

Training a T5 Using Lab-sized Resources

Manuel R. Ciosici

USC Information Sciences Institute

USA

manuelc@isi.edu

Leon Derczynski

ITU Copenhagen

Denmark

ld@itu.dk

Abstract

Training large neural language models on large datasets is resource- and time-intensive. These requirements create a barrier to entry, where those with fewer resources cannot build competitive models. This paper presents various techniques for making it possible to (a) train a large language model using resources that a modest research lab might have, and (b) train it in a reasonable amount of time. We provide concrete recommendations for practitioners, which we illustrate with a case study: a T5 model for Danish, the first for this language.

1 Introduction

Pre-training large language models (LLMs) is costly. There is an environmental cost (Strubell et al., 2019; Patterson et al., 2021); money, in running costs (Sharir et al., 2020); and potentially money in hardware, where the equipment to provide the needed computational power is often outside of a modest research lab’s budget.

This paper presents two contributions. The first is a set of recommendations to practitioners for training a modern LLM on modest hardware, limited data, and a reasonable compute time. The second is a demonstration of the recommendations resulting in DanT5, a T5 model for Danish, the first for this language.¹

2 Computational challenges

GPU memory is the most significant hurdle for training LLMs. State-of-the-art optimizers like Adam (Kingma and Ba, 2015; Loshchilov and Hutter, 2018) converge much faster than traditional Stochastic Gradient Descent due to tracking the first and second order momentum, but the performance comes at a cost. However,

to track the momentum, Adam must keep in memory two additional values for each parameter in the model, thus adding a 2X memory overhead.

Recent research has investigated **smaller-than-default number representations**, taking a floating point value (typically 32 bits) down to 16, 8, or fewer bits (Lin et al., 2017; Wang et al., 2018; Kalamkar et al., 2019). While these advances have the potential to ease and so democratize large model training, they require hardware and software support, which is still “just over the horizon”.

Using `float16` representations typically reduces a model’s GPU memory consumption by half (Micikevicius et al., 2018) and require a relatively modern GPU and software stack.² However, these reduced-range representations can overflow if training a model that expects a higher dynamic range. This can happen, for example, when fine-tuning a model initially trained using TPUs, which default to `bfloat16` for float representation.

The `bfloat16` representation should reduce compute needs (Kalamkar et al., 2019), though using it requires hardware³ and software support, and implementations across common software frameworks are still “not quite finished just yet”. Using even coarser representations, such as `bitsandbytes` (Dettmers et al., 2021) does, should also help, but these are just beginning to be implemented in popular NLP frameworks.⁴

One approach to training LLMs with modern GPUs is model parallelism, where a language model is split over multiple GPUs, with each GPU storing one or more layers. However, model parallelism brings the challenge of **data communication**. GPU-to-GPU communication typically goes through the slow PCI Express bus and sometimes

¹DanT5-small and DanT5-large are available at <https://huggingface.co/strombergnlp/dant5-small> and <https://huggingface.co/strombergnlp/dant5-large>.

²NVIDIA Pascal architecture GPU or newer, CUDA 8+. Modern PyTorch or TensorFlow.

³An NVIDIA Ampere GPU or newer.

⁴For example, the Hugging Face `transformers` library only added support for 8-bit optimizers starting with version 4.19, released on May 12, 2022.

the even slower CPU-to-CPU NUMA interconnect. Communication effects can be ameliorated via hardware and software. In hardware, one can use NVLink bridges to support high-speed GPU-to-GPU communication. NVLink bridges are a cost-effective solution for linking GPU pairs in a machine. NVLink bridges have a bandwidth of 50-100GB/s (depending on the GPU generation) compared to 31.5 GB/s for PCIe 4.0. On the software side, DeepSpeed (Ren et al., 2021; Rajbhandari et al., 2020, 2021) allows complex GPU offloading – supporting, for example, moving the expensive Adam optimizer from GPU memory and into (much larger) CPU memory. DeepSpeed CPU optimizers are implemented in efficient C++ code that uses the CPU’s native SIMD support. While offloading the optimizer to system memory increases the system memory requirements, RAM capacity is usually extendable at a reasonable cost. GPU memory is not extendable.

