Benchmarking down-scaled (not so large) pre-trained language models

Matthias Aßenmacher♠ **Patrick Schulze**♣ **Christian Heumann**♠

Department of Statistics
Ludwig-Maximilians-Universität
Ludwigstr. 33, D-80539 Munich, Germany
♠{matthias,chris}@stat.uni-muenchen.de, ♣pa.schulze@campus.lmu.de

Abstract

Large Transformer-based language models are pre-trained on corpora of varying sizes, for a different number of steps and with different batch sizes. At the same time fundamental components, such as the pre-training objective or architectural hyperparameters, are modified. In total, it is therefore difficult to ascribe changes in performance to specific factors. Since searching the hyperparameter space over the full systems is too costly, we pre-train down-scaled versions of several popular Transformer-based architectures on a common pre-training corpus and benchmark them on a subset of the GLUE tasks (Wang et al., 2018). Specifically, we systematically compare three pre-training objectives for different shape parameters and model sizes, while also varying the number of pre-training steps and the batch size. In our experiments MLM + NSP (BERT-style) consistently outperforms MLM (RoBERTa-style) as well as the standard LM objective. Furthermore, we find that additional compute should be mainly allocated to an increased model size, while training for more steps is inefficient. Based on these observations, as a final step we attempt to scale up several systems using compound scaling (Tan and Le, 2019) adapted to Transformer-based language models.

1 Introduction

The introduction of the Transformer (Vaswani et al., 2017) together with the application of transfer learning (Thrun and Pratt, 1998) has led to major advances in Natural Language Processing (NLP). While many different lines of research exist, most attention is generally paid to the largest systems which often reach new state-of-the-art (SOTA) results. The current trend is to scale up such systems to ever new orders of magnitude: 213M parameters in the Transformer, 300M parameters in BERT (Devlin et al., 2019), 1.5B parameters in GPT-2 (Radford et al., 2019) and 175B in GPT-3 (Brown et al., 2020). Since these models are pre-trained on corpora of widely varying sizes, for a different number of training steps and with different batch sizes, comparability suffers (Aßenmacher and Heumann, 2020). At the same time, new systems often apply fundamentally different methods, such as using a different pre-training objective or modified architectural hyperparameters. While altering multiple components simultaneously can help achieve new SOTA results, which is an important endeavor, it is difficult to disentangle the effects of the various factors. Though there exist various ablation studies, these often show only a small excerpt from the broad spectrum of experimental opportunities and can thus not provide a comprehensive picture. In this work, we conduct a systematic study of three Transformer-based architectures with respect to several pre-training hyperparameters.

2 Related work

One line of research empirically derives generalization results for large neural NLP systems. Rosenfeld et al. (2019) study how the generalization error of language models (LMs) depends on model and data set size. Regarding model size, they provide an approximation of the test loss, assuming that a LM is scaled with respect to a pre-defined scheme, such as increasing solely the embedding dimension. A related but more comprehensive study was conducted by Kaplan et al. (2020), examining power laws of the test loss when scaling large neural LMs with respect to a broad variety of different dimensions. These dimensions include architectural hyperparameters, model size, data set size, number of training steps and batch size. A central question in their work is how these factors can be combined to attain an optimal performance given a fixed amount of compute.

Compute efficient training is also investigated by Li et al. (2020), recognizing that an optimal allocation of computational resources is crucial for improving model performance. Considering Masked Language Modeling (MLM) pre-training, Li et al. (2020) examine the optimal choice of number of training steps and batch size in the relation to the model size. In a large-scale study, Raffel et al. (2019) cover an even broader variety of modeling scenarios than Kaplan et al. (2020), but train a much smaller number of systems per scenario. For instance, they include several variants of the Transformer, different pre-training objectives and various fine-tuning strategies in their analysis. Finally, based on their observations, Raffel et al. (2019) also scale-up a system to 11B parameters.

3 Materials and Methods

Pre-training data We pre-train all models on WikiText-103[1] (Merity et al., 2016), a large-scale text corpus for training and evaluating language models on long-range contexts, which has served as an evaluation data set (Radford et al., 2019; Dai et al., 2019; Shoeybi et al., 2019) as well as for pre-training (Howard and Ruder, 2018). We pre-train all models on the training set of WikiText-103, which allows for learning long-range dependencies (Rae et al., 2019). The validation set is employed to compare different architectures by their validation loss during pre-training. WikiText-103 is much smaller than most pre-training corpora of modern language models. For instance, Devlin et al. (2019) trained BERT on a $3,300M$ words corpus, which is approximately 32x the size of WikiText-103. Aside from this, pre-training data sets of different models often vary considerably in size, which makes fair comparisons difficult (Aßenmacher and Heumann, 2020). Pre-training on the same corpus allows us to exclude the amount and quality of pre-training data as confounding factors when evaluating the different model components.

Models We compare three different model types: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and GPT-2 (Radford et al., 2019). BERT is a bidirectional Transformer encoder which is trained with both MLM as well as Next Sentence Prediction (NSP). Its direct successor RoBERTa relies on the exact same architecture and differs from BERT solely in the pre-training procedure. Amongst other

changes, Liu et al. (2019) abandoned the NSP objective and introduced a dynamic masking[2] procedure for the MLM objective[3]. GPT-2 is a Transformer decoder, and thus a unidirectional model, trained with the standard LM objective.

