RoBERTa: A Robustly Optimized BERT Pretraining Approach

Yinhan Liu[*§] **Myle Ott**[*§] **Naman Goyal**[*§] **Jingfei Du**[*§] **Mandar Joshi**[†]
Danqi Chen[§] **Omer Levy**[§] **Mike Lewis**[§] **Luke Zettlemoyer**[†§] **Veselin Stoyanov**[§]

[†] Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA
{mandar90,lsz}@cs.washington.edu

[§] Facebook AI
{yinhanliu,myleott,naman,jingfeidu,
danqi,omerlevy,mikelewis,lsz,ves}@fb.com

arXiv:1907.11692v1 [cs.CL] 26 Jul 2019

Abstract

Language model pretraining has led to significant performance gains but careful comparison between different approaches is challenging. Training is computationally expensive, often done on private datasets of different sizes, and, as we will show, hyperparameter choices have significant impact on the final results. We present a replication study of BERT pretraining (Devlin et al., 2019) that carefully measures the impact of many key hyperparameters and training data size. We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it. Our best model achieves state-of-the-art results on GLUE, RACE and SQuAD. These results highlight the importance of previously overlooked design choices, and raise questions about the source of recently reported improvements. We release our models and code.[1]

1 Introduction

Self-training methods such as ELMo (Peters et al., 2018), GPT (Radford et al., 2018), BERT (Devlin et al., 2019), XLM (Lample and Conneau, 2019), and XLNet (Yang et al., 2019) have brought significant performance gains, but it can be challenging to determine which aspects of the methods contribute the most. Training is computationally expensive, limiting the amount of tuning that can be done, and is often done with private training data of varying sizes, limiting our ability to measure the effects of the modeling advances.

We present a replication study of BERT pretraining (Devlin et al., 2019), which includes a careful evaluation of the effects of hyperparmeter tuning and training set size. We find that BERT was significantly undertrained and propose an improved recipe for training BERT models, which we call RoBERTa, that can match or exceed the performance of all of the post-BERT methods. Our modifications are simple, they include: (1) training the model longer, with bigger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences; and (4) dynamically changing the masking pattern applied to the training data. We also collect a large new dataset (CC-NEWS) of comparable size to other privately used datasets, to better control for training set size effects.

When controlling for training data, our improved training procedure improves upon the published BERT results on both GLUE and SQuAD. When trained for longer over additional data, our model achieves a score of 88.5 on the public GLUE leaderboard, matching the 88.4 reported by Yang et al. (2019). Our model establishes a new state-of-the-art on 4/9 of the GLUE tasks: MNLI, QNLI, RTE and STS-B. We also match state-of-the-art results on SQuAD and RACE. Overall, we re-establish that BERT's masked language model training objective is competitive with other recently proposed training objectives such as perturbed autoregressive language modeling (Yang et al., 2019).[2]

In summary, the contributions of this paper are: (1) We present a set of important BERT design choices and training strategies and introduce

[*] Equal contribution.

[1] Our models and code are available at: https://github.com/pytorch/fairseq

[2] It is possible that these other methods could also improve with more tuning. We leave this exploration to future work.

alternatives that lead to better downstream task performance; (2) We use a novel dataset, CC-NEWS, and confirm that using more data for pretraining further improves performance on downstream tasks; (3) Our training improvements show that masked language model pretraining, under the right design choices, is competitive with all other recently published methods. We release our model, pretraining and fine-tuning code implemented in PyTorch (Paszke et al., 2017).

2 Background

In this section, we give a brief overview of the BERT (Devlin et al., 2019) pretraining approach and some of the training choices that we will examine experimentally in the following section.

2.1 Setup

BERT takes as input a concatenation of two segments (sequences of tokens), $x_1, \ldots, x_N$ and $y_1, \ldots, y_M$. Segments usually consist of more than one natural sentence. The two segments are presented as a single input sequence to BERT with special tokens delimiting them: $[CLS], x_1, \ldots, x_N, [SEP], y_1, \ldots, y_M, [EOS]$. M and N are constrained such that $M + N < T$, where T is a parameter that controls the maximum sequence length during training.

The model is first pretrained on a large unlabeled text corpus and subsequently finetuned using end-task labeled data.

2.2 Architecture

BERT uses the now ubiquitous transformer architecture (Vaswani et al., 2017), which we will not review in detail. We use a transformer architecture with L layers. Each block uses A self-attention heads and hidden dimension H.

