018), Section 8, titled "Relative Step Size"

Mixed Precision	Torch 2.0 compile	Grad Acc	Time per 1 Pre-training step	Total Pre-training time
FP32	No	2	∼4.10s	∼74.6h
TF32	No	2	∼1.39s	∼25.3h
BF16	No	2	∼1.30s	∼23.7h
TF32	Yes	2	∼0.95s	∼17.3h
BF16	Yes	1	∼0.56s	∼10.2h

Table 1: Efficiency metrics across various configuration settings during pre-training, with the "Total Pre-training Time" column referencing 2^{16} steps following the default config.

	Inverse-Square-Root	Cosine
Adafactor	1.995	1.993
AdamW	2.040	**1.953**

Table 2: Comparison of negative log-likelihood scores on the held-out set of C4 using different optimization methods and learning rate schedules.

guage Modelling (google/t5-base-lm-adapt). To cover all bases, we evaluated both, and successfully reproduced the original results.

Figure 1 presents a performance comparison of the model we trained in various time increments (ranging from 4 to 24 hours) against the original T5-base-v1.1 model weights from Huggingface Hub and its language modeling-adapted version. Notably, our model, trained for 16 hours on a single GPU, lagged by only 0.2 RougeL on average compared to the original T5-base-v1.1. This is an impressive result given the vast disparity in training data (the T5 paper indicates training on approximately 150x more data than we did). The language modeling-adapted checkpoint outperformed both the original T5-base-v1.1 model and ours, but this language modeling model adaptation extends beyond the scope of this study. A single fine-tuning step in our setup took approximately 0.18s, culminating in roughly an hour for the entire fine-tuning process.

4.3 Efficiency Statistics

Table 1 showcases the efficiency metrics from our pre-training experiments. It details the time taken for a single pre-training step and the overall pre-training time based on our default configuration desribed in Section 3.2. A noteworthy observation is that, because of the large batch size (128) used for pre-training, for numerical precisions other than BF16 we need to increase the number of gradient accumulation steps from 1 to 2.

Attempts at Boosting Efficiency In our pursuit of efficiency, we experimented with various strategies, albeit with limited success:

- **Optimization Algorithms**: We assessed the performance of recent optimizers like Lion (Chen et al., 2023) and Sophia (Liu et al., 2023). However, neither outperformed the AdamW with RMS scaling.

- **Positional Embeddings**: We tried replacing T5's learned relative positional embeddings with ALiBi (Press et al., 2021). Although such a switch had the potential to reduce the number of parameters, leading to faster training and inference rates, and paving the way for integrating Flash Attention (Dao et al., 2022) (currently limited to non-parametric bias), our trials revealed that training with ALiBi was more volatile and yielded suboptimal pre-training loss.

- **FP16 Precision**: Unfortunately, all our attempts using FP16 precision consistently diverged.

5 Conclusions

In this study, we demonstrated the feasibility of pre-training a substantial model like T5 under resource constraints, specifically using a single A100 GPU within a 24-hour timeframe. Through selection of optimization methods and configurations, we achieved results comparable to large-scale training settings. Our intention in sharing the codebase, configurations, and training logs is to bridge the gap between research accessibility and computational resource limitations in the NLP domain. We invite and welcome community suggestions to further refine and enhance our approach.

Moving forward, we aim to enrich our codebase by incorporating additional training objectives, such as those suggested by (Tworkowski et al., 2023; Tay et al., 2022), in hopes of further optimizing the training pipeline.

Acknowledgements

This work was supported by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) and the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. Symbolic discovery of optimization algorithms. *ArXiv*, abs/2302.06675.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher R'e. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *ArXiv*, abs/2205.14135.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Jonas Geiping and Tom Goldstein. 2022. Cramming: Training a language model on a single gpu in one day. *ArXiv*, abs/2212.14034.

Yeb Havinga. Pre-training dutch t5 models.

Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt J. Kusner. 2023. No train no gain: Revisiting efficient training algorithms for transformer-based language models. *ArXiv*, abs/2307.06440.