Since we train a multitude of down-scaled versions for each model type, thus modifying the specifications of the original models, we introduce the following conventions: We label models trained with MLM & NSP as *BERT-style*, models trained with MLM as *RoBERTa-style*, and models trained with LM as *GPT-2-style*. Alongside with the pre-training objectives, we also use the respective tokenizers of the different models. This means using byte-level BPE (Radford et al., 2019) for *RoBERTa-* and *GPT-2-style* and the WordPiece algorithm (Schuster and Nakajima, 2012) for *BERT-style* models, all of them exhibiting a uniform vocabulary size of 30,000 tokens

Fine-tuning data We fine-tune and evaluate our systems on GLUE (Wang et al., 2018). We mainly compare performances on MNLI (Williams et al., 2017), QQP (Shankar et al., 2017) and QNLI (Wang et al., 2018), which are the three largest GLUE tasks, since the results on these tasks are the most reliable. In particular, we therefore calculate the average score over the validation set performances of the three tasks, which we denote by *GLUE-Large*. For MNLI, we consider only the matched validation set when calculating this score. Whenever meaningful results for the two next largest data sets SST-2 (Socher et al., 2013) and CoLa (Warstadt et al., 2019) were achieved[4], those will also be reported.

Training details Hyperparameters and the pre-training/fine-tuning procedure are largely adopted from the original models (cf. Appendix A and B).

4 Experiments

4.1 Comparison of different Shapes[5]

In computer vision it has been observed that the performance of a neural network strongly depends on the choice of architectural hyperparameters, such

---

[1] www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/

[2] We also use dynamic masking throughout this study.

[3] There were further alterations, none of which are crucial for our experiments since we are using fixed pre-training data sets, batch sizes, learning rates, etc. for better comparability.

[4] For the smaller model sizes the performance on these smaller data sets did not significantly differ from zero.

[5] There exist several other choices, but examining the entire spectrum of possible shapes is out of the scope of this study.

as width or depth (Tan and Le, 2019). In contrast, Kaplan et al. (2020) observed a similar LM test loss over a wide range of shape parameters. Similarly, for MLMs, Li et al. (2020) found that the validation loss does not depend strongly on the model shape. This holds true also for the MNLI validation accuracy of fine-tuned systems.

In this study, we examine the impact of three different architectural hyperparameters in Transformer-based models: *depth*, *width* and the *number of attention heads*. Depth is given by the number of layers L. Stacking many layers in Transformer-based systems can be somewhat inefficient and does not always lead to a considerable increase in performance (Lan et al., 2019). Width corresponds to the embedding dimension H. Increasing H has in general produced slightly better results than increasing L in Transformer-based systems (Lan et al., 2019; Raffel et al., 2019; Li et al., 2020). Attention Heads are used to discriminate between different regions of the embedding space. In most applications of the Transformer, the number of attention heads A is set in fixed relation to H, such as $H = 64 \times A$. Decreasing performance has been reported for larger ratios (Vaswani et al., 2017; Brown et al., 2020).

4.2 Model Size, Training Steps and Batch size

Several recent studies have investigated the problem of compute efficient training of Transformer-based systems (Raffel et al., 2019; Li et al., 2020; Kaplan et al., 2020). The consensus among these studies is that, under a restricted budget, optimal performance is achieved by training very large models and stopping training well before convergence. Furthermore, additional compute should rather be used to increase the batch size instead of training for more steps. To examine convergence characteristics, we monitor the pre-training validation loss of several systems and test how this loss corresponds to different model sizes and shapes. Additionally, we conduct experiments regarding the effect of the batch size and the number of training steps. In particular, we evaluate how the training time and the model performance depend on both factors.

4.3 Definition of the Model Size

We follow Kaplan et al. (2020) and use the approximate number of non-embedding parameters to define the model size, which we denote as N_{model}. Since the share of embedding parameters decreases for larger models, similarly to Kaplan et al. (2020)

we expect that discarding the number of embedding parameters allows for better generalization of our results to large models. Another advantage of defining the model size as the number of non-embedding parameters is that it is closely linked to the number of (non-embedding related) floating point operations (FLOPs) per input token (Kaplan et al., 2020). This enables us to design benchmarking scenarios by training different models of comparable size, which at the same time require roughly similar amounts of computation.

Omitting biases and other sub-leading terms, the number of non-embedding parameters is given by

$$N_{\text{model}} = 12LH^2, \tag{1}$$

assuming that queries, keys and values are all transformed to dimension $\frac{H}{A}$ and the feed-forward dimension is $4H$. For a more in-depth explanation, please see Appendix E.

5 Results[6]

We start by evaluating how varying single shape dimensions affects the performance on GLUE-Large for the three different pre-training objectives (cf. Sec. 5.1). This aims at investigating whether the performance gain diminishes after a certain level, comparing how the performance changes when scaling different dimensions, and examining whether models with different pre-training objectives respond differently to single-dimension scaling. Subsequently in Section 5.2, we change multiple shape dimensions simultaneously to investigate whether the different dimensions depend on each other. In Sections 5.3 and 5.4 we study how to train efficiently by varying the model size, the number of training steps and the batch size. In Section 5.5 we put together our observations from the previous sections and scale networks to different sizes.

5.1 Scaling Single Shape Dimensions

In this section, we separately scale L and H, while holding all other dimensions constant. As shown in Figure 1, BERT-style systems perform significantly better than GPT-2-style and RoBERTa-style systems on GLUE-Large, contrary to the results of Liu et al. (2019) and in line with the original findings of Devlin et al. (2019).

