BERTchen: Training the best and most efficient German BERT model

Frederic Sadrieh
Hasso Plattner Institute
Prof.-Dr.-Helmert-Straße 2-3
14482 Potsdam
frederic.sadrieh@student.hpi.de

Abstract

Encoder-only models perform well in a variety of tasks. However, their efficient pretraining and language adaptation remain underexplored. This study presents a method for training efficient, state-of-the-art German encoder-only models. Our research highlights the inefficiency of BERT models, in particular due to the plateau effect, and how architectural improvements such as the MosaicBERT architecture and curriculum learning approaches can combat it. We show the importance of an in-domain tokenizer and investigate different pretraining sequence lengths and datasets. BERTchen can beat the previous best model GottBERT on GermanQuAD, increasing the F1 score from 55.14 to 95.1 and the exact match from 73.06 to 91.9. Our research provides a foundation for training efficient encoder-only models in different languages.[1]

1 Introduction

Traditionally, language models have been pretrained for several days to maximize performance (Devlin et al., 2019). There is a need for well-performing, efficient pretraining, since the same training times are required when adapting to a new language (Chan et al., 2020).

For the BERT model (Devlin et al., 2019) loss plateaus cause inefficient training (Nagatsuka et al., 2021). In these plateaus, the loss and model activations remain constant (Ainsworth and Shin, 2020). Thus, two ideas emerged to make BERT pretraining more efficient: Improving the architecture ((Liu et al., 2019), (Geiping and Goldstein, 2023), (Portes et al., 2023)), and a curriculum learning approach that increases sequence length during training to eliminate plateaus ((Nagatsuka et al., 2021), (Anonymous, 2023)). While the latter has been shown to perform well on BERT (Nagatsuka et al., 2021), it has not been tested in combination with new model architectures such as MosaicBERT (Portes et al., 2023). Therefore, we combine both ideas and improve the curriculum learning approach by dynamically switching to predefined datasets according to a plateau detection in the validation loss.

We seek to find the optimal pretraining sequence length and dataset for efficient pretraining. We investigate which approach is best for plateau elimination and the impact of tokenizer swapping.

Our main contributions include the creation and open-sourcing of the currently best and most efficient German BERT model. We show the best approach to eliminate plateaus is MosaicBERT. In addition, we develop best practices for training your own local BERT model, such as tokenizer swapping or the use of CulturaX (Nguyen et al., 2023).

2 Related Work

BERT BERT (Devlin et al., 2019) was pretrained for 4 days on 4 Cloud TPUs with Next Sentence Prediction (NSP) and Mask Language Modeling (MLM) (Devlin et al., 2019). RoBERTa (Liu et al., 2019) took the performance of BERT to a new level. Liu et al. found the NSP target to be redundant and removed it in favor of longer single-context patches. RoBERTa still requires 1024 V100 GPUs and a day of pretraining (Liu et al., 2019). These training times and resources are not feasible for many researchers. Therefore, there are papers that try to improve the performance while limiting the training time of BERT.

Geiping and Goldstein pretrain a BERT model in 24 hours on a RTX A6000 GPU. They achieve much better results than BERT trained with the same computational effort and comparable perfor-

[1]Our code is available at https://github.com/FSadrieh/BERTchen and the models can be downloaded at https://huggingface.co/frederic-sadrieh/BERTchen-v0.1

mance to the original BERT (Geiping and Goldstein, 2023) by omitting the NSP pretraining goal, a variety of architectural changes, and different hyperparameters.

MosaicBERT (Portes et al., 2023) is trained on 8 A100 GPUs for 1.13 hours and shows competitive performance with BERT. Portes et al. make pretraining faster by using FlashAttention (Dao et al., 2022). FlashAttention increases attention speed by reducing the amount of memory required. By "tiling" (Dao et al., 2022) there is no need to materialize the attention matrix since all computations can be done on the smaller tiles. In addition, MosaicBERT uses ALiBi (Press et al., 2022) as an alternative to positional embeddings. ALiBi includes token distances in the attention. Therefore it is able to handle longer sequences in inference than in training (Press et al., 2022). The further two tokens are apart, the higher the extra cost. In addition, MosaicBERT has GeGLU as the activation function (Shazeer, 2020). Using bfloat16 (Wang and Kanwar, 2019) and unpadding (Portes et al., 2023) results in even faster training. However, training on 8 GPUs simultaneously cannot be considered ultra-budgetary. Therefore, we will train the model with even less resources.