Andrej Karpathy. 2021. nanogpt. `https://github.com/karpathy/nanoGPT`. GitHub repository.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. 2023. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *ArXiv*, abs/2305.14342.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Frederick Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2017. Mixed precision training. *ArXiv*, abs/1710.03740.

Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Févry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam M. Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, and Colin Raffel. 2021. Do transformer modifications transfer across implementations and applications? *ArXiv*, abs/2102.11972.

Piotr Nawrot, Jan Chorowski, Adrian La'ncucki, and E. Ponti. 2022. Efficient transformers with dynamic token pooling. In *Annual Meeting of the Association for Computational Linguistics*.

Neptune team. 2019. neptune.ai.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang,

Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *ArXiv*, abs/2108.12409.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Teven Le Scao, Angela Fan, Christopher Akiki, Elizabeth-Jane Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Rose Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurenccon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa Etxabe, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris C. Emezue, Christopher Klamm, Colin Leong, Daniel Alexander van Strien, David Ifeoluwa Adelani, Dragomir R. Radev, Eduardo Gonz'alez Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady ElSahar, Hamza Benyamina, Hieu Trung Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jorg Frohberg, Josephine L. Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro von Werra, Leon Weber, Long Phan, Loubna Ben Allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, Mar'ia Grandury, Mario vSavsko, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad Ali Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto L'opez, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, S. Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal V. Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Févry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiang Tang, Zheng Xin Yong, Zhiqing Sun, Shaked Brody, Y Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre Franccois Lavall'ee, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aur'elie N'ev'eol, Charles Lovering, Daniel H Garrette, Deepak R. Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Xiangru Tang, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, S. Osher Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdenvek Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ananda Santa Rosa Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Olusola Ajibade, Bharat Kumar Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David M. Lansky, Davis David, Douwe Kiela, Duong Anh Nguyen, Edward Tan, Emily Baylor, Ezinwanne Ozoani, Fatim T Mirza, Frankline Ononiwu, Habib Rezanejad, H.A. Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jan Passmore, Joshua Seltzer, Julio Bonis Sanz, Karen Fort, Lívia Macedo Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, M. K. K. Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nourhan Fahmy, Olanrewaju Samuel, Ran An, R. P. Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas L. Wang, Sourav Roy, Sylvain Viguier, Thanh-Cong Le, Tobi Oyebade, Trieu Nguyen Hai Le, Yoyo Yang, Zachary Kyle Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Kumar Singh, Benjamin Beilharz, Bo Wang, Caio Matheus Fonseca de Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel Le'on Perin'an, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio

Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully A. Burns, Helena U. Vrabec, Iman I.B. Bello, Isha Dash, Ji Soo Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthi Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, María Andrea Castillo, Marianna Nezhurina, Mario Sanger, Matthias Samwald, Michael Cullan, Michael Weinberg, M Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patricia Haller, R. Chandrasekhar, R. Eisenberg, Robert Martin, Rodrigo L. Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Pratap Bharati, T. A. Laud, Th'eo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yashasvi Bajaj, Y. Venkatraman, Yifan Xu, Ying Xu, Yun chao Xu, Zhee Xao Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Noam M. Shazeer. 2020. Glu variants improve transformer. *ArXiv*, abs/2002.05202.

Noam M. Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ArXiv*, abs/1701.06538.

Noam M. Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *ArXiv*, abs/1804.04235.

Thomas Wolf Philipp Schmid Zachary Mueller Sourab Mangrulkar Marc Sun Benjamin Bossan Sylvain Gugger, Lysandre Debut. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier García, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2022. Ul2: Unifying language learning paradigms. In *International Conference on Learning Representations*.

Szymon Tworkowski, Konrad Staniszewski, Mikolaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Milo's. 2023. Focused transformer: Contrastive training for context scaling. *ArXiv*, abs/2307.03170.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, M. Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddharth Deepak Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hanna Hajishirzi, and Daniel Khashabi. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*.

Omry Yadan. 2019. Hydra - a framework for elegantly configuring complex applications. Github.