Observation 1 *The pre-training objective has a large impact on the performance of a fine-tuned*

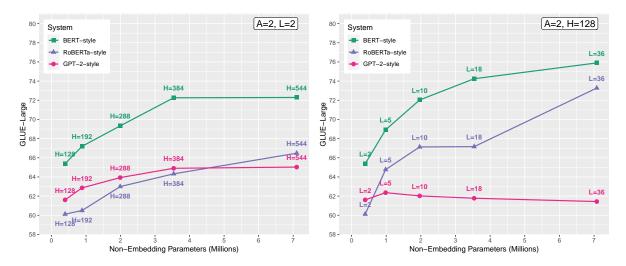[6]Source code: https://github.com/PMSchulze/NLP-benchmarking

Figure 1: Average score on GLUE-Large, when varying H (left) vs. when varying L (right). For detailed performance values on the single tasks, see Table 5 and Table 6 in Appendix C.

system. Pre-training with the combination of MLM & NSP achieves the best results on sentence-pair tasks[7], while pre-training with the unidirectional LM objective shows in general the worst performance.

Furthermore, for BERT-style systems the average performance is a relatively smooth function of the model size. Scaling up H results in an increasing performance, which saturates at approximately 72%, while for L we cannot clearly see this saturation (even not at 75%). For RoBERTa-style systems, the difference between scaling L and H individually is much larger. Furthermore, a saturation (as for BERT-style systems) can not be observed.[8] For GPT-2-style systems, the average score slightly increases when scaling the embedding size, but interestingly, stacking more layers shows no positive effect at all. This suggests that GPT-2-style systems require more pre-training data compared to BERT-style and RoBERTa-style systems.

Observation 2 *In most cases, the performance of a fine-tuned system increases up to a certain level when scaling either width or depth, but the progression depends strongly on the pre-training objective.*

5.2 Scaling Multiple Shape Dimensions

We next examine whether the performance can be improved by scaling multiple dimensions at the same time. First, we increase both H and L and

compare the performance with the results from Section 5.1. Fig. 2 shows that for RoBERTa-style and BERT-style systems, scaling both dimensions significantly improves the performance on GLUE-Large.

Observation 3 *Scaling multiple shape dimensions can lead to a better performance than scaling single dimensions.*

Therefore, we conclude that the shape dimensions are not independent of each other. For GPT-2-style systems, however, we do not observe a performance increase, as shown in Table 1.

BERT-Style				Validation Set Performance
A	H	L	N_{model}	GLUE-Large
2	204	7	3,495,744	77.1
2	256	9	7,077,888	78.6
8	544	2	7,102,464	78.4
GPT-2-Style				**Validation Set Performance**
A	H	L	N_{model}	GLUE-Large
2	204	7	3,495,744	63.6
2	256	9	7,077,888	63.8
8	544	2	7,102,464	66.0
RoBERTa-Style				**Validation Set Performance**
A	H	L	N_{model}	GLUE-Large
2	204	7	3,495,744	72.9
2	256	9	7,077,888	75.0
8	544	2	7,102,464	70.9

Table 1: Performance on GLUE-Large when increasing multiple shape dimensions at the same time.

So far, we did not increase A when scaling H and observed that, without using more attention heads, wide systems perform worse than deep systems (cf. Fig. 1). To evaluate whether a larger num-

---

[7]Note that this does not necessarily generalize to other languages or other types of tasks.

[8]Note that the relatively low average score for the 18-layer RoBERTa-style system, shown in the right plot of Figure 1, is due to a weak performance on the QNLI task.
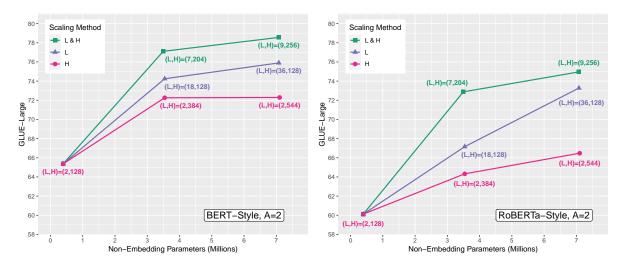
Figure 2: Performance on GLUE-Large when increasing multiple shape dimensions.

ber of attention heads can boost the performance of wide systems, we re-implement our widest systems with $A = 8$ attention heads, which corresponds to $\frac{H}{A} = 68$. We observe that the score of the widest system on GLUE-Large improved substantially by doing so (cf. Fig. 1 and Tab. 1). In particular, when using $A = 8$ instead of $A = 2$, the wide BERT-style system ($A = 8, H = 544, L = 2$) performs even better than the deep BERT-style system of comparable size ($A = 2, H = 128, L = 36$). Furthermore, as also shown in Table 1, the wide BERT-style system (with increased A) performs close to the balanced one ($A = 2, H = 256, L = 9$).

Observation 4 *The fine-tuning performance can be similar over a wide range of shapes. For BERT-style systems, wide systems perform slightly better than deep systems, if the number of attention heads is adapted to the embedding dimension.*

In contrast to BERT-style systems, deep RoBERTa-style systems still perform better than wide systems, even when increasing the number of attentions heads. For GPT-2-style systems, adding more attention heads hardly increases the performance.

5.3 Monitoring the Validation Loss

In the previous sections, different models were made comparable by their number of non-embedding parameters. As stated in section 4.3, this number is related to the computational cost when evaluated as the number of FLOPs per token. Reporting the computational cost in FLOPs neglects, however, that some operations can be run in parallel, while others cannot. In order to assess the speed of convergence, following Li et al. (2020),

we therefore directly report the wall-clock time in seconds.