Given the challenges discussed above, we recommend the following when choosing compute hardware for training Large Language Models:

- Prioritize GPU memory over computation speed. LLM training is usually limited by: the effects of low memory; communication bandwidth between GPUs; and gradient accumulation over small batches.
- Install NVLink bridges to speed up GPU-to-GPU communication. Modern ML frameworks automatically use NVLink for GPU-to-GPU data transfer.
- Prioritize GPUs with support for `float16` and `bfloat16` in order to take advantage of optimizations whose widespread support is very close/nascent.
- Plan for system memory of 512GB or more. If the budget is tight, buy larger capacity RAM modules and leave empty RAM slots for future upgrades.
- Leaving PCIe slots for future GPU acquisitions can extend the life of a lab’s computing machines. For example, buying computers with support for eight GPUs even though budget only permits four, allows for staggered GPU acquisition.

3 Working with sub-gargantuan corpora

There is a large disparity in the size of corpora available for English and for other languages. While English has, for example, “The Pile” (Gao et al., 2020), an automatically-gathered corpus of over 800GiB, most other languages have to get by with datasets orders of magnitude smaller (Dunn and Adams, 2020).

Apart from size, corpus quality matters. Large corpora, such as the Common Crawl, include significant amounts of undesirable/abusive/illegal material (Luccioni and Viviano, 2021; Birhane et al., 2021), which leads to harmful model behavior (Bender et al., 2021); web-spider-derived corpora also tend to duplicate data, requiring deduplication, a deceptively complex and often challenging task (Lee et al., 2021). While these issues may have partly reduced impact with massive corpora, the effect of each unwanted piece of content is likely to be larger the smaller the corpus. When dealing with corpora of a few gigabytes or less, quality becomes much more important, and the advantages of automatic data collection can wane.

As a case study, we look at building a model for Danish using the T5 (Raffel et al., 2020) architecture, DanT5. Danish is a modestly-resourced language (Kirkedal et al., 2019). The original T5 used 34B sentence piece tokens from C4. The Norwegian T5,⁵ for a language similar to Danish, used the *Norwegian Colossal Corpus* (NCC, Kummer-vold et al. (2021), 7B words⁶). Unfortunately, there is no high-quality Danish language text corpus the size of C4 or even the NCC.

The largest freely available high-quality Danish corpus is the *Danish Gigaword Corpus* (DAGW, Strømberg-Derczynski et al. (2021)), a curated 1B word corpus. Like the NCC, DAGW assembles text spanning several domains, dialects, time periods, and modalities.

A single training pass over DAGW is too little training data for a modern LLM, even when warm-starting the training (see Section 4). To compensate for this, when training DanT5, we perform 10 epochs over the DAGW with *dynamic masking*. Before training, we split the corpus into sequences of 512 word pieces, but do not mask any token. During training, when each batch is assembled, we randomly mask 15% of each sequence’s tokens.

⁵<https://huggingface.co/NbAiLab/nb-t5-base-v2>

⁶<https://github.com/NBAiLab/notram>

The random, just-in-time masking results in different masks for each epoch. Exposing DanT5 to constantly changing masks attempts to compensate for DAGW’s size by generating multiple masked configurations for any given sequence in the source corpus. We mask 15% of the tokens in each sequence, following T5’s original training procedure.

