Pre-training a BERT with Curriculum Learning by Increasing Block-Size of Input Text

Koichi Nagatsuka
Soka University
Tokyo, Japan
e20d5301@soka-u.jp

Clifford Broni-Bediako
Soka University
Tokyo, Japan
e18d5252@soka-u.jp

Masayasu Atsumi
Soka University
Tokyo, Japan
matsumi@soka.ac.jp

Abstract

Recently, pre-trained language representation models such as BERT and RoBERTa have achieved significant results in a wide range of natural language processing (NLP) tasks, however, it requires extremely high computational cost. Curriculum learning (CL) is one of the potential solutions to alleviate this problem. CL is a training strategy where training samples are given to models in a meaningful order instead of random sampling. In this work, we propose a new CL method which gradually increases the block-size of input text for training the self-attention mechanism of BERT and its variants using the maximum available batch-size. Experiments in low-resource settings show that our approach outperforms the baseline in terms of convergence speed and final performance on down-stream tasks.

1 Introduction

Recent years have seen a series of breakthroughs in pre-trained language representation models. The development of pre-training methods like BERT (Devlin et al., 2019) and its variants (Liu et al., 2019) have led to large improvements in many down-stream tasks such as paraphrase identification, sentence textual similarity, sentiment analysis, and natural language inference. One of the advantages in training these models is that they can leverage the unlabeled large-scale corpora which are more available compared to the labeled ones. However, training these models with large-scale corpora is pretty expensive in terms of computational time and memory footprint. In the literature, there are three main approaches that have been adopted to address this problem. These are architecture-based approach (Sanh et al., 2019; Voita et al., 2019; Sukhbaatar et al., 2019; de Wynter and Perry, 2020; Lan et al., 2020), task-based approach (Yang et al., 2019; Clark et al., 2020) and dataset-based approach (Elman, 1993; Bengio et al., 2009; Moore and Lewis, 2010; Gururangan et al., 2020). While the architecture-based and task-based methods have been extensively studied in the context of pre-training methods for natural language processing (NLP), dataset-based approach is relatively unexplored. To this end, we adopt a dataset-based method called Curriculum Learning (CL) which controls the order of training samples so that the model might converge faster with better performance.

The idea of CL-like approach was originally proposed by Elman (1993). The idea is based on the actual learning mechanism of humans and animals, where basic concepts are acquired first, then more complex ones are gradually learned. Bengio et al. (2009) formalized this concept as CL to train neural networks. Through experimental analysis, Bengio et al. (2009) showed the benefit of CL on convergence speed and performance in shape recognition and language modeling tasks. One of the most significant challenges when adapting CL to a new task is to figure out a criterion for measuring the difficulty of the training samples. For example, in object recognition task, the size of objects is a good measure of difficulty (Shi and Ferrari, 2016; Ionescu et al., 2016), and the presence of low-frequent words in input text is an indicator of difficulty in language modeling (Bengio et al., 2009). These criteria vary greatly depending on the task, thus, it is not easy to define a measure of difficulty which is suitable for a particular task.

Most studies in the field of CL for NLP have proposed variety of difficulty measure by leveraging heuristics of the target tasks with neural networks (Bengio et al., 2009; Kocmi and Bojar, 2017; Soviany et al., 2021; Spitkovsky et al., 2009; Cirik et al., 2016; Rajeswar et al., 2017). On the other hand, it is not clear how to design CL for language representation models such as BERT. In pre-training BERT, distributed word representations are learned through optimizing masked language modeling (MLM) loss which is computed by predicting a masked word or token in an input text. The input to the model is not a single sentence but an arbitrary-length span of text called a *block*. This indicates that it is not obvious how to measure the difficulty of the training samples using CL-based approach proposed in the previous studies.

The key component of BERT is the multi-head self-attention mechanism that learns to compute token embeddings from its context (Devlin et al., 2019). The multi-head self-attention mechanism can be thought as a problem of searching for important token-pairs based on the relative magnitude of attention among all the token-pairs in an input text. This process can be served as a clue which leads us to speculate that it might be possible to formulate CL strategy by focusing on the effective training of the self-attention mechanism in BERT. Although each individual head of the multi-head self-attention mechanism can learn any dependency among tokens, most of the heads tend to pay more attention to local dependencies than global ones (Kovaleva et al., 2019; Brunner et al., 2019; Sukhbaatar et al., 2019; Jiang et al., 2020). It could be easier to train local self-attention in shorter blocks of input text than global self-attention in longer ones. Therefore, the block-size of input text can be used as the effective criterion to measure the difficulty-level of training samples for BERT.