Figure 3 shows the validation loss for BERT-style systems of different shape, when pre-trained on the short sequences.[9] The left plot depicts several pre-training loss curves corresponding to the single-dimension scaling experiments from Section 5.1. Interestingly, when comparing the validation loss with the GLUE-Large results (cf. Fig. 1), we find that, although increasing H (while holding A fixed) results in a lower validation loss than increasing L, the GLUE-Large score shows a higher increase in the latter case.

Observation 5 *The pre-training validation loss is not necessarily a good indicator for the performance of a fine-tuned system.*

Dependent on the downstream task some architectures presumably favor fine-tuning more than others, which can offset a relatively worse initialization point. This finding suggests that, although Kaplan et al. (2020) observe similar test losses for different shapes, benchmarking the corresponding fine-tuned versions may present a different picture.

In the left plot of Figure 3 we furthermore observe that shape has a significant effect on the pre-training time. In particular, stacking many layers requires much longer pre-training. It is also evident that increasing the size does not lead to a proportionate increase in the pre-training time. This holds true especially when scaling multiple dimensions, as depicted in the right plot of Figure 3. When doubling the number of pre-training parameters, the

---

[9]We do pre-training on short and long sequences. For a detailed description, see Appendix A and Appendix F.
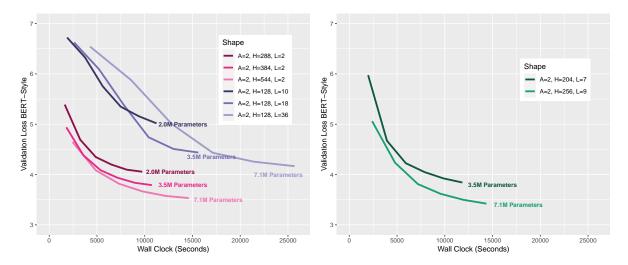
Figure 3: Loss curves of BERT-Style systems of different shape. All loss curves are associated with the first stage of pre-training, where we train on short sequences with a of 128 tokens (For the loss curves for the subsequent training on the long sequences, see Appendix D). The depicted parameter counts refer to the model size N_{model}.

training time only increases from approximately $11,800$ seconds to approximately $14,400$ seconds. In particular, the loss of the larger system is smaller at any measured point in time.

Observation 6 *Given a fixed time budget, training large systems for a relatively small number of steps is more efficient than training small systems for a large number of steps.*

The 9-layer system in the right plot of Figure 3 achieves a notably lower validation loss than the 7-layer system after $10,000$ seconds, which corresponds to approximately $65,800$ and $79,800$ steps, respectively. Li et al. (2020) made a similar observation by showing that larger Transformer-based systems generally reach a lower pre-training validation perplexity in shorter time. A point of concern might be that larger systems overfit more easily during fine-tuning. However, Li et al. (2020) showed that, when stopping models of different size at the same pre-training validation perplexity, large systems generally achieve comparable downstream task performances to small systems, which contradicts the overfitting argument.

5.4 Number of Training Steps and Batch Size

The amount of processed data can be increased by increasing either the number of trainig steps or the batch size. In Table 2 we compare how halving the number of steps vs. halving the batch size impacts model performance. As baseline we use our best performing system thus far ($A = 2, H = 256, L = 9$), pre-trained RoBERTa- and BERT-style.

In both cases we find that reducing the number of training steps is more detrimental to the performance than reducing the batch size. Conversely, it follows that when scaling up a system, a better model performance can be achieved when doubling the amount of training steps than when doubling the batch size, which is consistent with the results of Raffel et al. (2019). On the other hand, we observe that the systems with the smaller batch size were trained for a significantly longer time than the systems with the reduced number of training steps. Therefore, increasing the batch size may result in a more favorable training duration than increasing the number of training steps. The modest drop in GLUE-Large performance, when halving the number of training steps is consistent with our findings from Section 5.3 and provides additional evidence that training for a large number of steps is inefficient.

Observation 7 *Doubling the number of training steps marginally increases the downstream task performance, whereas doubling the batch size significantly reduces the average training time of an input sequence.*

As stated, several other studies have shown that using a larger batch size is in general more efficient than training for more steps (Kaplan et al., 2020). This means that the reduction of training time by using larger batches dominates the marginal performance gains resulting from an increased number of training steps. However, for each specific model and training configuration there exists a critical batch size, after which the performance hardly im-

BERT-Style		**Validation Set Performance**				
Training Strategy	Total Time	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
Baseline	$21,358$s	78.6	72.0/72.7	81.2	82.5	83.4
$\frac{1}{2}$x steps, 1x batch	$10,736$s	77.4	70.2/71.2	80.5	81.5	82.5
1x steps, $\frac{1}{2}$x batch	$14,575$s	78.2	71.5/71.9	80.9	82.3	83.9
RoBERTa-Style		**Validation Set Performance**				
Training Strategy	Total Time	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
Baseline	$19,760$s	75.0	68.4/70.9	78.2	78.3	75.0
$\frac{1}{2}$x steps, 1x batch	$9,906$s	73.7	67.0/69.0	76.7	77.4	83.5
1x steps, $\frac{1}{2}$x batch	$13,101$s	75.6	68.2/70.0	79.5	78.9	84.4

Table 2: GLUE results and total pre-training time when halving batch size vs. number of training steps.

proves, if at all (Kaplan et al., 2020; Li et al., 2020). Our results suggest that this critical size is very small in our experiments, which we believe is due to the small size of the pre-training data set, as also observed by Kaplan et al. (2020).