4 Warm starting from a different language

Warm-starting is a shortcut to training a large language model (Rothe et al., 2020). Warm-starting does not train a model from scratch but continues training a pre-existing LLM. Therefore, warm-starting reduces computation costs and environmental impact by taking advantage of some of the computation that went into the source LLM. We used the original T5 checkpoints (Raffel et al., 2020) as the basis for DanT5. Warm-starting DanT5 from an English T5 also supports transfer learning as the two languages are somewhat related.⁷

Language models trained from pre-existing models are constrained to using the source model’s tokenizer and, therefore, its vocabulary of tokens. The English T5’s vocabulary contains sub-word tokens representing mostly English, with a minority of tokens specialized for Romanian, German, and French.⁸ Reusing the sub-word token vocabulary for Danish works but introduces excessive word splitting and leaves many tokens rarely used. Splitting words excessively results in long input sequences, reducing the amount of content that can fit T5’s maximum input length of 512. Rarely-used tokens occupy space in the embedding layer, displacing tokens relevant to the target language.

Several approaches have been proposed to modify a pre-trained language model’s vocabulary. But, most methods cannot convert a mostly-monolingual vocabulary into another mostly-monolingual vocabulary, as needed for DanT5. For multi-language machine translation models, Garcia et al. (2021) extend a highly-multilingual vocabulary with tokens for a new language, but assume that the vocabulary grows by only a tiny fraction. Chronopoulou et al. (2020) first train a language model using a tokenizer trained only on the source language. Then they train a tokenizer on the concatenation of the source and target language cor-

pora. The target language model then reuses word embeddings for the tokens appearing in both tokenizers. This approach cannot support adapting a pretrained language model for warm-start training in a new language. More recently, Xue et al. (2022) proposed doing away with tokenizers altogether and training T5 models directly on byte encodings. This method promises token-free models, but the requirement for a deeper encoder and the 30% larger training cost push this approach even further away from what can be achieved with the resources of a modest research lab.

Below, we present a new but simple way to adapt the vocabulary and parameters of English T5 to create a good (warm-)starting point for training DanT5. Our approach can generate warm-starting models for any language and transformer architecture (encoder-only, decoder-only, or encoder-decoder, like T5).

Warm-start tokenization translation. We first follow Raffel et al. (2020) and train a mostly Danish sentence-piece tokenizer on a mixture of 90% Danish and 10% English. We include English for the same reason as Raffel et al. (2020): so that we can later fine-tune the model to support machine translation, in this case Danish-English. DanT5’s tokenizer has exactly the same size as T5’s to make the warm starting process easier, but one could extend or reduce the vocabulary as needed. For warm-starting, we adapt the original T5 to the Danish tokenizer using a simple approach. First, we attempt to translate into English every token in the Danish tokenizer using Google Translate. We record the successful translations, and where the translation fails, we set the translation to equal the Danish word. We then create the starter DanT5 model by cloning T5 and adjusting the input embeddings.⁹ We set each embedding in DanT5 to the mean of the English embeddings of the English translation. Table 1 shows examples of Danish words translating to single or multiple English words and the contingency in case of translation failure (last row).

5 DanT5 Training

We follow T5’s original training procedure (Raffel et al., 2020) with a few exceptions. When we trained DanT5, there was no `bfloat16` support in DeepSpeed or `transformers`, so we trained our models with `float32` representations.

⁹T5’s input and output embeddings are identical, so we effectively adjust both the input and output embeddings.

⁷English is a West Germanic language, while Danish is North Germanic

⁸These tokens are present in the original T5 to allow the model to perform machine translation for some language pairs.

Danish	English	en tokens
doktor	doctor	doctor
dokumentet	the document	the; document
værsgo	here you go	here; you; go
Yndling	Favorite	Favor; it; e
Aarhus	ERROR!	A; ar; hus

Table 1: Example Danish to English token mappings. The last row shows our approach for cases where translation fails.

Optimizer. We use DeepSpeed’s AdamW implementation instead of Adafactor (Raffel et al., 2020; Shazeer and Stern, 2018). We use a learning rate of $4e - 3$, with warmup over the first 5 000 steps and a linear learning rate decay for the remaining. We chose AdamW over Adafactor due to the availability of a CPU-offloadable implementation of AdamW through DeepSpeed, an essential requirement when constrained by GPU memory. Even so, to achieve a batch size of 128, we accumulate gradients over 8 forward passes.