In this paper, we introduce a new CL method which gradually increases the block-size of input text for pre-training BERT using the maximum available batch-size to accomplish convergence speedup, and also improve performance in the down-stream tasks. Since our approach is very simple, it is easy to apply it to BERT and its variants with little effort. Using a small-scale corpus, the experimental results demonstrated that our proposed approach outperforms the baseline on GLUE tasks with faster convergence speed.

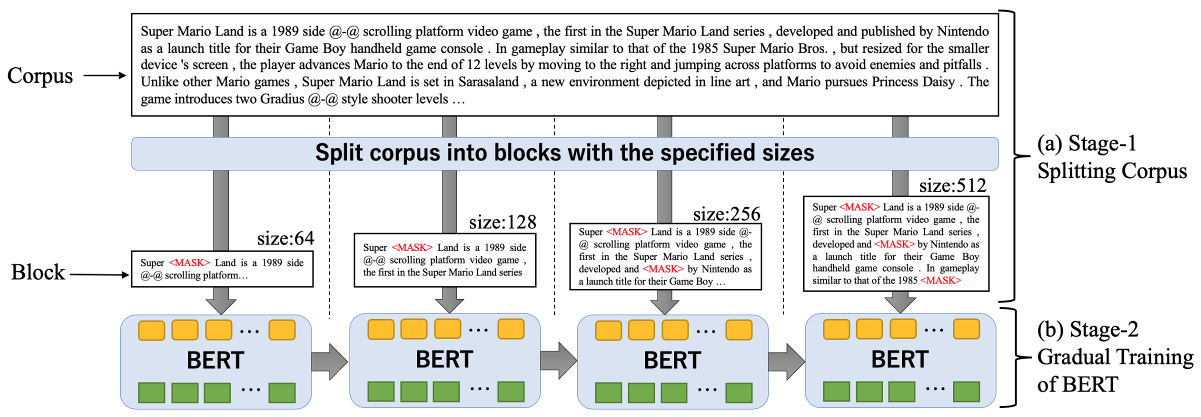
2 Related Work

To reduce the memory footprint and improve the training speed of pre-trained language models, prior works have shown that architecture-based approaches are very useful. Sanh et al. (2019) proposed to leverage knowledge distillation to train a smaller version of BERT with faster training speed while maintaining comparative performance. Lan et al. (2020) used factorized embedding parametarization and cross-layer parameter sharing, which led to the reduction of parameter size and training time. de Wynter and Perry (2020) applied neural architecture search to select the optimal architecture of BERT and successfully compressed the size of the model. Task-based approaches have also been explored for pre-training language models with high training efficiency. Yang et al. (2019) introduced permutation language modeling which retains the benefits of autoregressive models and allows the models to capture bidirectional context. Instead of performing pre-training with MLM task, Clark et al. (2020) trained a BERT as a discriminator that determines whether each corrupted token was replaced by a generator model.

Recent studies have shown that CL is a successful approach for a wide range of machine learning applications (Soviany et al., 2021; Wang et al., 2021), including the fine-tuning of large-scale language models such as BERT (Xu et al., 2020). Some of large-scale language models like GPT-3 (Brown et al., 2020) and T5 (Raffel et al., 2020) adopted non-uniform mixing strategies which control the amount of training samples from multiple corpora. However, CL strategy has not directly been applied to pre-training large-scale language models. There exists many studies of CL which used the length of sentences or input sequences as a measure of difficulty in NLP tasks including neural machine translation (Kocmi and Bojar, 2017), sentiment analysis (Cirik et al., 2016), parsing (Spitkovsky et al., 2009), poem generation (Rajeswar et al., 2017) and reading comprehension task (Tay et al., 2019). In this work, we exploit the block-size of input text in the context of self-attention mechanism as a measure of difficulty for pre-training BERT.

