Training Transformers Together

Alexander Borzunov*  
Max Ryabinin*  
HSE University, Yandex  
borzunov.alexander@gmail.com  
MRYABININ0@GMAIL.COM

Tim Dettmers*  
University of Washington  
dettmers@cs.washington.edu

Quentin Lhoest*  
Lucile Saulnier*  
Hugging Face  
quentin@huggingface.co  
lucile@huggingface.co

Michael Diskin  
HSE University, Yandex  
michael.s.diskin@gmail.com

Yacine Jernite  
Thomas Wolf  
Hugging Face  
yacine@huggingface.co  
thomas@huggingface.co

Editors: Douwe Kiela, Marco Ciccone, Barbara Caputo

Abstract
The infrastructure necessary for training state-of-the-art models is becoming overly expensive, which makes training such models affordable only to large corporations and institutions. Recent work proposes several methods for training such models collaboratively, i.e., by pooling together hardware from many independent parties and training a shared model over the Internet. In this demonstration, we collaboratively trained a text-to-image transformer similar to OpenAI DALL-E. We invited the viewers to join the ongoing training run, showing them instructions on how to contribute using the available hardware. We explained how to address the engineering challenges associated with such a training run (slow communication, limited memory, uneven performance between devices, and security concerns) and discussed how the viewers can set up collaborative training runs themselves. Finally, we show that the resulting model generates images of reasonable quality on a number of prompts.

Keywords: distributed training, volunteer computing, transformers, text-to-image, memory efficiency, communication efficiency, heterogeneous hardware, security

1. Introduction
Training state-of-the-art deep learning models is becoming ever more computationally demanding. One infamous example of this trend is transformers (Vaswani et al., 2017), a popular architecture widely used in NLP (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020), speech processing (Gulati et al., 2020; Li et al., 2019), and computer vision (Dosovitskiy et al., 2020; Touvron et al., 2021; Caron et al., 2021). Transformers benefit from having billions of parameters (Brown et al., 2020; Kaplan et al., 2020; Ott et al., 2018) and large-batch training (Popel and Bojar, 2018), which makes them dependent on large-scale training infrastructure (Narayanan et al., 2021; Shoeybi et al., 2019; Lepikhin et al., 2020).
Unfortunately, this kind of infrastructure can be prohibitively expensive, whether one buys the hardware or rents cloud resources (Turner; Li, 2020). As a result, most researchers simply cannot afford to conduct the necessary experiments to develop their ideas, which ultimately slows down scientific progress.

To make large-scale deep learning more accessible, recent work proposes to train these models collaboratively, i.e., to pool together the hardware from many independent parties and train a shared model over the Internet (Pascutto and Linscott, 2019; Ryabinin and Gusev, 2020; Kijsipongse et al., 2018; Atre et al., 2021; Diskin et al., 2021). Such work proposes general distributed algorithms for training on many devices with uneven compute capability and reliability. However, to make them practical, one must overcome several engineering challenges, such as slow communication, limited memory, and security concerns.

In this demonstration, we collaboratively trained a text-to-image transformer similar to DALL-E (Ramesh et al., 2021). Our contributions are the following:

- We modify the DALL-E model, making it suitable for training over the Internet using the method from Diskin et al. (2021) and the hivemind library (hivemind, 2020). We set up the infrastructure for such a training run and publish the training results.

- We provide a webpage\footnote{See https://training-transformers-together.github.io} explaining how to join the ongoing training run, address challenges related to collaborative training runs (slow communication, low memory budget, support of heterogeneous devices), and set up such a training run by yourself.

- We provide an interactive “calculator” that shows the memory consumed by different models in case of using various memory-efficiency techniques. Also, we present a tutorial on setting up dataset streaming and model compression using the datasets and bitsandbytes libraries (Lhoest et al., 2021; Dettmers et al., 2021).

2. Demonstration Contents

2.1. Main webpage

The central part of our demonstration is a webpage where people can explore the demonstration materials. The webpage describes the motivation behind collaborative training projects, the method for efficient training from Diskin et al. (2021), and the ongoing collaborative training of our adapted version of DALL-E (see Section 3). Here, we also show a plot of the training objective and the number of active participants.

Next, we provide instructions on how to join the training run using free cloud providers or their own GPU. This involves (1) joining a specific Hugging Face organization, where we can authenticate the users and measure their contribution, and (2) running a Jupyter notebook (Kluyver et al., 2016) with the training code. Our intention was that the user can explore our collaborative training environment through active participation while at the same time reading the detailed explanations of how it works. Here, we also provide the link to the interactive dashboard which shows the statistics and the leaderboard of contributors and provides further information about the training run, such as model checkpoints uploaded to the Model Hub, notebooks for inference, and links to the source code.

Then, we proceed to discuss the engineering challenges of collaborative training runs:
Training Transformers Together

- **Communication efficiency.** Most distributed training algorithms are designed for the networks inside HPC clusters with a 10–100 Gbit/s bandwidth. However, typical Internet connections are orders of magnitude slower (10–100 Mbit/s). To make training over the Internet practical, one can reduce the communication costs using large-batch training (You et al., 2020), gradient compression (Dettmers, 2015; Lin et al., 2018; Vogels et al., 2019; Tang et al., 2021), parameter sharing (Lan et al., 2020; Xue et al., 2021), and overlapping computation with communication (Ren et al., 2021).