Batch Packing. Unlike T5’s original training, we do not pack batches with multiple sequences to obtain batches of roughly the same number of tokens. Instead, we dynamically pad each batch so that all inputs in a batch are padded to the maximal length within the batch. Therefore, length varies across batches but minimizes padding within each batch. As opposed to static padding, this technique speeds up computation on GPUs as it reduces the time wasted computing activations for pad tokens. While batch packing can be adapted to GPU computation, it is best suited for computations of TPUs which do not support dynamic padding.

Dynamic Masking. The original T5’s training consumed 1 trillion tokens from the *Colossal Clean Crawled Corpus (C4)*, a processed subset of *Common Crawl*. Since no equivalently large corpus exists for Danish, we used the *Danish Gigaword Corpus (DAGW)* (Strømberg-Derczynski et al. (2021)) together with *dynamic masking* (see Section 3).

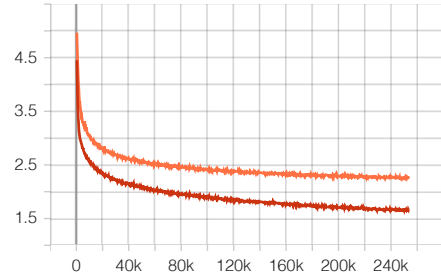
Hardware. We train DanT5 on a single machine with two AMD Epyc 7252 8-Core CPUs, 128 GB of RAM, and four NVIDIA A100 GPUs with 40 GB of memory each. We did not have NVLink bridges for this project.

6 Case Study Results

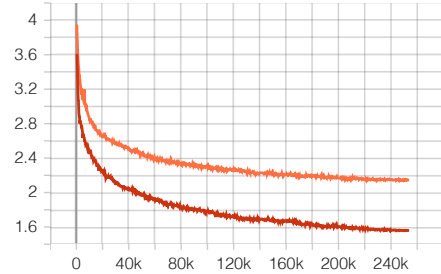
We trained two models: `dant5-small`, a 60M parameter model based on T5-small,

and `dant5-large`, a 770M parameter model based on T5-large. Both models trained for 10 epochs, as described in Section 3. Training `dant5-small` took 91 hours, while `dant5-large` trained in 508 hours.

Figure 1a indicates that `dant5-large` fit the training data better. The training loss decreases up to the end, suggesting that the models are still learning. The evaluation loss curve (shown in Figure 1b) closely tracks the training loss, indicating that the models are not overfitting the data.



(a) Training loss



(b) Evaluation loss

Figure 1: Training and Evaluation loss for `dant5-small` (orange, 60M parameters) and `dant5-large` (red, 770M parameters).

7 Conclusion

This paper presents a set of concrete recommendations to enable training LLMs using modest research lab resources, in a reasonable amount of time. We provide parameter values, software and hardware configuration strategies, and techniques for enhancing the use of available data. These recommendations are then demonstrated in the creation and release of “DanT5”, the first T5 model for Danish.

Acknowledgements

This work was conducted with support from the Independent Danish Research under the project VerifAI, 9131-00131B, and the Novo Nordisk Foundation project ClinRead, NNF19OC0059138.