3 Method

The overview of the proposed CL method is presented in Figure 1. The method is divided into two stages: (a) Splitting a corpus based on specific block-sizes and (b) Gradual training of BERT by increasing the block-size. In the first stage, we split the original corpus into a series of input blocks with the pre-defined length. In the second stage, we train a model by changing the training samples from the short block-size to the long one depending on the pre-defined number of training steps. In training, some tokens in a block are randomly masked to perform the MLM task. We describe the MLM task and the details of the two stages of our



Corpus
Super Mario Land is a 1989 side @-@ scrolling platform video game , the first in the Super Mario Land series , developed and published by Nintendo as a launch title for their Game Boy handheld game console . In gameplay similar to that of the 1985 Super Mario Bros. , but resized for the smaller device 's screen , the player advances Mario to the end of 12 levels by moving to the right and jumping across platforms to avoid enemies and pitfalls . Unlike other Mario games , Super Mario Land is set in Sarasaland , a new environment depicted in line art , and Mario pursues Princess Daisy . The game introduces two Gradius @-@ style shooter levels …
Split corpus into blocks with the specified sizes
Block
size:64
size:128
size:256
size:512
BERT
(a) Stage-1 Splitting Corpus
(b) Stage-2 Gradual Training of BERT

Figure 1: The overview of the proposed CL method.

CL approach in this section.

3.1 Masked Language Modeling (MLM)

Let $\boldsymbol{x} = x_1, x_2, ..., x_T$ denotes a sequence of original tokens, where T is a block-size. By randomly masking an arbitrary number of tokens, we obtain an input sequence $\hat{\boldsymbol{x}}$. Given the corrupted sequence $\hat{\boldsymbol{x}}$, MLM is a task of predicting the original sequence $\boldsymbol{x}$. The training objective is formulated as:

$$\max_{\theta} \log p_\theta(\boldsymbol{x} \mid \hat{\boldsymbol{x}}) \approx \sum_{i=1}^{T} m_i \log p_\theta(x_i \mid \boldsymbol{x}_{<i}, \boldsymbol{x}_{>i}) \tag{1}$$

where x_i is the predicted token at position i and θ is the parameters of a model. m_i indicates the presence of a masked token where $m_i = 1$ if x_i is masked, otherwise 0. For this objective, we optimize the model parameters using the cross-entropy loss. In the MLM task, models infer masked tokens from bi-directional context ($\boldsymbol{x}_{<i}$ and $\boldsymbol{x}_{>i}$). The block-size restricts the available context information in both directions and thus affects the MLM accuracy.

3.2 Splitting a Corpus Based on Block-sizes

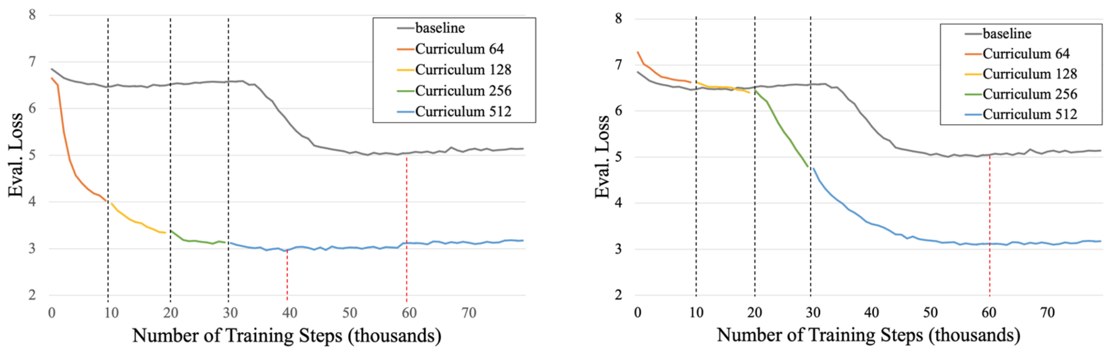
In the first stage, we split the original corpus into training samples with the specified size. Each input text for training BERT is not a linguistically coherent unit like a sentence or multiple sentences, but a fixed span of contiguous text (Devlin et al., 2019) that we called a *block*. In other words, the input is not guaranteed to end with a period nor start with a first word in a sentence. Liu et al. (2019) argues that it is desirable to acquire the input sequence to be at most 512 tokens through the extensive experiments. We follow this setting to obtain the block of a specified length from the corpus as a training sample. We train a byte-level Byte-Pair Encoding (BPE) tokenizer as in (Radford et al., 2019) to split the raw text into a sequence of tokens. By using byte-level BPE, we can decompose all words including out-of-vocabularies, which are likely to appear at test time especially when using a small training dataset. In the experiment, we set the vocabulary size to 20,000.

