Journal of Machine Learning Research 24 (2023) 1-8 Submitted 6/23; Revised 10/23; Published 11/23

Scaling Up Models and Data with `t5x` and `seqio`

Lead Authors
Adam Roberts* adarob@google.com
Hyung Won Chung* hwchung@google.com
Gaurav Mishra* mishragaurav@google.com
Anselm Levskaya levskaya@google.com
James Bradbury jekbradbury@google.com

Technical Contributors, Advisors and Leadership
Daniel Andor †, **Sharan Narang**†, **Brian Lester**†, **Colin Gaffney**†, **Afroz Mohiuddin**†, **Curtis Hawthorne**†, **Aitor Lewkowycz**†, **Alex Salcianu**†, **Marc van Zee**†, **Jacob Austin**†, **Sebastian Goodman**†, **Livio Baldini Soares**†, **Haitang Hu**†, **Sasha Tsvyashchenko**†, **Aakanksha Chowdhery**†, **Jasmijn Bastings**†, **Jannis Bulian**†, **Xavier Garcia**†, **Jianmo Ni**†, **Andrew Chen**†, **Kathleen Kenealy**†, **Kehang Han**†, **Michelle Casbon**†, **Jonathan H. Clark**†, **Stephan Lee**†, **Dan Garrette**†, **James Lee-Thorp**†, **Colin Raffel** ‡, **Noam Shazeer**‡, **Marvin Ritter**‡, **Maarten Bosma**‡, **Alexandre Passos**‡, **Jeremy Maitin-Shepard**‡, **Noah Fiedel** §, **Mark Omernick**§, **Brennan Saeta**§, **Ryan Sepassi**§, **Alexander Spiridonov**§, **Joshua Newlan**§, **Andrea Gesmundo**§¶

Editor: Zeyi Wen

Abstract

Scaling up training datasets and model parameters have benefited neural network-based language models, but also present challenges like distributed compute, input data bottlenecks and reproducibility of results. We introduce two simple and scalable software libraries that simplify these issues: `t5x` enables training large language models at scale, while `seqio` enables reproducible input and evaluation pipelines. These open-source libraries have been used to train models with hundreds of billions of parameters on multi-terabyte datasets. Configurations and instructions for T5-like and GPT-like models are also provided. The libraries can be found at `https://github.com/google-research/t5x` and `https://github.com/google/seqio`.

Keywords: Large language models, data parallelism, model parallelism, data processing

∗. Equal Contributions
†. Technical Contributors
‡. Technical Advisors
§. Leadership
¶. Authors are ordered by impact within groups.

1. Introduction

Scaling transformers(Vaswani et al., 2017) to hundreds of billions of parameters has shown significant improvement, but training at such scale and consistently finetuning and prompting these models for downstream usage and evaluation requires a research-friendly and scalable software framework. In this paper, we introduce `t5x`, an open-source library to build Transformer models at scale by leveraging Jax's(Bradbury et al., 2018; Frostig et al., 2018) user-friendly NumPy-like(Harris et al., 2020) user interface and its powerful `jax.pjit` API for parallelism backed by XLA GSPMD(Xu et al., 2021).

Additionally, training at scale requires large datasets. We also introduce `seqio`, an open-source library for managing data pipelines and model evaluations. `seqio` builds on `tensorflow.data`, adds support for SPMD-based data parallelism and is compatible with popular modeling frameworks including JAX, TensorFlow(Abadi et al., 2015), and PyTorch(Paszke et al., 2019).

2. `t5x`

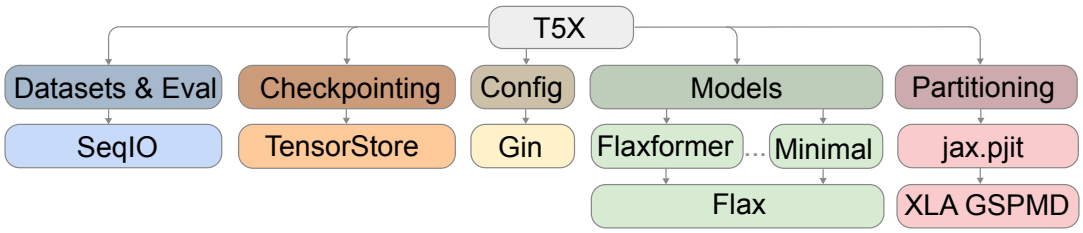

Figure 1: Overall structure of `t5x`, showing components used for principal functionalities.