References

- Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. [On the dangers of stochastic parrots: Can language models be too big?](#) In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21*, page 610–623, New York, NY, USA. Association for Computing Machinery.
- Abeba Birhane, Vinay Uday Prabhu, and Emmanuel Kahembwe. 2021. [Multimodal datasets: misogyny, pornography, and malignant stereotypes](#). *arXiv preprint arXiv:2110.01963*.
- Alexandra Chronopoulou, Dario Stojanovski, and Alexander Fraser. 2020. [Reusing a Pretrained Language Model on Languages with Limited Corpora for Unsupervised NMT](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2703–2711, Online. Association for Computational Linguistics.
- Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. [8-bit Optimizers via Block-wise Quantization](#). *arXiv*, abs/2110.02861.
- Jonathan Dunn and Ben Adams. 2020. [Geographically-Balanced Gigaword Corpora for 50 Language Varieties](#). In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2521–2529, Marseille, France. European Language Resources Association.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. [The Pile: An 800GB Dataset of Diverse Text for Language Modeling](#). *arXiv:2101.00027 [cs]*.
- Xavier Garcia, Noah Constant, Ankur Parikh, and Orhan Firat. 2021. [Towards Continual Learning for Multilingual Machine Translation via Vocabulary Substitution](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1184–1192. Association for Computational Linguistics.
- Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. 2019. [A Study of BFLOAT16 for Deep Learning Training](#). *arXiv*, arXiv:1905.12322.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A Method for Stochastic Optimization](#). In *Proc. ICLR (Poster)*.
- Andreas Kirkedal, Barbara Plank, Leon Derczynski, and Natalie Schluter. 2019. [The lacunae of Danish natural language processing](#). In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 356–362, Turku, Finland. Linköping University Electronic Press.
- Per E Kummervold, Javier De la Rosa, Freddy Wetjen, and Svein Arne Brygfeld. 2021. [Operationalizing a national digital library: The case for a Norwegian transformer model](#). In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 20–29, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.
- Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. [Deduplicating Training Data Makes Language Models Better](#). *arXiv:2107.06499 [cs]*. Accepted to ACL 2022.
- Xiaofan Lin, Cong Zhao, and Wei Pan. 2017. [Towards accurate binary convolutional neural network](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Ilya Loshchilov and Frank Hutter. 2018. [Decoupled Weight Decay Regularization](#). In *International Conference on Learning Representations*.
- Alexandra Luccioni and Joseph Viviano. 2021. [What’s in the box? an analysis of undesirable content in the Common Crawl corpus](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 182–189, Online. Association for Computational Linguistics.
- Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. [Mixed Precision Training](#). In *International Conference on Learning Representations*.
- David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. [Carbon Emissions and Large Neural Network Training](#). *arXiv:2104.10350 [cs]*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.

- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. [ZeRO: Memory optimizations toward training trillion parameter models](#). In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16.
- Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. [Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning](#). In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '21*, New York, NY, USA. Association for Computing Machinery.
- Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. [ZeRO-Offload: Democratizing Billion-Scale model training](#). In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564. USENIX Association.
- Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. [Leveraging Pre-trained Checkpoints for Sequence Generation Tasks](#). *Transactions of the Association for Computational Linguistics*, 8:264–280.
- Or Sharir, Barak Peleg, and Yoav Shoham. 2020. [The Cost of Training NLP Models: A Concise Overview](#). *arXiv:2004.08900*.
- Noam Shazeer and Mitchell Stern. 2018. [Adafactor: Adaptive Learning Rates with Sublinear Memory Cost](#). In *Proceedings of the 35th International Conference on Machine Learning*, pages 4596–4604. PMLR.
- Leon Strømberg-Derczynski, Manuel R. Ciosici, Rebekah Baglini, Morten H. Christiansen, Jacob Aarup Dalsgaard, Riccardo Fusaroli, Peter Juel Henriksen, Rasmus Hvingelby, Andreas Kirkedal, Alex Speed Kjeldsen, Claus Ladefoged, Finn Årup Nielsen, Jens Madsen, Malte Lau Petersen, Jonathan Hvithamar Rystrom, and Daniel Varab. 2021. [The Danish Gigaword corpus](#). In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 413–421, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.
- Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. [Energy and policy considerations for deep learning in NLP](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.
- Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. 2018. [Training deep neural networks with 8-bit floating point numbers](#). In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. [ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models](#). *Transactions of the Association for Computational Linguistics*, 10:291–306.