3.3 Gradual Training

In the second stage, we train a model step-by-step with four different block-sizes which are 64, 128, 256, and 512. We first train the model with the shortest block-size, which is 64 in this case, for an arbitrary number of steps. Then, we retrain the model in the order of 128 and 256 block-sizes respectively for the same number of steps. Finally, we retrain the model with the longest block-size of 512 until it converges. For masking tokens, we use the fixed masking rate of 0.15. When restarting the training, we always initialize the learning rate. To accelerate training, we use the maximum available batch-size depending on the block-size. Since our proposed method is designed to limit the block-size in the early training phase, we employ larger batch-size with shorter block-size which improves the whole training efficiency.

4 Experiments

In the experiments, we evaluate our proposed CL approach in terms of the convergence speed and model performance. We use wikitext-2 (Merity



(a) Increase of the block-size with the maximum available batch-size.

(b) Increase of the block-size with the fixed batch-size (16).

Figure 2: Comparison of our approach and the baseline on the validation losses. Left (a): The result of CL which increases the block-size with the maximum available batch-size. Right (b): The result of CL which increases block-sizes with the fixed batch size (16). Black dotted lines indicates the points where the block-size of training samples is changed, and the red dotted line indicates each convergence point.

Model (block-size)	Training time	Number of steps	Memory (batch-size)
Baseline(512)	5:28:38	60K	17.5(16)
Curriculum(64)	1:19:15	10K(fixed)	19.2(256)
Curriculum(128)	1:21:02	10K(fixed)	21.1(128)
Curriculum(256)	1:07:16	10K(fixed)	19.9(48)
Curriculum(512)	0:50:10	10K	17.5(16)
Curriculum(total)	**4:37:43**	**40K**	—

Table 1: Statistics on training of the baseline and each curriculum training phase.

et al., 2016) for pre-training RoBERTa (Liu et al., 2019), which is a variant of BERT. For fine-tuning on down-stream tasks, we use the General Language Understanding Evaluation (GLUE) dataset (Wang et al., 2018). All the training and fine-tuning were carried out on a GeForce RTX3090 with 24GB memory.

4.1 Datasets

Wikitext-2: Although BERT and its variants (e.g. RoBERTa) are commonly trained with large-scale corpora which contain over 3 billion words, we use wikitext-2 (Merity et al., 2016) which is a small corpus to enable pre-training with a limited computational resource. Wikitext-2 is one of the standard corpora for language models, and consists of 720 good-quality articles from English Wikipedia. It has about 2M tokens for training, and 217K and 245K tokens for validation and testing respectively.

GLUE Benchmarks: We fine-tune our models on the GLUE benchmarks (Wang et al., 2018). GLUE consists of nine datasets for measuring the generalization performance of pre-trained language models. We use only 7 datasets (SST-2, MRPC, QQP, MNLI-m, QNLI, RTE, and WNLI) out of the 9 GLUE benchmarks. CoLA and STS-B are removed due to a tendency to fall into over-fitting which stems from the small-scale pre-training.

4.2 Training Details

We perform both *curriculum training* and *anti-curriculum training* in the pre-training of RoBERTa. In curriculum training, we increase the block-size of training samples from the shortest to the longest. On the other hand, in anti-curriculum training, training samples with the longest block-size are first given to the model as the most difficult ones, then the difficulty-level of training samples is gradually reduced by shortening the block-size in the training process. By comparing curriculum training with anti-curriculum training, which follows the opposite sampling order, we show that increasing block-size is an effective CL method for pre-trained language representation models.