Modular Design. Figure 1 illustrates the overall modular structure of `t5x`, highlighting the use of open-source libraries to implement different functionalities. For **datasets and evaluation** - `t5x` uses `seqio` to create reproducible "tasks", which we cover in detail in Section 3. For **checkpointing**, we built our own library utilizing `TensorStore`[1] as a tool for scalably reading and writing sliced tensors. This enables efficient management of checkpoints when parameters are distributed across multiple host processes. For **configuration**, we use Gin[2] for dependency injection, allowing users to inject hyperparameters, model objects, and other components (for example, custom checkpointer) without modifying the core library. This makes `t5x` easily configurable, supporting fast iteration over research ideas. For **model** implementation, `t5x` leverages specialized features in the Flax (Heek et al., 2020) library, built on JAX, which are described further below. For **Partitioning**, we use the XLA GSPMD partitioner (Xu et al., 2021) to automatically shard the computation graph and use `jax.pjit` as a frontend to interact with GSPMD, providing our own simplified API to allow users to parallelize over data, parameters, and activations, described further below. The modular structure allows users to replace these components with alternative standard and custom components.

1. `https://github.com/google/tensorstore`
2. `https://github.com/google/gin-config`

XLA GSPMD partitioning with `jax.pjit`. `t5x` supports both data parallelism and model parallelism to scale large models by defining orthogonal axes of the physical device mesh: `model` and `data`. Data parallelism involves splitting input data and intermediate activations over along the global batch axis, either by replicating parameters and optimizer state ("1D parameter partitioning") or sharding them over `data` ("2D parameter partitioning"). Model parallelism involves partitioning parameters and intermediate activations along axes other than the batch dimension. Replicating intermediate activations over `model` is referred to as "1D activation partitioning", while sharding them is "2D activation partitioning".

These options correspond to previously described parallelism techniques: 2D parameter partitioning is also known as ZeRO-3 (Rajbhandari et al., 2020) or fully sharded data parallelism; 1D activation partitioning is also known as Megatron (Shoeybi et al., 2019) and is the default in the Mesh TensorFlow Transformer (Shazeer et al., 2018); and 2D activation partitioning is the "fully sharded" case described in Xu et al. (2021). `t5x` supports flexible partitioning configurations, including these built-in options, using the Flax APIs described in the following section.

Model Implementation. `t5x` is compatible with Flax-based model implementations with minor caveats. User-defined logical axis annotations via `flax.partitioning.param_with_axes` are required for parameter and activation partitioning. These logical axes group tensor dimensions that must be partitioned in the same way, for example, "batch" (across examples in a batch), "kv" (across dimensions of key-value matrices in attention layers), and "head" (across heads in multi-headed attention). While XLA GSPMD automatically selects matching partitions for intermediate activations, users can override with `flax.partitioning.with_sharding_constraint` for better memory usage and inter-device communication. At runtime, users provide a map of logical axes to hardware axes (`model` or `data`). Alternatively, logical axes can be mapped to `None` to indicate replication across all devices.

A `flax.nn.module` implemented with these annotations is wrapped in a `t5x.BaseModel` subclass defining the loss, evaluation, and inference methods to make it compatible with the core `t5x` interface. `t5x` model support is flexible–layers and modules can be written directly with Flax or using higher-level libraries like Flaxformer[3]. Dependency injection with Gin enables easy swapping of models. Checkpoints from other libraries can be made compatible, including legacy T5 checkpoints[4] based on Mesh TensorFlow, which can be read directly by `t5x` or converted to the native `t5x` format for faster reading.

Example Models. We provide well-tested (validated by reproducing the T5 models from Raffel et al. (2020) originally implemented in Mesh TensorFlow) "Minimal" model implementations along with checkpoints for T5 (Raffel et al., 2020) and T5.1.1 (introduced after the paper), mT5 (Xue et al., 2021), ByT5 (Xue et al., 2022), a model configuration (without checkpoints) for a decoder-only architecture compatible with LaMDA (Thoppilan et al., 2022), and Scalable T5 - an implementation of T5.1.1 using `jax.scan` to significantly reduce compilation time and provide finer-grained control over activation memory. These use Flax with limited abstractions, closely following pedagogical Flax examples[5].

3. `https://github.com/google/flaxformer`
4. `https://github.com/google-research/text-to-text-transfer-transformer`
5. `https://github.com/google/flax/tree/main/examples`

GPU Support. We provide examples and instructions[6] to run `t5x` on GPUs in single-node and multi-node configurations, with optimizations for better throughput. More examples can be found in the NVIDIA Rosetta repository[7] which includes H100 FP8 support and performance improvements.

3. `seqio`

`seqio` is a data processing library for training, inference, and evaluation. It uses `tensorflow.data` for scalable pipelines, compatible with frameworks like JAX or PyTorch by easily transforming datasets to NumPy iterators. A key differentiator is the Task-based API illustrated in Figure 2, which associates data sources with preprocessing and evaluation. Feature converters transform task features into values passed to the model, making Tasks reusable across architectural variants such as encoder-decoder or decoder-only. Multiple Tasks can also be combined into a Mixture for multi-task training.