German BERT Early on, there were attempts to make BERT multilingual (Devlin et al., 2019) or to pretrain it with a German corpus ((Chan et al., 2019), (Chan et al., 2020)). First the German BERT (Chan et al., 2019) and then the better GBERT (Chan et al., 2020) were released by the deepset. GBERT follows the BERT architecture. As pretraining datasets, Chan et al. use a mixture of Wikipedia, the German OSCAR dataset and other human written texts such as speeches or legal texts. They pretrained the model for 7 days on a Google Cloud TPUs v3 with 8 cores (Chan et al., 2020).

There is a German RoBERTa variant called GottBERT (Scheible et al., 2020) which is pretrained following the original RoBERTa recipe on the German OSCAR dataset for 100k steps on a 256 core TPU pod (Scheible et al., 2020). GottBERT and GBERT have similar performance on standard downstream tasks (Scherrmann, 2023) and are, to our knowledge, the best performing open-source German BERT models currently available.

Plateau effect The plateau in pretraining loss (Nagatsuka et al., 2021) is a major obstacle to efficient BERT pretraining. Therefore, Nagatsuka et al. propose a curriculum learning approach in which they increase the sequence length over time. Starting from 64, they double the sequence length every n steps. This "static" change leads to a predictable training loop, but adds another hard to get right hyperparameter: after how many steps should one change? To still be computationally efficient, they decrease the microbatch size over time, which results in each sequence length having the same number of tokens per batch. The "dynamic" approach, as presented in (Anonymous, 2023), pretrains BERT on the maximum sequence length (512) until it detects a plateau by checking the training loss. If the loss remains the same for a certain number of steps, they switch to a dataset with a sequence length of 64 until the model leaves the plateau (Anonymous, 2023). The static approach has the advantage of training on the full context length and switching only when necessary, but it introduces non-deterministic behavior. Therefore, we introduce the hybrid approach, which trains on the same four sequence lengths as the static approach, but switches on a similar criterion as the dynamic approach.

Pretraining datasets BERT has been pretrained on a mix of Wikipedia articles and books (Devlin et al., 2019). The datasets promise to be of high quality as they are written and reviewed by humans. As the model size increases, more data is needed. Therefore, many practitioners train on web crawls such as C4 (Raffel et al., 2020). The quality of these datasets can not be controlled as easily. Therefore, a trend towards automatic data cleansing has emerged. New datasets are created using deduplication strategies (Tirumala et al., 2024) or content filtering based on data quality (Abadji et al., 2022). Oscar23.01 (Abadji et al., 2022) uses language identification to get better monolingual splits. Abadji et al. filter short, noisy or not child-friendly documents. Nguyen et al. extend this filtering by using deduplication, more filtering criteria, and intra-sample cleaning for CulturaX.

For efficient model training, we do not need the immense amount of data present in the crawls. Therefore, we can test what is the best dataset for smaller efficient models.

Masked Language Modeling Most papers keep the MLM objective unchanged ((Liu et al., 2019), (Geiping and Goldstein, 2023)), while some vouch for a higher masking percentage ((Portes et al., 2023), (Wettig et al., 2023)). Ankner et al. create a masking schedule that linearly decreases the

masking percentage over time. They show that this schedule provides, a broader training update at the beginning and better contextual understanding at the end. We use the idea of this schedule in combination with our hybrid approach.

3 Methods

Computational efficiency Geiping and Goldstein name throughput as the most important metric for efficient pretraining. We take advantage of known BERT pretraining speedups (see 2) by using the MosaicBERT architecture (Portes et al., 2023). MosaicBERT uses FlashAttention (Dao et al., 2022), which we upgrade to FlashAttention 2 (Dao, 2023). FlashAttention 2 requires fewer non-matmul FLOPs, making it even faster. Like MosaicBERT, we remove the [CLS] token and train only on the MLM objective to increase the number of usable tokens per sample. With these improvements over the base BERT architecture, we can increase the throughput from 190,000 tokens per second to about 250,000 tokens per second. Following, Chowdhery et al. (2023) we also report the model FLOPS utilization (MFU). We can increase the MFU from 40% for BERT to 65.87% for our model (see Appendix B, for the calculation).