5.5 Systematic Scaling

In this section we apply a modified version of the compound scaling method that was used to scale up EfficientNet (Tan and Le, 2019), a model that achieved a notably better accuracy on ImageNet (Deng et al., 2009) than previous approaches using less compute. For scaling, we only consider BERT-style systems and propose the following compound scaling method for Transformer-based systems:

$$L = \alpha^\phi, \qquad H = \beta^\phi, \qquad A \approx H/64,$$
$$\text{s.t. } \alpha\beta^2 \approx 2, \quad \text{with} \quad \alpha \geq 1, \beta \geq 1. \tag{2}$$

For suitable values of α and β, a system is scaled up by increasing the *compound coefficient* ϕ. Doubling L doubles N_{model}, while H leads to a fourfold increase. Since N_{model} dominates the amount of compute in a Transformer, the constraint $\alpha\beta^2 \approx 2$ thus ensures that when scaling the network from ϕ_{old} to ϕ_{new}, the amount of compute (which is approx. independent of A) approximately increases by the factor $2^{\phi_{\text{new}} - \phi_{\text{old}}}$. Following existing approaches and using Observation 4, we therefore set the number of attention heads to $A \approx H/64$.

Grid search To determine α and β, we follow Tan and Le (2019) and perform a grid search over a set of nine small networks of comparable size trained only on the short sequences. Subsequently, we select the three systems with the lowest validation loss. Based on Observation 5, we then fine-tune and evaluate these three systems on GLUE-Large, which leads to the best performing system

having $L = 3$ and $H = 104$ (cf. Tab. 7 in Appendix C). From the constraint in Eq. (2) it follows that the size of this system corresponds to a compound coefficient of $\phi = \log_2(LH^2) = 14.99 \approx 15$, such that we obtain $\alpha = 3^{\frac{1}{15}} \approx 1.076$, $\beta = 104^{\frac{1}{15}} \approx 1.363$. Note that the resulting coefficients favor scaling width over depth. In general, we believe that this is reasonable, especially in light of the much longer training times of deep networks compared to wide networks (cf. Fig. 3). However, we also want to emphasize that further research is needed, whether these scaling coefficients are suitable for BERT-style systems. For GPT-2-style systems, Kaplan et al. (2020) proposed to scale such that width/depth remains fixed. Importantly, however, Kaplan et al. (2020) did not study the effect of shape parameters on the GLUE-Large performance, but instead only monitored the LM test loss. In machine translation, on the other hand, Transformer-based systems are scaled preferably by increasing width (Shazeer et al., 2018; Li et al., 2020). Other approaches focus on increasing depth, while making modifications to the Transformer to allow for more efficient training (Al-Rfou et al., 2019).

Scaling Based on Observation 6, we successively increase the compound coefficient to scale three systems to larger sizes than all previously trained systems, but train for less steps. For our smallest system, we train for 5 epochs on both the long and the short sequences.[10] The results are listed in Table 4. Furthermore, Table 3 shows a comparison of the smallest of the three systems to the best performing system so far, as well as to a modification of this system which fulfills the requirement

---

[10]Since validation loss on the long sequences did not further decrease after 3 epochs, the two larger systems were only trained for 3 epochs on these sequences (cf. Appendix D).

BERT-Style							Validation Set Performance	
ϕ	A	H	L	N_{model}	Total Time	Epochs	GLUE-Large	Final Loss
NA	2	256	9	7,0778,88	21,358s	6	78.6	3.24
NA	4	256	9	7,0778,88	21,703s	6	78.9	3.29
19.865	7	469	4	10,558,128	**20,873s**	5	**79.4**	**3.13**

Table 3: Verification of the scaling method: The proposed modifications lead to a better GLUE score and a lower validation loss, while requiring less training time compared to previous best performing models.

BERT-Style					Validation Set Performance					
ϕ	A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA
20.578	9	585	5	20,553,500	80.7	75.3/75.5	83.5	83.4	85.1	16.5
21.716	13	832	5	41,553,440	**81.4**	**75.6/75.9**	**84.1**	**84.4**	**85.8**	**21.3**

Table 4: GLUE results of BERT-style systems, scaled up based on the observations made in the previous sections.

$A \approx H/64$. As can be observed, both the performance on the large GLUE-Large tasks and the final validation loss are improved, while requiring less training time. For the two larger systems, each obtained by approximately doubling the model size, downstream performance and validation loss are further improved (cf. Tab. 4). Note that these systems are rather large compared to the amount of pre-training data. This demonstrates the remarkable robustness of these systems with respect to overfitting on the pre-training data, which is in line with the results of Kaplan et al. (2020).

6 Conclusion & Future work

Limitations The most severe limitation is the small pre-training data set. Based on the observations of Kaplan et al. (2020), systems train faster if more training examples are used. The small size of the pre-training data set might also be the cause of overfitting on smaller tasks. Therefore, for further experiments, we suggest to expand the amount of pre-training data. Furthermore, we did no hyperparameter tuning, but instead adopted the configurations from the original models. It would be advisable to adjust the hyperparameters accordingly (Li et al., 2020), especially since we used different batch sizes as the original models.

Directions for Further Research Kaplan et al. (2020) studied the effect of the amount of pre-training data, however, not with regard to downstream task performance. Due to the fact that current NLP systems are trained on vastly different amounts of pre-training data, we believe that this relationship should be explored further.

Although attempts have been made to study the relationship between different pre-training objectives and the performance on downstream tasks (Arora et al., 2019), this relation is yet not well understood. Empirically, contrastive pre-training objectives, such as replaced token detection (Clark et al., 2020) have shown very promising results. It would be interesting to extend the study to such contrastive objectives. Since we observed that the NSP task is beneficial for learning sentence-pair relationships, comparing it to ALBERT's SOP task (Lan et al., 2019) could yield further insights.