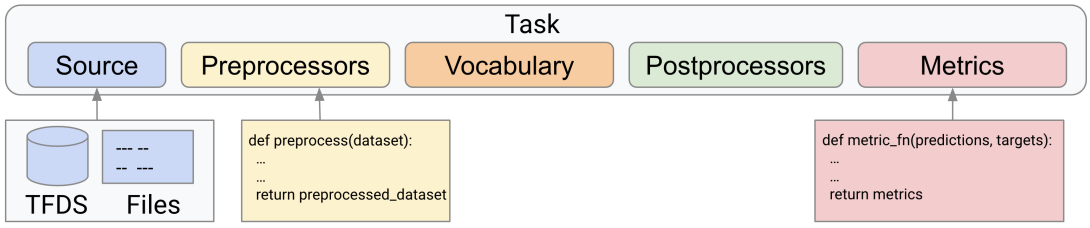

Figure 2: Structure of a `seqio` Task, highlighting customizable use of APIs.

4. Related Work

Previous Google-released libraries for training sequence models based on TensorFlow include Tensor2Tensor (Vaswani et al., 2018), Lingvo (Shen et al., 2019), and the Mesh TensorFlow (Shazeer et al., 2018)-based T5 (Raffel et al., 2020). Comparable projects from other research groups include model libraries like fairseq (Ott et al., 2019), large-scale parallelism libraries like FairScale (Baines et al., 2021), and libraries that include both, like DeepSpeed (Rasley et al., 2020) and Megatron (Smith et al., 2022).

Major differentiators of `t5x` are its use of JAX and Flax for model expression, its support for TPU (including TPU v4), and its Gin-based configuration system that allows users to modify any aspect of the model and training procedure. `t5x`'s native support for multi-host model parallelism allows reliably training models at massive scale. `t5x` doesn't support pipeline parallelism, a major component of systems like DeepSpeed, because the inter-chip network of TPUs has performance similar to within-node GPU interconnects but scales to thousands of chips, making model and data parallelism sufficient to train efficiently at scale.

6. `https://github.com/google-research/t5x/blob/main/t5x/contrib/gpu/`
7. `https://github.com/NVIDIA/JAX-Toolbox/tree/main/rosetta/rosetta/projects/t5x`

5. Project Status and Adoption

We started the project in the fall of 2020 and open sourced the library code in October 2021. During that time, `t5x` and `seqio` achieved widespread adoption by teams across Google: `t5x` has been launched on TPU hundreds of thousands of times at Google, and the total number of internal `t5x` and `seqio` users exceeds 1,000. Teams are using these libraries for research projects (from small-scale research to the largest language models trained at Google) and user-facing products. External adopters include academic and commercial users of Cloud TPUs, such as portions of the the Big Science project (Wang et al., 2022).

Users of `t5x` and `seqio` cite the usability and research-friendliness of the libraries as reasons for adoption. We are continuing to actively develop both libraries, prioritizing future work based on researcher needs and feedback.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Mandeep Baines, Shruti Bhosale, Vittorio Caggiano, Naman Goyal, Siddharth Goyal, Myle Ott, Benjamin Lefaudeux, Vitaliy Liptchinsky, Mike Rabbat, Sam Sheiffer, Anjali Sridhar, and Min Xu. Fairscale: A general purpose modular pytorch library for high performance and large scale training. `https://github.com/facebookresearch/fairscale`, 2021.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Roy Frostig, Matthew Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. 2018. URL `https://mlsys.org/Conferences/doc/2018/146.pdf`.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Na-*

ture, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL http://github.com/google/flax.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://aclanthology.org/N19-4009.

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. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140): 1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020. ISBN 9781728199986.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406703. URL https://doi.org/10.1145/3394486.3406703.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. Mesh-tensorflow: Deep learning for supercomputers. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran

Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/3a37abdeefe1dab1b30f7c5c7e581b93-Paper.pdf.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*, 2019.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv e-prints*, art. arXiv:1909.08053, September 2019.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. *arXiv e-prints*, art. arXiv:2201.11990, January 2022.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. LaMDA: Language Models for Dialog Applications. *arXiv e-prints*, art. arXiv:2201.08239, January 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*, 2018.

Thomas Wang, Adam Roberts, David Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective work best for zero-shot generalization? *arXiv e-prints*, 2022.

Yuanzhong Xu, HyoukJoong Lee, Dehao Chen, Blake Hechtman, Yanping Huang, Rahul Joshi, Maxim Krikun, Dmitry Lepikhin, Andy Ly, Marcello Maggioni, Ruoming Pang, Noam Shazeer, Shibo Wang, Tao Wang, Yonghui Wu, and Zhifeng Chen. GSPMD:

General and Scalable Parallelization for ML Computation Graphs. *arXiv e-prints*, art. arXiv:2105.04663, May 2021.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models. *Transactions of the Association for Computational Linguistics*, 10: 291–306, 03 2022. ISSN 2307-387X. doi: 10.1162/tacl_a_00461. URL https://doi.org/10.1162/tacl_a_00461.