To make better use of the tokens and increase the information throughput, we use the German tokenizer from Dobler and de Melo (2024). The tokenizer is a SentencePiece BPE tokenizer trained on a subset of OSCAR23.01 (Abadji et al., 2022), with a vocabulary size of 32,768 tokens (Dobler and de Melo, 2024). We replace the tokenizers special tokens with the standard BERT special tokens, except for the [CLS] token[2].

We use packing with a [SEP]-token between examples, to get the right sequence lengths for our datasets.

Hybrid approach In previous research ((Ainsworth and Shin, 2020), (Nagatsuka et al., 2021), (Anonymous, 2023)) and in our initial training runs, BERT was shown to have loss plateaus. These plateaus are long periods of training where the loss does not change, and the activation patterns remain constant (Ainsworth and Shin, 2020). Nagatsuka et al. give an intuition as to why plateaus can occur, the model first learns local attention, which can be difficult to learn on long sequence lengths. In the plateau, BERT is just "random guessing". When BERT starts to use context, it is able to reduce the loss dramatically. Using smaller sequence lengths in the beginning, allows the model to learn the context faster and BERT stops "random guessing" earlier.

Given our 4-hour training time and the constraint of a single GPU, the model would spend most of the training in the plateau and would not converge. To combat loss plateaus, we introduce a hybrid approach that combines the strategies of Nagatsuka et al. (2021) and Anonymous (2023).

Similar to early stopping, the loss is stored after each validation epoch. We define a patience p and a threshold δ. If for $p + 1$ steps the loss is not δ better than the previous best, the sequence length doubles. In Algorithm 1 the pseudocode of the hybrid approach is presented. The code should be executed after validation and initialized with switches $= 0$, patience $= 0$ and best $= \infty$.

Our approach can handle unpredictable training curves better than the static approach and removes the switch after n steps hyperparameter. Due to the fixed order of the datasets, we have better control over the data attributes than the dynamic approach. We use the same 4 sequence lengths (64, 128, 128, 512) as Nagatsuka et al..

Masking rate The masking rate is defined as the percentage of tokens that are replaced by the [MASK] token in the MLM objective. The pretrain loss is calculated only on these [MASK] tokens. On the one hand, a higher masking rate leads to a less noisy gradient update, since we are averaging over more tokens. On the other hand, if the masking rate is too high, the model will not be able to make meaningful predictions because it will not have enough context. Therefore, it is important to tune the masking rate. Following Ankner et al. (2024), one can decrease the masking rate over the training duration to prioritize first robust updates and then long context learning. We combine this idea with our hybrid approach to have a different masking rate per sequence length.

4 Experimental Results

4.1 Experimental setup

Implementation details For training, we use a single NVIDIA A100-SXM4-40GB or NVIDIA Ampere A100, 40 GB[3]. We limit pretraining to 4

[2]Our variant of the tokenizer can be found here: https://github.com/FSadrieh/BERTchen/tree/main/tokenizer

[3]We have found that the choice between nodes does not make much difference. Due to resource constraints, we use both.

hours and finetune for about 1 hour.

We use Hugging Face transformers (Wolf et al., 2020) to download the models and a PyTorch lightning trainer. Our training and evaluation pipeline is built on top of a template (Dobler et al., 2023), incorporating best practices from previous work into our experimental design.

Pretrain procedure For pretraining we use the MosaicBERT hyperparameters[4], except for the training goal, which we set to 2,500 to better estimate the number of steps the model will make. In addition, we test which sequence length is most advantageous for efficient pretraining (see Table 1). To have, as Portes et al., $524,288$ tokens per batch, we scale the batch size with the sequence length. We find that the training runs reach an average of 2,000 steps in 4 hours, resulting in about 1 billion tokens seen. For comparison, BERT has seen 128 billion tokens(Devlin et al., 2019), MosaicBERT 36.7 billion tokens(Portes et al., 2023), and LLAMA 3.1 13 trillion tokens during pretraining[5].