2.3 Training Objectives

During pretraining, BERT uses two objectives: masked language modeling and next sentence prediction.

Masked Language Model (MLM) A random sample of the tokens in the input sequence is selected and replaced with the special token $[MASK]$. The MLM objective is a cross-entropy loss on predicting the masked tokens. BERT uniformly selects 15% of the input tokens for possible replacement. Of the selected tokens, 80% are replaced with $[MASK]$, 10% are left unchanged, and 10% are replaced by a randomly selected vocabulary token.

In the original implementation, random masking and replacement is performed once in the beginning and saved for the duration of training, although in practice, data is duplicated so the mask is not always the same for every training sentence (see Section 4.1).

Next Sentence Prediction (NSP) NSP is a binary classification loss for predicting whether two segments follow each other in the original text. Positive examples are created by taking consecutive sentences from the text corpus. Negative examples are created by pairing segments from different documents. Positive and negative examples are sampled with equal probability.

The NSP objective was designed to improve performance on downstream tasks, such as Natural Language Inference (Bowman et al., 2015), which require reasoning about the relationships between pairs of sentences.

2.4 Optimization

BERT is optimized with Adam (Kingma and Ba, 2015) using the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon =$ 1e-6 and L_2 weight decay of 0.01. The learning rate is warmed up over the first 10,000 steps to a peak value of 1e-4, and then linearly decayed. BERT trains with a dropout of 0.1 on all layers and attention weights, and a GELU activation function (Hendrycks and Gimpel, 2016). Models are pretrained for $S =$ 1,000,000 updates, with minibatches containing $B =$ 256 sequences of maximum length $T =$ 512 tokens.

2.5 Data

BERT is trained on a combination of BOOKCORPUS (Zhu et al., 2015) plus English WIKIPEDIA, which totals 16GB of uncompressed text.[3]

3 Experimental Setup

In this section, we describe the experimental setup for our replication study of BERT.

3.1 Implementation

We reimplement BERT in FAIRSEQ (Ott et al., 2019). We primarily follow the original BERT

[3] Yang et al. (2019) use the same dataset but report having only 13GB of text after data cleaning. This is most likely due to subtle differences in cleaning of the Wikipedia data.

optimization hyperparameters, given in Section 2, except for the peak learning rate and number of warmup steps, which are tuned separately for each setting. We additionally found training to be very sensitive to the Adam epsilon term, and in some cases we obtained better performance or improved stability after tuning it. Similarly, we found setting $\beta_2 = 0.98$ to improve stability when training with large batch sizes.

We pretrain with sequences of at most $T = 512$ tokens. Unlike Devlin et al. (2019), we do not randomly inject short sequences, and we do not train with a reduced sequence length for the first 90% of updates. We train only with full-length sequences.

We train with mixed precision floating point arithmetic on DGX-1 machines, each with 8 × 32GB Nvidia V100 GPUs interconnected by Infiniband (Micikevicius et al., 2018).

3.2 Data

BERT-style pretraining crucially relies on large quantities of text. Baevski et al. (2019) demonstrate that increasing data size can result in improved end-task performance. Several efforts have trained on datasets larger and more diverse than the original BERT (Radford et al., 2019; Yang et al., 2019; Zellers et al., 2019). Unfortunately, not all of the additional datasets can be publicly released. For our study, we focus on gathering as much data as possible for experimentation, allowing us to match the overall quality and quantity of data as appropriate for each comparison.

We consider five English-language corpora of varying sizes and domains, totaling over 160GB of uncompressed text. We use the following text corpora:

- BookCorpus (Zhu et al., 2015) plus English Wikipedia. This is the original data used to train BERT. (16GB).
- CC-News, which we collected from the English portion of the CommonCrawl News dataset (Nagel, 2016). The data contains 63 million English news articles crawled between September 2016 and February 2019. (76GB after filtering).[4]
- OpenWebText (Gokaslan and Cohen, 2019), an open-source recreation of the WebText corpus described in Radford et al. (2019). The text is web content extracted from URLs shared on Reddit with at least three upvotes. (38GB).[5]
- Stories, a dataset introduced in Trinh and Le (2018) containing a subset of CommonCrawl data filtered to match the story-like style of Winograd schemas. (31GB).

3.3 Evaluation

Following previous work, we evaluate our pretrained models on downstream tasks using the following three benchmarks.