- **Uneven device performance.** Traditional data-parallel training waits for the slowest device on every batch. Diskin et al. (2021) allow the devices to process different numbers of samples for a batch, while keeping the guarantees of synchronous training.

- **Memory efficiency.** Distributed training requires either storing all parameters and optimizer statistics on each participant, which is challenging in the case of low-end hardware, or using model parallelism which introduces another level of complexity. Fortunately, the first option is often viable if we reduce the memory consumption with 8-bit optimizers (Dettmers et al., 2021), by offloading the statistics to CPU, with gradient checkpointing or parameter sharing (Lan et al., 2020; Xue et al., 2021).

- **Dataset streaming.** Participants often cannot store or even download the whole dataset, since datasets used for pretraining transformers may contain hundreds of gigabytes of data. To address that, one can use dataset streaming tools, such as the datasets library (Lhoest et al., 2021).

- **Security.** Crucially, the participants only exchange tensors and never send code to be executed on each other’s computers. Since a malicious participant also could influence the training outcome by sending wrong tensors, we should either authenticate participants, as described in Diskin et al. (2021), and/or use gradient aggregation techniques robust to outliers (Karimireddy et al., 2020; Gorbunov et al., 2021).

Finally, we provide a recipe on how to combine all that and set up a new collaborative training run using the hivemind library (hivemind, 2020).

2.2. Memory calculator

The demonstration webpage includes an interactive “calculator” showing the benefits of various memory-efficiency techniques and their combinations. It can compute the consumption of RAM and GPU memory for BERT (Devlin et al., 2019), T5 (Raffel et al., 2020), GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), GPT-J (Wang and Komatsuzaki, 2021), and DALL-E (Ramesh et al., 2021) in case of using 8-bit optimizers, offloading the optimizer statistics to CPU, using gradient checkpointing and parameter sharing.

2.3. Tutorial on memory-efficiency techniques

The demonstration webpage refers to a tutorial on setting up dataset streaming with the datasets library (Lhoest et al., 2021) and model compression with the bitsandbytes library (Dettmers et al., 2021). The goal of the tutorial is to fine-tune the GPT-2 Large model (Radford et al., 2019) on the C4 dataset (Raffel et al., 2020) using only a low-end GPU, which is possible with the 8-bit Adam optimizer.
3. Collaborative Training Run

3.1. Model

For the practical example of a collaborative training run, we chose to train a text-to-image transformer similar to DALL-E (Ramesh et al., 2021), based on the code from Wang (2021). Specifically, we used a decoder-only transformer with 1024 hidden units and 64 layers, each of which uses 16 attention heads with a per-head state size of 64 ($\approx 1.1B$ parameters in total). We alternated the attention masks as in the original paper, i.e., repeated “row, column, row, row” masks until the last layer, which had the convolutional mask.

To improve communication and memory efficiency, we tied weights of all “row, column, row, row” layer groups (Lan et al., 2020) and tied the input and output embeddings (Press and Wolf, 2016), so the model uses $\approx 8x$ fewer parameters (but the same amount of compute). We also used reversible layers (Brügger et al., 2019) to reduce memory usage and rotary embeddings (Su et al., 2021) to improve training stability.

We replaced dVAE with VQ-GAN (Esser et al., 2021), since it has a smaller reconstruction error. We used the checkpoint with $f=8$ and the codebook size 8192. Finally, we used CLIP ViT/B-32 (Radford et al., 2021) to choose the best 4 out of 128 generated images.

3.2. Dataset

We trained the model on the first 100 million image-text pairs from LAION-400M (Schuhmann et al., 2021). We skipped $\approx 10\%$ images due to short captions, extreme aspect ratios, and NSFW labels.

Before training, we preprocessed all images with VQGAN and uploaded the VQGAN codes and captions, both compressed with Brotli (Alakuijala et al., 2018), to the Hugging Face Dataset Hub (Lhoest et al., 2021). During training, we streamed the compressed codes instead of the original images, thus consuming $\approx 18x$ less bandwidth.

3.3. Training procedure

We followed the distributed training procedure from Diskin et al. (2021) and used the 8-bit LAMB optimizer (You et al., 2020; Dettmers et al., 2021) offloaded to CPU. We used the linear training schedule with 31250 steps (the first 10% is the warm-up) and the peak learning rate of $2.5 \cdot 10^{-3}$. While exchanging gradients and parameters, we used the 8-bit quantization (Dettmers, 2015) for tensors with $\geq 2^{16}$ elements and the 16-bit precision for other tensors. Unlike the original paper, we did not use PowerSGD (Vogels et al., 2019).

3.4. Results

The training run lasted for 2.5 months and passed $\approx 80\%$ of the training schedule. Besides the authors, 37 volunteers have contributed for at least 10 minutes (see Appendix A).

During inference, we note that limiting sampling to top 256 logits or top logits whose probability sums up to $p = 0.75$ greatly improves the image quality. The final model generates realistic images for some prompts but fails to draw correct shapes for the others, while using the appropriate image style, textures, and colors (see Appendix B). We attribute that to the fact that our model is too small to remember the full diversity of images in LAION-400M. Still, the model can generalize to the concepts not present in the dataset.
References


Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training, 2021.


Elliot Turner. Estimate of GPT-3 training cost based on public cloud GPU/TPU cost models, from Elliot Turner’s personal page (accessed on May 29, 2020).