For all the models, we use the same RoBERTa-base architecture which has 12 layers with a hidden size of 768. Each layer has 12 attention heads. We

Model (block-size)	Samples per second	Batch-size	Validation loss
Baseline	48.68	16	5.045
Curriculum(64)	538.38	256	6.624
Curriculum(128)	263.26	128	4.030
Curriculum(256)	118.92	48	3.132
Curriculum(512)	53.15	16	2.950
Curriculum Ave.	**262.85**	—	**2.950**

Table 2: Comparison of training efficiency between the baseline and our curriculum model.

use AdamW (Loshchilov and Hutter, 2017) with a learning rate of 1e-5 in the pre-training with four different batch-sizes depending on the block-sizes as shown in Table 2. In fine-tuning, we also use the same optimizer as used in pre-training and set a learning rate to 5e-5 and batch-size to 64 for all task except for QNLI where we use learning rate of 2e-5 and batch-size of 16 due to the memory limitation.

We define the training time of the overall curriculum training as a total of the training time for every training phase corresponding to each block-size. In both curriculum training and anti-curriculum training, our models are trained for 10,000 steps with each block-size except for the last block-size where we continue the training until the convergence of the models. For comparative evaluation, we train RoBERTa without CL by using random sampling as the baseline model.

4.3 Results

4.3.1 Convergence Speed

Figure 2(a) shows the comparison of our curriculum model which increases the block-size with the maximum available batch-size and the baseline on the validation losses throughout pre-training. Compared to the loss of the baseline model that converged at around 5.0, the loss of curriculum model decreased steadily and achieved the faster convergence speed outperforming the baseline by about 2 points in validation loss. The learning curve of the baseline model were plateau until 35K steps, and then the loss finally restarted to descend. On the other hand, the loss of the curriculum model stably decreased every time we switched the difficulty-level of training samples. To analyze the effect of increasing a batch-size on convergence speed, we demonstrated an ablation study by fixing the batch-size to 16 (which is the maximum size when block-size is set to 512). Figure 2(b) shows the result of the curriculum model which increases block-size with the fixed batch-size. Compared to the our proposed curriculum model (Figure 2(a)), it required about 60K steps to converge, which is the same training time as the baseline. This result indicates that CL improves final performance but does not contribute to the convergence speedup in case the batch-size is fixed.

Table 1 presents the statistical information about the training of the baseline and each curriculum phase. While the baseline model converged after about 60K steps, our curriculum model required just 40K steps in total, which is about 1.5 times faster than the baseline. Although using the large batch-size depending on the small block-size tended to take long training time, it allows training a large number of training samples and the total training time was reduced by about 1.0 hours. Table 2 represents the comparison of training efficiency between the baseline and our curriculum model. With respect to the training samples per second, curriculum model achieved better training efficiency, which is 5 times as higher as the baseline, and also resulted in much better validation loss.

4.3.2 GLUE Results

Table 3 shows the GLUE scores on development datasets. For all 6 down-stream tasks, our curriculum model at the bottom of the table outperformed the baseline model at the top. Especially, performances on STS-2, MRPC, QQP, MNLI-m and QNLI were higher than the baseline by a large margin (+4.47 on SST-2, +3.19 on MRPC, +6.48 F1 score and +3.37 accuracy on QQP, 8.89 on MNLI-m, and 15.74 on QNLI) while accuracy on RTE and WNLI were extremely low in both curriculum and baseline. Although each scores of our model is not high due to the small-scale pre-training, relative improvements of scores by CL were generally observed.

Model (The Order of block-size)	SST-2	MRPC	QQP (F1/Acc.)	MNLI-m	QNLI	RTE	WNLI
Baseline (512)	79.01	69.60	69.77 79.52	57.39	63.97	51.98	56.33
Anti-Curriculum (512, 256, 128, 64)	80.38	70.09	72.88 81.47	60.64	49.46	52.34	56.33
Curriculum w/o 64 (128, 256, 512)	81.99	69.60	74.81 82.04	64.97	78.74	47.29	46.47
Curriculum w/o 128 (64, 256, 512)	83.37	70.58	75.21 82.29	66.34	77.74	45.12	56.33
Curriculum w/o 256 (64, 128, 512)	82.45	70.34	75.22 82.40	65.76	77.75	50.18	46.47
Curriculum w/o 512 (64, 128, 256)	80.61	70.83	75.76 **82.93**	66.53	75.76	51.26	32.39
Curriculum 2-stage (64, 512)	80.84	72.05	76.21 82.85	**66.82**	77.22	48.73	56.33
Curriculum (Ours) (64, 128, 256, 512)	**83.48**	**72.79**	**76.25** 82.89	66.28	**79.71**	**53.42**	56.33