Pretraining Datasets We download and preprocess all datasets using the Hugging Face Dataset library (Lhoest et al., 2021). As mentioned in section 2, one can choose between human-written and controlled datasets, like Wikipedia, which contain relatively few samples, and the massive Internet crawls, which do not have the same quality and are therefore cleaned afterwards. Since we do not need as many tokens, we can evaluate which of these dataset types is best for efficient pretraining.

We use the German Wikipedia dataset (Foundation)[6], which contains 2.6 million samples, and a dataset of 3218 German prose literature (Fischer and Strötgen, 2017)[7]. The literature dataset contains German books and books translated into German. We have found that these texts contain a lot of racism and sexism, as they are uncensored works from the sixteenth to the twentieth century (Fischer and Strötgen, 2017). Therefore, we do not publish checkpoints from models trained on this dataset.

Following Portes et al., we use the C4 dataset (Raffel et al., 2020)[8]. As a cleaner dataset, we use OSCAR-23.01 (Abadji et al., 2022)[9]. The CulturaX dataset (Nguyen et al., 2023)[10] is currently the largest and cleanest Internet crawl, with a German subset. Since all of these datasets are created to pretrain large models for many steps, we will only see a fraction of the data in our training runs[11]. We take a seeded random 1.25 percent chunk from C4, a 20 percent chunk from Oscar2301, and a 2 percent chunk from CulturaX. Still, we do not exhaust the chunks during training.

Finetune Datasets Since we are training German models, we have chosen three German GLUE-like tasks. The simplest is the Germeval Task 2017 B (Wojatzki et al., 2017), a 3 class sentiment classification problem. The model has to classify whether a tweet is positive, neutral, or negative towards Deutsche Bahn in a dataset of about 21,000 tweets. In addition, we evaluate the GerMS-Detect subtask 1 from Germeval 2024. It is a 6-class sequence classification task that deals with potential sexism in tweets [12]. For each text, there are several labelers who rate the potential sexism from 0 (no sexism) to 5 (extreme sexism). In our implementation, the model should predict the majority label of the reviewers. As a standard GLUE task, we use GermanQuAD (Möller et al., 2021), which is the German equivalent of SQuAD (Rajpurkar et al., 2016). The model is given a context and a question and has to find out which, if any, sub-passage of the context answers the question.

The samples in the Germeval datasets are relatively short, with median sequence lengths of 48 and 34 tokens, respectively. GermanQuAD is much more challenging, with a median sequence length of 317 tokens. We do not truncate this task for all MosaicBERT models, as they can extrapolate using ALiBi.

[4]They can be found in Portes et al. (2023) and https://github.com/mosaicml/examples/tree/main/examples/benchmarks/bert/yamls (accessed: 07.07.2024)

[5]See https://huggingface.co/meta-llama/Meta-Llama-3.1-405B

[6]Downloaded from https://huggingface.co/datasets/legacy-datasets/wikipedia (accessed: 07.07.2024)

[7]Downloaded from https://figshare.com/articles/dataset/Corpus_of_German-Language_Fiction_txt_/4524680 (accessed: 07.07.2024)

[8]Downloaded from https://huggingface.co/datasets/allenai/c4 (accessed: 07.07.2024)

[9]Downloaded from https://huggingface.co/datasets/oscar-corpus/OSCAR-2301 (Accessed: 07.07.2024). We remove samples with the quality warnings: "adult", "noisy", "header", "footer", "tiny", "short_sentences" (following (Dobler and de Melo, 2024))

[10]Downloaded from https://huggingface.co/datasets/uonlp/CulturaX (accessed: 07.07.2024)

[11]The German subsets of the datasets have 1.8, 206, and 420 million samples, respectively.

[12]https://ofai.github.io/GermEval2024-GerMS/subtask1.html (accessed: 07.07.2024)

Finetuning procedure For finetuning, we use the hyperparameters of the closest GLUE task from Portes et al. (2023). That is, we finetune the Germeval tasks with the MosaicBERT hyperparameters for SST-2 and GermanQuAD with the SQuAD hyperparameters. We change the epochs of GermanQuAD to 20, because it is the most complex task and we see the biggest gains by continuing the training. [13]

Since we have removed the [CLS] token, we need to handle the classification tasks differently. We average the unpadded last hidden state of the model and then feed it through a normal classification head.