GLUE The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019b) is a collection of 9 datasets for evaluating natural language understanding systems.[6] Tasks are framed as either single-sentence classification or sentence-pair classification tasks. The GLUE organizers provide training and development data splits as well as a submission server and leaderboard that allows participants to evaluate and compare their systems on private held-out test data.

For the replication study in Section 4, we report results on the development sets after finetuning the pretrained models on the corresponding single-task training data (i.e., without multi-task training or ensembling). Our finetuning procedure follows the original BERT paper (Devlin et al., 2019).

In Section 5 we additionally report test set results obtained from the public leaderboard. These results depend on a several task-specific modifications, which we describe in Section 5.1.

SQuAD The Stanford Question Answering Dataset (SQuAD) provides a paragraph of context and a question. The task is to answer the question by extracting the relevant span from the context. We evaluate on two versions of SQuAD: V1.1 and V2.0 (Rajpurkar et al., 2016, 2018). In V1.1 the context always contains an answer, whereas in

[4] We use `news-please` (Hamborg et al., 2017) to collect and extract CC-News. CC-News is similar to the RealNews dataset described in Zellers et al. (2019).

[5] The authors and their affiliated institutions are not in any way affiliated with the creation of the OpenWebText dataset.

[6] The datasets are: CoLA (Warstadt et al., 2018), Stanford Sentiment Treebank (SST) (Socher et al., 2013), Microsoft Research Paragraph Corpus (MRPC) (Dolan and Brockett, 2005), Semantic Textual Similarity Benchmark (STS) (Agirre et al., 2007), Quora Question Pairs (QQP) (Iyer et al., 2016), Multi-Genre NLI (MNLI) (Williams et al., 2018), Question NLI (QNLI) (Rajpurkar et al., 2016), Recognizing Textual Entailment (RTE) (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) and Winograd NLI (WNLI) (Levesque et al., 2011).

V2.0 some questions are not answered in the provided context, making the task more challenging.

For SQuAD V1.1 we adopt the same span prediction method as BERT (Devlin et al., 2019). For SQuAD V2.0, we add an additional binary classifier to predict whether the question is answerable, which we train jointly by summing the classification and span loss terms. During evaluation, we only predict span indices on pairs that are classified as answerable.

RACE The ReAding Comprehension from Examinations (RACE) (Lai et al., 2017) task is a large-scale reading comprehension dataset with more than 28,000 passages and nearly 100,000 questions. The dataset is collected from English examinations in China, which are designed for middle and high school students. In RACE, each passage is associated with multiple questions. For every question, the task is to select one correct answer from four options. RACE has significantly longer context than other popular reading comprehension datasets and the proportion of questions that requires reasoning is very large.

4 Training Procedure Analysis

This section explores and quantifies which choices are important for successfully pretraining BERT models. We keep the model architecture fixed.[7] Specifically, we begin by training BERT models with the same configuration as $\text{BERT}_{\text{BASE}}$ ($L = 12$, $H = 768$, $A = 12$, 110M params).

4.1 Static vs. Dynamic Masking

As discussed in Section 2, BERT relies on randomly masking and predicting tokens. The original BERT implementation performed masking once during data preprocessing, resulting in a single *static* mask. To avoid using the same mask for each training instance in every epoch, training data was duplicated 10 times so that each sequence is masked in 10 different ways over the 40 epochs of training. Thus, each training sequence was seen with the same mask four times during training.

We compare this strategy with *dynamic masking* where we generate the masking pattern every time we feed a sequence to the model. This becomes crucial when pretraining for more steps or with larger datasets.

[7] Studying architectural changes, including larger architectures, is an important area for future work.

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

Table 1: Comparison between static and dynamic masking for $\text{BERT}_{\text{BASE}}$. We report F1 for SQuAD and accuracy for MNLI-m and SST-2. Reported results are medians over 5 random initializations (seeds). Reference results are from Yang et al. (2019).

Results Table 1 compares the published $\text{BERT}_{\text{BASE}}$ results from Devlin et al. (2019) to our reimplementation with either static or dynamic masking. We find that our reimplementation with static masking performs similar to the original BERT model, and dynamic masking is comparable or slightly better than static masking.

Given these results and the additional efficiency benefits of dynamic masking, we use dynamic masking in the remainder of the experiments.