Finally, by fine-tuning on a larger variety of tasks we could break down in more detail how different modeling choices affect the performances on different tasks. We believe that further investigation of such relationships will open many opportunities for future research.

Conclusion In our experiments, BERT-style systems consistently outperform RoBERTa-style and GPT-2-style systems. We therefore conclude that, at least in case of a relatively small pre-training data set, the combination of MLM & NSP is preferable to MLM or LM. Although our experiments were conducted on a much smaller scale than other studies, we were able to reproduce many previous findings. For instance, we observed that, provided multiple dimensions are scaled, systems with very different shapes can achieve similar performances.

Consistent with previous studies (Kaplan et al., 2020; Li et al., 2020) we found that it is in general inefficient to train until convergence and that training for more steps improves the performance rather marginally. Instead, in accordance with Kaplan et al. (2020), we believe that increasing the batch size is more beneficial than training for more steps.

More importantly, also consistent with the results of Kaplan et al. (2020) and Li et al. (2020), we conclude that the model size is the key factor in Transformer-based systems. We observed that even for rather large systems, both the final pre-training validation loss and the GLUE performance benefit from further increasing the size. At the same time, the total pre-training time increases at a rather low rate. In particular, given a fixed time budget, large systems reach a lower loss than small systems. Therefore, we believe that additional compute should be allocated mainly to increase the model size.

Acknowledgements We would like to thank the three anonymous reviewers for their insightful comments and their feedback on our work.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. 2019. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*.

Matthias Aßenmacher and Christian Heumann. 2020. On the comparability of pre-trained language models. In *Proceedings of the 5th Swiss Text Analytics Conference and 16th Conference on Natural Language Processing*, Zurich, Switzerland (Online). CEUR Workshop Proceedings.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

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.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *OpenReview.net*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *arXiv preprint arXiv:2002.11794*.

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*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

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

Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. 2019. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*.

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

Iyer Shankar, Dandekar Nikhil, and Csernai Kornél. 2017. First quora dataset release: Question pairs. Accessed: 2021-02-01.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. 2018. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, pages 10414–10423.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Mingxing Tan and Quoc V Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

Sebastian Thrun and Lorien Pratt. 1998. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

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.

Appendix

A Pre-training details

Training duration To ensure a fair comparison of the different pre-training objectives, we pre-train RoBERTa-style and GPT-2-style systems for 10 epochs, and BERT-style systems for 6 epochs, which in all cases equates to approximately $137,000$ total training steps combined over both partitions.[11] Since the data is duplicated when training with MLM & NSP, it is natural to simply lower the number of epochs in relation to the amount of pre-training data. While the amount of pre-training data of RoBERTa-style and GPT-2-style systems amounts to more than 60% of the data of BERT-style systems, we found that, on the other hand, the average WordPiece token contains slightly more information than the average byte-level BPE token.

Optimization Apart from the experiments in section 5.4, we use a batch size of 64 when training on the short sequences and a batch size of 16 for the long sequences. We optimize all systems with Adam (Kingma and Ba, 2014) using the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = $ 1e-6 and L_2 weight decay of 0.01. For BERT-style and RoBERTa-style systems we use a maximum learning rate of 1e-4, and for GPT2-style systems the maximum learning rate is 2.5e-4. In all cases we use a linear warmup for the first 1000 steps, which corresponds to approximately 1% of the total steps. Furthermore, for all systems we employ dropout with a rate of 0.1 on all layers. The activation function of all systems is the GELU (Hendrycks and Gimpel, 2016). The hyperparameters are in general chosen as in the original systems, except for RoBERTa-style systems, because RoBERTa was trained with significantly larger batches, which requires different hyperparameters. For RoBERTa-style systems we therefore choose the same hyperparameters as for BERT-style systems.

Implementation We pre-train all systems on a single NVIDIA 16GB V100 GPU, making use of the Hugging Face transformers library (Wolf et al., 2020). The same also holds true for fine-tuning.

Short and long sequences With our pre-training procedure we follow Devlin et al. (2019): The

---

[11]In sections where we do not compare the different objectives the number of epochs may differ.

first 90% of the steps on short sequences (128 tokens), the remaining 10% on long ones (512 tokens). When inspecting the validation loss, we adjust the evaluation sequence lengths to the lengths of the training sequences, so ensure the same distribution for training and validation data. This causes the validation loss on the long sequences to start at a slightly higher point than the final validation loss on the short sequences (cf. Appendix D).

B Fine-tuning details

We follow Devlin et al. (2019) and train for three epochs on all GLUE tasks. We use a batch size of 16 and a learning rate of 2e-5 for each task. Apart from these hyperparameter configurations, we apply the same fine-tuning procedures that were used by the original systems. For GPT-2-style systems, we implemented the fine-tuning approach of GPT (because GPT-2 was not fine-tuned).

However, we do make one small modification to the original implementations. In contrast to BERT-style systems, the pre-training objective of RoBERTa-style and GPT-2-style systems does not contain a classification task. When performing the NSP task, in the original BERT the contextualized representation of the CLS token is obtained by feeding the corresponding final hidden state through a linear layer with dropout and $tanh$ activation. Subsequently, the contextualized representation is fed through another linear layer with dropout, which is the output layer mapping the contextualized representation to the class probabilities. Consequently, when fine-tuning BERT-style systems on a classification task, there are in fact two linear layers between the final hidden state and the output classes. However, RoBERTa and GPT in their original implementation use only one linear layer. In order to be as consistent as possible, in contrast, we use two linear output layers for all systems. The first linear layer is followed by a $tanh$ activation and both layers are implemented with a dropout rate of 0.1. For more information regarding this issue see huggingface's discussion forum.