Evaluation Both Germeval tasks are evaluated using the classification accuracy (acc). GermanQuAD is evaluated using the F1 measure over the predicted and ground truth passages and the exact match (EM) metric, which is only satisfied if the passages match perfectly (Möller et al., 2021). Möller et al. provides human benchmarks for this task, with an F1 score of 89.5 and an exact match score of 66.4. [14]

Baseline We compare the performance of our model with the performance of GBERT (Chan et al., 2020) and GottBERT (Scheible et al., 2020), to our knowledge, the best German pretrained BERT models. In addition, we compare it to MosaicBERT (Portes et al., 2023), which is only trained on English sentences, but serves as a starting point for our optimization.

4.2 Results

Influence of sequence length As seen in Table 1, there is a big difference between the pretraining sequence lengths. Specifically for the GermanQuAD task, we get twice the performance on the two longer sequence lengths. GermanQuAD has a median sequence length of 317 tokens. Therefore, it will have many examples with longer contexts than the short pretraining sequence lengths. With ALiBi, the model is able to extrapolate, but as noted in Press et al. (2022), performance drops off at about twice the pretraining sequence length. We do not observe a strong difference in the Germeval task, since the median sequence lengths are much shorter than even the smallest pretraining sequence length. A sequence length of 512 will be used from now on.

Sequence length	64	128	256	512
Pretrain Batch size	8192	4096	2048	1024
GermanQuAD F1	49.1	48.9	92.7	**95.1**
GermanQuAD EM	41.8	42.5	88.8	**91.9**
Germeval 17 acc.	0.959	0.961	**0.962**	**0.962**
Germeval 24 acc.	0.89	0.898	0.906	**0.908**

Table 1: Downstream metrics for different sequence lengths after finetuning. All runs were pretrained on a subset of CulturaX (Nguyen et al., 2023). Note that we change the batch size depending on the sequence length to keep the tokens per batch at 524,288. A sequence length of 512 outperforms all other sequence lengths and is selected for further experiments.

Data source	Gq F1	Gq EM	G17 acc.	G24 acc.
Wiki	94.7	91.5	0.959	0.891
Books	61.1	54.68	0.96	0.906
Wiki+Books	92.5	88.5	0.958	0.895
C4	**96.4**	**93.6**	0.96	0.887
OSCAR23.01	87.4	83	**0.964**	0.902
CulturaX	95.1	91.9	0.962	**0.908**

Table 2: Downstream metrics after finetuning for different pretraining datasets. All runs were trained with a sequence length of 512 and a batch size of 1024. While Wikipedia, C4 and CulturaX perform well, Books underperform.

Influence of pretraining dataset The German Wikipedia dataset performs quite well, just behind the much newer datasets (see Table 2). For GermanQuAD this can be explained by the use of Wikipedia articles as context. The German books corpus underperforms. It should not be mixed with Wikipedia for these tasks, as the combination performs worse than Wikipedia alone.

For the larger crawled datasets, each dataset performs best on one of the tasks (see Table 2). While C4 performs well on the GermanQuAD task, OSCAR23.01 performs best on Germeval 17 and CulturaX on Germeval 24. CulturaX seems to be the most stable, and Oscar23.01 drastically underperforms GermanQuAD. It should be noted that we only use a small percentage of data from each of these datasets and cannot make any general statements.

For all subsequent experiments, CulturaX is chosen because it is the largest and would allow for

[13] All hyperparameter configurations can be found in Portes et al. (2023) and https://github.com/FSadrieh/BERTchen/tree/main/cfgs

[14] From here on, we will call the Subtask B from Germeval 2017: Germeval 17 or G17, and our implementation of the task from Germeval 2024: Germeval 24 or G24. In the tables we will shorten GermanQuAD to Gq

Strategy	Gq F1	Gq EM	G17 acc.	G24 acc.
Baseline	95.1	91.9	**0.962**	0.908
Hybrid	94.2	90.1	**0.962**	0.902
Hybrid sorted	84.1	79.4	0.96	**0.909**
Hybrid + mask	**95.5**	**92.2**	0.961	0.906
Hybrid + mask x2	84	79.4	0.955	0.904

Table 3: Downstream metrics for different hybrid strategies after finetuning. The base hybrid approach does not improve model performance, and sorting the data makes it worse. Together with a masking scheduler, performance is improved.

easy scaling of the model. We release the checkpoint trained on C4 as it proved to be the best on the most challenging task[15].