4.2 Model Input Format and Next Sentence Prediction

In the original BERT pretraining procedure, the model observes two concatenated document segments, which are either sampled contiguously from the same document (with $p = 0.5$) or from distinct documents. In addition to the masked language modeling objective, the model is trained to predict whether the observed document segments come from the same or distinct documents via an auxiliary Next Sentence Prediction (NSP) loss.

The NSP loss was hypothesized to be an important factor in training the original BERT model. Devlin et al. (2019) observe that removing NSP hurts performance, with significant performance degradation on QNLI, MNLI, and SQuAD 1.1. However, some recent work has questioned the necessity of the NSP loss (Lample and Conneau, 2019; Yang et al., 2019; Joshi et al., 2019).

To better understand this discrepancy, we compare several alternative training formats:

- SEGMENT-PAIR+NSP: This follows the original input format used in BERT (Devlin et al., 2019), with the NSP loss. Each input has a pair of segments, which can each contain multiple natural sentences, but the total combined length must be less than 512 tokens.

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ are from Yang et al. (2019).

- SENTENCE-PAIR+NSP: Each input contains a pair of natural *sentences*, either sampled from a contiguous portion of one document or from separate documents. Since these inputs are significantly shorter than 512 tokens, we increase the batch size so that the total number of tokens remains similar to SEGMENT-PAIR+NSP. We retain the NSP loss.
- FULL-SENTENCES: Each input is packed with full sentences sampled contiguously from one or more documents, such that the total length is at most 512 tokens. Inputs may cross document boundaries. When we reach the end of one document, we begin sampling sentences from the next document and add an extra separator token between documents. We remove the NSP loss.
- DOC-SENTENCES: Inputs are constructed similarly to FULL-SENTENCES, except that they may not cross document boundaries. Inputs sampled near the end of a document may be shorter than 512 tokens, so we dynamically increase the batch size in these cases to achieve a similar number of total tokens as FULL-SENTENCES. We remove the NSP loss.

Results Table 2 shows results for the four different settings. We first compare the original SEGMENT-PAIR input format from Devlin et al. (2019) to the SENTENCE-PAIR format; both formats retain the NSP loss, but the latter uses single sentences. We find that **using individual sentences hurts performance on downstream tasks**, which we hypothesize is because the model is not able to learn long-range dependencies.

We next compare training without the NSP loss and training with blocks of text from a single document (DOC-SENTENCES). We find that this setting outperforms the originally published $\text{BERT}_{\text{BASE}}$ results and that **removing the NSP loss matches or slightly improves downstream task performance**, in contrast to Devlin et al. (2019). It is possible that the original BERT implementation may only have removed the loss term while still retaining the SEGMENT-PAIR input format.

Finally we find that restricting sequences to come from a single document (DOC-SENTENCES) performs slightly better than packing sequences from multiple documents (FULL-SENTENCES). However, because the DOC-SENTENCES format results in variable batch sizes, we use FULL-SENTENCES in the remainder of our experiments for easier comparison with related work.

4.3 Training with large batches

Past work in Neural Machine Translation has shown that training with very large mini-batches can both improve optimization speed and end-task performance when the learning rate is increased appropriately (Ott et al., 2018). Recent work has shown that BERT is also amenable to large batch training (You et al., 2019).

Devlin et al. (2019) originally trained $\text{BERT}_{\text{BASE}}$ for 1M steps with a batch size of 256 sequences. This is equivalent in computational cost, via gradient accumulation, to training for 125K steps with a batch size of 2K sequences, or for 31K steps with a batch size of 8K.

In Table 3 we compare perplexity and end-

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

task performance of $\text{BERT}_{\text{BASE}}$ as we increase the batch size, controlling for the number of passes through the training data. We observe that training with large batches improves perplexity for the masked language modeling objective, as well as end-task accuracy. Large batches are also easier to parallelize via distributed data parallel training,[8] and in later experiments we train with batches of 8K sequences.

Notably You et al. (2019) train BERT with even larger batche sizes, up to 32K sequences. We leave further exploration of the limits of large batch training to future work.

4.4 Text Encoding

Byte-Pair Encoding (BPE) (Sennrich et al., 2016) is a hybrid between character- and word-level representations that allows handling the large vocabularies common in natural language corpora. Instead of full words, BPE relies on subwords units, which are extracted by performing statistical analysis of the training corpus.