Table 3: GLUE scores on development datasets. batch-size=64, lr=5e-5, but in QNLI, batch-size=16, lr=2e-5.

4.3.3 Comparison with Anti-Curriculum

Compared with our curriculum model, performances of anti-curriculum model were lower on every down-stream tasks. This result indicates that not decreasing but increasing a block-size is the effective for improving the generalization performances. Interestingly, the performances of anti-curriculum were better or equal to the baseline in all tasks except for QNLI. One possible reason for this result is that generating training samples with various block-sizes may have the same impact as data augmentation. Anti-curriculum model, however, failed to learn the QNLI task because the model is optimized for short text like 64 tokens at the end while the input of QNLI contains samples whose input length is longer than 64.

4.3.4 Ablation Study

As an ablation study, we tested two types of models including 3-stage curriculum and 2-stage curriculum. For the 3-stage curriculum, we removed a specific block-size from our training schedule and conducted the CL with the rest of block-sizes. For 2-stage curriculum, we trained the model only with the shortest block-size (64 tokens) and longest one (512 tokens).

As Table 3 shows, our curriculum model with the full training schedule is equal to or slightly better than the 2-stage or 3-stage models on each down-stream tasks. However, for tasks where performance gaps are not significant, the 2-stage and 3-stage curricula are more advantageous because of the shorter training time. As in the case of the anti-curriculum, the curriculum model without block-size of 512 tokens, that was not optimized for the largest block-size, had lower performance in QNLI. The 2-stage curriculum, which requires the least amount of training time, achieved almost the same accuracy as the normal curriculum in MRPC and MNLI-m, but relatively poor performance in tasks such as SST-2. These experiments show that there is room to further speed-up of CL by modifying the curriculum schedule on the block-size. Moreover, the result also indicates that the impact of CL on the performance will be different depending on each down-stream task.

5 Conclusion

In this paper, we proposed a new CL method for pre-training BERT, which progressively increase a block-size of input text. Our approach is very simple and thus handy to implement. Experiments in the low-resource setting have shown that proposed method leads to faster convergence speed and better performances in down-stream tasks. In further research, we expand the corpus and validate the scalability of our approach. In addition, we speculate that it is important to investigate when the difficulty-level should be changed through the training and how it affect model performances.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2019. On identifiability in transformers. *arXiv preprint arXiv:1908.04211*.

Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. 2016. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204*.

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*.

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.

Jeffrey L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Radu Tudor Ionescu, Bogdan Alexe, Marius Leordeanu, Marius Popescu, Dim P. Papadopoulos, and Vittorio Ferrari. 2016. How hard can it be? estimating the difficulty of visual search in an image. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2157–2166.

Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. Convbert: Improving bert with span-based dynamic convolution. *arXiv preprint arXiv:2008.02496*.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 379–386.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv:1909.11942 [cs]*.

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*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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

Robert C Moore and Will Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Miaojing Shi and Vittorio Ferrari. 2016. Weakly supervised object localization using size estimates. In *European Conference on Computer Vision*, pages 105–121.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2021. Curriculum learning: A survey. *CoRR*, abs/2101.10382.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2009. Baby steps: How “less is more” in unsupervised dependency parsing. In *NIPS 2009 Workshop on Grammar Induction, Representation of Language and Language Learning*.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335.

Yi Tay, Shuohang Wang, Anh Tuan Luu, Jie Fu, Minh C. Phan, Xingdi Yuan, J. Rao, S. C. Hui, and A. Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *ArXiv*, abs/1905.10847.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP.

Adrian de Wynter and Daniel J. Perry. 2020. Optimal subarchitecture extraction for bert. *CoRR*, abs/2010.10499.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32.