Hybrid approach In initial tests, we could confirm that the hybrid approach shortens the training time for BERT models, but is not necessary for MosaicBERT. MosaicBERT has no plateaus during pretraining due to architectural changes. Probably the switch to GeGLU (Shazeer, 2020) solves the plateau problem, as the ReLU function is suspected to be the cause (Ainsworth and Shin, 2020). In Table 3 it is shown that the hybrid approach performs worse than training on the full sequence length. As seen in Table 1, the shorter sequence lengths perform worse, which may explain the difference.

Sorting the data before chunking it into datasets could help avoid context fragmentation, as short sequences will be in the datasets with short sequence lengths, and vice versa. Table 3 shows a decrease in performance due to this measure. CulturaX contains a wide range of example lengths, with our subset starting at 16 sequence length examples. This results in the smaller datasets having a lot of context switches, which seems to be detrimental for training.

Masking Probability Changing the masking probability at each dataset switch beats baseline performance (see Table 3). We start with a masking ratio of 30 percent and decrease it by 5 percent on each dataset switch. Doubling the initial ratio and going from 60 to 30 percent does not work well because the model has very little context for prediction in the beginning.

Impact of tokenizer Using an in-domain tokenizer turns out to be crucial. The original English BERT tokenizer performs much worse than the German tokenizer from Dobler and de Melo (see Table 4). The difference is especially large for the GermanQuAD task, where the German tokenizer is able to compress the context better, resulting in a shorter sequence length and reducing the number of possible sub-passage starts.

Tokenizer	Gq F1	Gq EM	G17 acc.	G24 acc.
Baseline	**95.1**	**91.9**	**0.962**	**0.908**
BERT Tokenizer	29.5	1.6	0.9537	0.8908

Table 4: Downstream metrics for different tokenizers after finetuning. The German pretrained tokenizer, from Dobler and de Melo (2024), performs much better than the original English BERT tokenizer.

	Gq F1	Gq EM	G17 acc.	G24 acc.
Human	89.5	66.4	–	–
Ours	**95.1**	**91.9**	**0.962**	**0.908**
MosaicBERT	12.5	1.6	0.822	0.736
Our GBERT	46.964	34.786	0.802	0.738
GBERT	54.98	72.82	–	–
GottBERT	55.14	73.06	–	–

Table 5: Downstream metrics for different models after finetuning. Human performance as shown by Möller et al. (2021). We report the results for BERTchen, MosaicBERT, and GBERT finetuned with our hyperparameters. We also show the results of GBERT and GottBERT on GermanQuAD from Scherrmann (2023). Our model clearly outperforms not only every other model, but also the human baseline.

4.3 BERTchen

Final model architecture We combine our findings from the previous experiments and release three checkpoints of MosaicBERT models pretrained from scratch:

- The baseline model, which we used for all experiments. It has a sequence length of 512 and was pretrained on CulturaX.[16]
- The model pretrained with the same hyperparameters on C4[17]
- The model pretrained on CulturaX with the hybrid approach and the mask scheduler[18]

[15] See https://huggingface.co/frederic-sadrieh/BERTchen-v0.1-C4

[16] https://huggingface.co/frederic-sadrieh/BERTchen-v0.1

[17] https://huggingface.co/frederic-sadrieh/BERTchen-v0.1-C4

[18] https://huggingface.co/frederic-sadrieh/hybrid-BERTchen-v0.1

Comparison against baseline architecture Table 5 shows that the performance of GBERT finetuned with our hyperparameters is very poor. A possible explanation is that we use the hyperparameters tuned on MosaicBERT, which may not work well on BERT models. To provide a strong baseline, we show the GermanQuAD performance of GBERT and GottBERT as reported in Schermann (2023). Scherrmann tunes the hyperparameters for each model and finetunes five times to adjust for randomness. Note that since GBERT and GottBERT do not use ALiBi, we have to truncate the samples to allow for model prediction. MosaicBERT underperforms for GermanQuAD, which is probably related to the tokenizer, as seen in Table 4. BERTchen outperforms all other models and human performance by a wide margin.