BPE vocabulary sizes typically range from 10K-100K subword units. However, unicode characters can account for a sizeable portion of this vocabulary when modeling large and diverse corpora, such as the ones considered in this work. Radford et al. (2019) introduce a clever implementation of BPE that uses *bytes* instead of unicode characters as the base subword units. Using bytes makes it possible to learn a subword vocabulary of a modest size (50K units) that can still encode any input text without introducing any "unknown" tokens.

[8]Large batch training can improve training efficiency even without large scale parallel hardware through *gradient accumulation*, whereby gradients from multiple mini-batches are accumulated locally before each optimization step. This functionality is supported natively in FAIRSEQ (Ott et al., 2019).

The original BERT implementation (Devlin et al., 2019) uses a character-level BPE vocabulary of size 30K, which is learned after preprocessing the input with heuristic tokenization rules. Following Radford et al. (2019), we instead consider training BERT with a larger byte-level BPE vocabulary containing 50K subword units, without any additional preprocessing or tokenization of the input. This adds approximately 15M and 20M additional parameters for $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, respectively.

Early experiments revealed only slight differences between these encodings, with the Radford et al. (2019) BPE achieving slightly worse end-task performance on some tasks. Nevertheless, we believe the advantages of a universal encoding scheme outweighs the minor degredation in performance and use this encoding in the remainder of our experiments. A more detailed comparison of these encodings is left to future work.

5 RoBERTa

In the previous section we propose modifications to the BERT pretraining procedure that improve end-task performance. We now aggregate these improvements and evaluate their combined impact. We call this configuration **RoBERTa** for **R**obustly **o**ptimized **BERT** **a**pproach. Specifically, RoBERTa is trained with dynamic masking (Section 4.1), FULL-SENTENCES without NSP loss (Section 4.2), large mini-batches (Section 4.3) and a larger byte-level BPE (Section 4.4).

Additionally, we investigate two other important factors that have been under-emphasized in previous work: (1) the data used for pretraining, and (2) the number of training passes through the data. For example, the recently proposed XLNet architecture (Yang et al., 2019) is pretrained using nearly 10 times more data than the original BERT (Devlin et al., 2019). It is also trained with a batch size eight times larger for half as many optimization steps, thus seeing four times as many sequences in pretraining compared to BERT.

To help disentangle the importance of these factors from other modeling choices (e.g., the pretraining objective), we begin by training RoBERTa following the $\text{BERT}_{\text{LARGE}}$ architecture ($L = 24$, $H = 1024$, $A = 16$, 355M parameters). We pretrain for 100K steps over a comparable BOOKCORPUS plus WIKIPEDIA dataset as was used in

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $\text{BERT}_{\text{LARGE}}$. Results for $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

Devlin et al. (2019). We pretrain our model using 1024 V100 GPUs for approximately one day.

Results We present our results in Table 4. When controlling for training data, we observe that RoBERTa provides a large improvement over the originally reported $\text{BERT}_{\text{LARGE}}$ results, reaffirming the importance of the design choices we explored in Section 4.

Next, we combine this data with the three additional datasets described in Section 3.2. We train RoBERTa over the combined data with the same number of training steps as before (100K). In total, we pretrain over 160GB of text. We observe further improvements in performance across all downstream tasks, validating the importance of data size and diversity in pretraining.[9]

Finally, we pretrain RoBERTa for significantly longer, increasing the number of pretraining steps from 100K to 300K, and then further to 500K. We again observe significant gains in downstream task performance, and the 300K and 500K step models outperform $\text{XLNet}_{\text{LARGE}}$ across most tasks. We note that even our longest-trained model does not appear to overfit our data and would likely benefit from additional training.

In the rest of the paper, we evaluate our best RoBERTa model on the three different benchmarks: GLUE, SQuaD and RACE. Specifically we consider RoBERTa trained for 500K steps over all five of the datasets introduced in Section 3.2.

[9]Our experiments conflate increases in data size and diversity. We leave a more careful analysis of these two dimensions to future work.

5.1 GLUE Results

For GLUE we consider two finetuning settings. In the first setting (*single-task, dev*) we finetune RoBERTa separately for each of the GLUE tasks, using only the training data for the corresponding task. We consider a limited hyperparameter sweep for each task, with batch sizes $\in \{16, 32\}$ and learning rates $\in \{1e-5, 2e-5, 3e-5\}$, with a linear warmup for the first 6% of steps followed by a linear decay to 0. We finetune for 10 epochs and perform early stopping based on each task's evaluation metric on the dev set. The rest of the hyperparameters remain the same as during pretraining. In this setting, we report the median development set results for each task over five random initializations, without model ensembling.