C Detailed performance values for single shape dimensions and results for the grid search

Performance values on GLUE-Large and SST-2 for scaling H (Tab. 5) and for scaling L (Tab. 6). Table 7 shows the results of the grid search.

	BERT-Style			Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	65.4	59.0/60.2	72.3	64.8	78.0
2	192	2	884,736	67.2	62.1/62.8	74.0	65.4	82.6
2	288	2	1,990,656	69.3	63.7/65.2	76.0	68.3	82.0
2	384	2	3,538,944	72.3	65.7/66.6	77.8	73.2	81.1
2	544	2	7,102,464	72.3	66.8/68.1	78.0	72.0	83.3
	GPT-2-Style			Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	61.6	56.3/56.2	66.1	62.3	79.8
2	192	2	884,736	62.9	58.0/58.4	68.7	61.9	79.7
2	288	2	1,990,656	63.9	58.7/58.7	70.9	62.2	81.7
2	384	2	3,538,944	64.9	59.8/59.6	71.9	63.0	81.2
2	544	2	7,102,464	65.0	59.8/59.7	72.4	62.9	82.5
	RoBERTa-Style			Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	60.1	53.7/55.1	64.7	61.9	79.2
2	192	2	884,736	60.5	54.4/55.4	65.0	62.0	80.8
2	288	2	1,990,656	63.0	57.5/58.0	68.1	63.4	80.3
2	384	2	3,538,944	64.3	59.4/59.8	69.0	64.6	81.9
2	544	2	7,102,464	66.5	60.2/60.7	72.7	66.5	81.8

Table 5: Performance on GLUE when increasing only the embedding dimension.

	BERT-Style			Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	65.4	59.0/60.2	72.3	64.8	78.0
2	128	5	983,040	68.9	62.1/64.2	75.0	68.6	79.8
2	128	10	1,966,080	72.0	65.3/66.9	76.7	74.1	81.8
2	128	18	3,538,944	74.2	67.2/68.6	77.8	77.7	82.2
2	128	36	7,077,888	75.9	69.7/70.4	79.7	78.3	83.3
	GPT-2-Style			Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	61.6	56.3/56.2	66.1	62.3	79.8
2	128	5	983,040	62.4	57.6/56.1	67.4	62.0	80.5
2	128	10	1,966,080	62.0	56.9/57.0	67.7	61.5	81.4
2	128	18	3,538,944	61.8	56.1/56.4	66.8	62.4	80.6
2	128	36	7,077,888	61.4	56.6/56.7	66.6	61.1	80.7
	RoBERTa-Style			Validation Set Performance				
A	H	L	N_{model}	GLUE-Large	MNLI-(m/mm)	QQP	QNLI	SST-2
2	128	2	393,216	60.1	53.7/55.1	64.7	61.9	79.2
2	128	5	983,040	64.8	59.5/60.6	70.4	64.4	80.2
2	128	10	1,966,080	67.1	60.9/61.9	72.0	68.5	81.7
2	128	18	3,538,944	67.2	62.9/64.3	74.3	64.3	80.0
2	128	36	7,077,888	73.3	67.6/69.1	77.3	75.0	82.6

Table 6: Performance on GLUE when increasing only the number of layers.

	BERT-Style			Validation Loss (WikiText-103)	Validation Performance (GLUE)
A	H	L	N_{model}	BERT-Style Loss	GLUE-Large
2	128	2	393,216	**5.66**	66.6
2	104	3	389,376	**6.34**	**68.2**
2	90	4	388,800	**6.41**	67.1
2	74	6	394,272	6.47	-
2	64	8	393,216	6.50	-
2	58	10	403,680	6.54	-
2	52	12	389,376	6.58	-
2	48	14	387,072	6.62	-
2	46	16	406,272	6.62	-

Table 7: Grid search over nine small BERT-style systems.

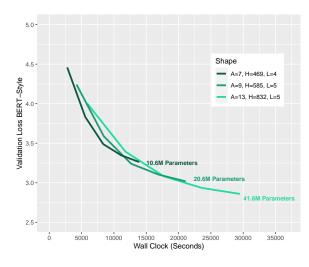## D Validation loss for scaled-up models

Figure 4: Validation loss of scaled-up BERT-style systems when pre-training on the short sequences. The depicted parameter counts refer to N_{model}.

E Definition of the model size

We follow Kaplan et al. (2020) and use the approximate number of non-embedding parameters to define the model size, which we denote as N_{model}. The embedding parameters consist of all token, position and (if present) segment embeddings. The number of embedding parameters does not depend on the network depth, and when scaling width and/or depth, it is a sub-leading term of the total number of parameters. Furthermore, the number of FLOPs related to embedding (and de-embedding) is also sub-leading term of the total number of FLOPs. Consistent with this is the observation of Kaplan et al. (2020) that discarding the number of embedding parameters when calculating model size and amount of compute results in significantly cleaner scaling laws. Since the share of embedding parameters decreases significantly for larger models, similarly to Kaplan et al. (2020) we expect that discarding the number of embedding parameters allows for a better generalization of our results to large models. Another advantage of defining the model size as the number of non-embedding parameters is that this number is closely linked to the number of (non-embedding related) FLOPs. This enables us to design benchmarking scenarios by training different models of comparable size, which at the same time require roughly similar amounts of computation.
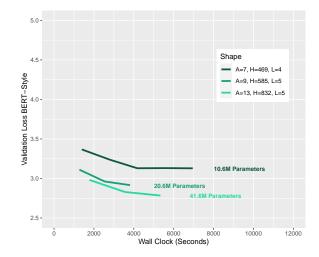

Figure 5: Validation loss of scaled-up BERT-style systems when pre-training on the long sequences. The depicted parameter counts refer to N_{model}.