5 Conclusion

Our research extends the discourse on efficient model training. We show that through better engineering, we can beat long-standing baselines while remaining more efficient.

Our main findings are that the hybrid approach helps BERT in pretraining, but is ultimately not needed, as we can eliminate plateaus with the better architectural choices of MosaicBERT. ALiBi's extrapolation is limited, with performance dropping rapidly beyond twice the pretrain sequence length. Unlike Portes et al. (2023), we do not vouch for training on shorter sequence lengths like 128. While there are many suitable datasets for pretraining, German downstream tasks are limited. Tokenizer switching is not as important for sequence classification tasks, but is crucial for longer question answering tasks.

If a future researcher wants to train their own efficient BERT-type model, we would advise them to:

- Use MosaicBERT (Portes et al., 2023) for efficiency gains
- Use a sequence length of 512
- Pretrain your own tokenizer based on (Dobler and de Melo, 2024)
- Use the CulturaX dataset, (Nguyen et al., 2023) as it is currently the largest and cleanest crawled dataset.

6 Future Work

One could extend this work by pretraining our model for longer periods of time, as we have no evidence that training for 4 hours is optimal. Training for a few hours or days would allow one to explore low-resource scaling laws of MosaicBERT.

In addition to scaling computation, one could also scale model size. Geiping and Goldstein find that larger models are more sample efficient. So what difference does it make to train a larger model in the same amount of time?

Since we are trying to allow researchers to quickly train models for new languages, a possible starting point would be the original MosaicBERT model. Can one improve performance by using its weights as initialization for pretraining?

In Table 1 we saw that a sequence length of 512 performed best. Could we even improve performance by scaling to higher sequence lengths? This might even improve the hybrid approach, as we saw in Table 1 that smaller sequence lengths underperformed, perhaps preventing the hybrid approach from reaching its full potential.

7 Limitations

We are limited by the use of a single language, as we wanted to test language adaptation outside of English and still be able to speak the language for manual quality control. Due to resource constraints, we use only one subset per pretraining dataset, which prevents us from making broader statements about the datasets. Of the three German downstream tasks, we found performance differences only for GermanQuAD. There is a general lack of German downstream datasets, which makes broad evaluation difficult. Without different model sizes and training times, it is impossible to explore the efficient scaling laws of BERT pretraining.

Another limitation is the cut-off after a certain training time. Due to resource constraints, we are working on a heterogeneous cluster where nodes are shared by many people. We have noticed small speed differences between nodes that could not be avoided.

References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoit Sagot. 2022. Towards a cleaner document-oriented multilingual crawled corpus. In *Proceedings of the thirteenth language resources and evaluation conference*, pages 4344–4355, Marseille, France. European Language Resources Association.

Mark Ainsworth and Yeonjong Shin. 2020. Plateau phenomenon in gradient descent training of ReLU networks: Explanation, quantification and avoidance. *Siam Journal On Scientific Computing*, 43:A3438–A3468.

Zachary Ankner, Naomi Saphra, Davis Blalock, Jonathan Frankle, and Matthew Leavitt. 2024. Dynamic masking rate schedules for MLM pretraining. In *Proceedings of the 18th conference of the european chapter of the association for computational linguistics (volume 2: Short papers)*, pages 477–487, St. Julian's, Malta. Association for Computational Linguistics.

Anonymous. 2023. Escaping The Plateau: Dynamic Context Length Adaptation for Efficient BERT Pretraining.

Branden Chan, Timo Möller, Malte Pietsch, and Tanay Soni. 2019. Open Sourcing German BERT Model.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model. In *Proceedings of the 28th international conference on computational linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Konstantin Dobler and Gerard de Melo. 2024. Language adaptation on a tight academic compute budget: Tokenizer swapping works and pure bfloat16 is enough. In *2nd workshop on advancing neural network training: Computational efficiency, scalability, and resource optimization (WANT@ICML 2024)*.

Konstantin Dobler, Maximilian Schall, and Oliver Zimmermann. 2023. nlp-research-template.

Frank Fischer and Jannik Strötgen. 2017. Corpus of German-Language Fiction (txt). Artwork Size: 428666655 Bytes Pages: 428666655 Bytes.

Wikimedia Foundation. Wikimedia downloads.

Jonas Geiping and Tom Goldstein. 2023. Cramming: Training a language model on a single GPU in one day. In *International conference on machine learning*, pages 11117–11143. PMLR.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 conference on empirical methods in natural language processing: System demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. ArXiv: 2109.02846 [cs.CL].

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Timo Möller, Julian Risch, and Malte Pietsch. 2021. GermanQuAD and GermanDPR: Improving non-english question answering and passage retrieval. ArXiv: 2104.12741 [cs.CL].

Koichi Nagatsuka, Clifford Broni-Bediako, and Masayasu Atsumi. 2021. Pre-training a BERT with curriculum learning by increasing block-size of input text. In *Proceedings of the international conference on recent advances in natural language processing (RANLP 2021)*, pages 989–996, Held Online. INCOMA Ltd.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. 2023. CulturaX: a cleaned, enormous, and multilingual dataset for large language models in 167 languages. ArXiv: 2309.09400 [cs.CL].

Jacob Portes, Alexander R Trott, Sam Havens, Daniel King, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. 2023. Mosaicbert: How to train bert with a lunch money budget. In *Workshop on efficient systems for foundation models@ ICML2023*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. ArXiv: 2108.12409 [cs.CL].

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Raphael Scheible, Fabian Thomczyk, Patric Tippmann, Victor Jaravine, and Martin Boeker. 2020. GottBERT: a pure german language model. ArXiv: 2012.02110 [cs.CL].

Moritz Scherrmann. 2023. German FinBERT: a german pre-trained language model. ArXiv: 2311.08793 [cs.CL].

Noam Shazeer. 2020. GLU Variants Improve Transformer. *CoRR*, abs/2002.05202. ArXiv: 2002.05202.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. 2024. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36.

Shibo Wang and Pankaj Kanwar. 2019. BFloat16: The secret to high performance on Cloud TPUs.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2023. Should you mask 15% in masked language modeling? In *Proceedings of the 17th conference of the european chapter of the association for computational linguistics*, pages 2985–3000, Dubrovnik, Croatia. Association for Computational Linguistics.

Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. GermEval 2017: Shared task on aspect-based sentiment in social media customer feedback. In *Proceedings of the GermEval 2017 – shared task on aspect-based sentiment in social media customer feedback*, pages 1–12, Berlin, Germany.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Hybrid approach algorithm

Algorithm 1 Hybrid approach

```
if switches < 3 then
    if val_loss+δ < best then
        wait ← 0
        best ← val_loss
    else
        wait ← 1+wait
        if wait ≥ p then
            double_sequence_length()
            switches ← 1+switches
            wait ← 0
        end if
    end if
end if
```

B Model FLOPS utilization calculation

Model FLOPS utilization (MFU) (Chowdhery et al., 2023) is a model and hardware agnostic efficiency metric. We base our calculation on Portes et al. (2023), which gives MFU $= \frac{6 \cdot T \cdot P}{n \cdot F}$, where T is the tokens per second, P is the model parameters, n the number of GPUs, and T the theoretical performance of the GPU.

Our one A100 GPU has a theoretical performance of 312 TFLOPS per second (Chowdhery et al., 2023)[19]. BERT$_{BASE}$ has 110 million parameters, while MosaicBERT$_{BASE}$ has 137 million parameters (Portes et al., 2023). The tokens per second were 190,000 for BERT$_{BASE}$ and roughly 250,00 for MosaicBERT$_{BASE}$ (see section 3). This yields a MFU of 40.19% for BERT$_{BASE}$ and 65.87% for MosaicBERT$_{BASE}$.

[19] See also https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf (Page 15; accessed: 29.07.2024)