Number of Non-Embedding Parameters

Omitting biases and other sub-leading terms, the number of non-embedding parameters, which is our definition of the model size, is given by

$$N_{\text{model}} := 12LH^2, \tag{3}$$

where we have assumed that $H_k = H_v = \frac{H}{A}$ and $H_{ff} = 4H$. Therefore, per layer there are approximately $12H^2$ non-embedding parameters. This number can be derived from the following three steps performed in each layer of a Transformer:

1. Input projection For each attention head, the queries, keys and values of dimension $\frac{H}{A}$ are obtained with the three matrices $\boldsymbol{W}_i^Q$, $\boldsymbol{W}_i^K$, and $\boldsymbol{W}_i^V$, which are each of size $H \times \frac{H}{A}$. In total, the input projection thus consists of $3 \cdot A \cdot \frac{H^2}{A} = 3H^2$ parameters.

2. Output projection First, note that performing attention on the projected inputs of dimension $\frac{H}{A}$ involves no additional parameters. The concatenated attention results are projected back to dimension H with the $H \times H$ matrix $\boldsymbol{W}^O$. Therefore, the output projection involves an additional set of H^2 parameters.

3. Feed-forward network The last sub-layer of each layer consists of applying a feed-forward network to the output projections. There exist $H \cdot 4H$ connections between the output projections and the neurons of the inner-layer, and another $4H \cdot H$ connections from the inner-layer to the final output neurons. This step hence involves $8H^2$ parameters.

Note that the feed-forward network accounts for the majority of non-embedding parameters, followed by the input and output projections, respectively.

Relation to FLOPs

As stated, the number of non-embedding parameters is closely linked to the number of non-embedding related FLOPs. We start by deriving the number of FLOPs per token and forward pass for GPT-2-style systems, where sub-leading terms such as biases and layer normalization are again omitted.

1. Input projection The matrix-vector products of each per-layer input with $\boldsymbol{W}_i^Q$, $\boldsymbol{W}_i^K$, and $\boldsymbol{W}_i^V$ involve approximately $3 \cdot 2 \cdot H \cdot \frac{H}{A}$ FLOPs per attention head. Considering all attention heads, the input projection thus requires approximately $6H^2$ FLOPs per token.

2. Attention The computation of the attention operation can be divided into two sub-components:

- **Computation of the weights**: On average, $\frac{N_{ctx}}{2}$ attention weights have to be computed per input token, since on average half of the tokens are masked for each input token. Computation of a dot-product attention weight requires approximately $2\frac{H}{A}$ FLOPs per head. In total, the computation of the attention weights hence involves approximately $N_{ctx}H$ FLOPs per token.

- **Computation of the weighted sum**: Since only half of the tokens are summed on average, given the attention weights, calculation of the weighted sum of the values has an average cost of approximately $N_{ctx}H$ FLOPs for each token.

3. Output projection The vector matrix product of the attention outputs with $\boldsymbol{W}^O$ requires approximately $2H^2$ FLOPs for each token.

4. Feed-forward network The feed-forward network consists of two consecutive matrix multiplications, where each matrix contains $4H^2$ parameters. Thus, the feed-forward network requires approximately $2 \cdot 2 \cdot 4H^2 = 16H^2$ FLOPs per token.

The number of FLOPs per token and forward pass in GPT-2-style systems, which we denote by $C_{forward}$, can hence be approximated as

$$
\begin{aligned}
C_{forward} &\approx L(6H^2 + N_{ctx}H + N_{ctx}H \\
&\quad + 2H^2 + 16H^2) \\
&= 24LH^2 + 2LN_{ctx}H \\
&= 2N_{\text{model}} + 2LN_{ctx}H.
\end{aligned}
\tag{4}
$$

BERT-style and RoBERTa-style systems require slightly more FLOPs than GPT-2-style systems, because these systems have no autoregressive attention mask. Hence, in both steps of the attention operation above, the computational cost is approximately twice as much, i.e., $2N_{ctx}H$ in each step. Therefore, BERT-style and RoBERTa-style systems require approximately $2N_{\text{model}} + 4LN_{ctx}H$ FLOPs per token and forward pass. As mentioned by Kaplan et al. (2020), if $H > N_{ctx}/12$, the context-dependent term in Eq. (4) only accounts for a relatively small fraction of the compute of GPT-2-style systems. In particular, when increasing H, the importance of the context-dependent term diminishes. For BERT-style and RoBERTa-style systems the context-dependent term becomes small if $H > N_{ctx}/6$. Both constraints are satisfied by a large margin for all our systems, especially since we mainly train on rather short sequences. The backward pass requires approximately twice as much compute as the forward pass (Kaplan et al., 2020), such that the total amount of non-embedding related compute per token and training step can be approximated as

$$
C := 6N_{\text{model}}.
\tag{5}
$$

F Sequence characteristics

The following Table 8 provides an overview on the number of tokens in short and long sequences.

System	Partition	Number of Tokens	
		Total	Average
BERT-Style	Short	$110,888,186$	110.04
	Long	$43,274,856$	375.52
RoBERTa-Style	Short	$70,025,709$	110.31
	Long	$27,692,351$	457.04
GPT-2-Style	Short	$70,564,106$	111.16
	Long	$27,729,551$	457.65

Table 8: Number of tokens for the short and the long sequences as well as the average sequence lengths resulting from the different tokenizers.