In the second setting (*ensembles, test*), we compare RoBERTa to other approaches on the test set via the GLUE leaderboard. While many submissions to the GLUE leaderboard depend on multi-task finetuning, **our submission depends only on single-task finetuning**. For RTE, STS and MRPC we found it helpful to finetune starting from the MNLI single-task model, rather than the baseline pretrained RoBERTa. We explore a slightly wider hyperparameter space, described in the Appendix, and ensemble between 5 and 7 models per task.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

Task-specific modifications Two of the GLUE tasks require task-specific finetuning approaches to achieve competitive leaderboard results.

QNLI: Recent submissions on the GLUE leaderboard adopt a pairwise ranking formulation for the QNLI task, in which candidate answers are mined from the training set and compared to one another, and a single (question, candidate) pair is classified as positive (Liu et al., 2019b,a; Yang et al., 2019). This formulation significantly simplifies the task, but is not directly comparable to BERT (Devlin et al., 2019). Following recent work, we adopt the ranking approach for our test submission, but for direct comparison with BERT we report development set results based on a pure classification approach.

WNLI: We found the provided NLI-format data to be challenging to work with. Instead we use the reformatted WNLI data from SuperGLUE (Wang et al., 2019a), which indicates the span of the query pronoun and referent. We finetune RoBERTa using the margin ranking loss from Kocijan et al. (2019). For a given input sentence, we use spaCy (Honnibal and Montani, 2017) to extract additional candidate noun phrases from the sentence and finetune our model so that it assigns higher scores to positive referent phrases than for any of the generated negative candidate phrases. One unfortunate consequence of this formulation is that we can only make use of the positive training examples, which excludes over half of the provided training examples.[10]

Results We present our results in Table 5. In the first setting (*single-task, dev*), RoBERTa achieves state-of-the-art results on all 9 of the GLUE task development sets. Crucially, RoBERTa uses the same masked language modeling pretraining objective and architecture as $\text{BERT}_{\text{LARGE}}$, yet consistently outperforms both $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$. This raises questions about the relative importance of model architecture and pretraining objective, compared to more mundane details like dataset size and training time that we explore in this work.

In the second setting (*ensembles, test*), we submit RoBERTa to the GLUE leaderboard and achieve state-of-the-art results on 4 out of 9 tasks and the highest average score to date. This is especially exciting because RoBERTa does not depend on multi-task finetuning, unlike most of the other top submissions. We expect future work may further improve these results by incorporating more sophisticated multi-task finetuning procedures.

5.2 SQuAD Results

We adopt a much simpler approach for SQuAD compared to past work. In particular, while both BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) augment their training data with additional QA datasets, **we only finetune RoBERTa using the provided SQuAD training data**. Yang et al. (2019) also employed a custom layer-wise learning rate schedule to finetune

[10] While we only use the provided WNLI training data, our results could potentially be improved by augmenting this with additional pronoun disambiguation datasets.

Model	SQuAD 1.1 EM	SQuAD 1.1 F1	SQuAD 2.0 EM	SQuAD 2.0 F1
Single models on dev, w/o data augmentation				
$\text{BERT}_{\text{LARGE}}$	84.1	90.9	79.0	81.8
$\text{XLNet}_{\text{LARGE}}$	**89.0**	94.5	86.1	88.8
RoBERTa	88.9	**94.6**	**86.5**	**89.4**
Single models on test (as of July 25, 2019)				
$\text{XLNet}_{\text{LARGE}}$			86.3†	89.1†
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			**87.0**†	**89.9**†

Table 6: Results on SQuAD. † indicates results that depend on additional external training data. RoBERTa uses only the provided SQuAD data in both dev and test settings. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively.

XLNet, while we use the same learning rate for all layers.

For SQuAD v1.1 we follow the same finetuning procedure as Devlin et al. (2019). For SQuAD v2.0, we additionally classify whether a given question is answerable; we train this classifier jointly with the span predictor by summing the classification and span loss terms.

Results We present our results in Table 6. On the SQuAD v1.1 development set, RoBERTa matches the state-of-the-art set by XLNet. On the SQuAD v2.0 development set, RoBERTa sets a new state-of-the-art, improving over XLNet by 0.4 points (EM) and 0.6 points (F1).

We also submit RoBERTa to the public SQuAD 2.0 leaderboard and evaluate its performance relative to other systems. Most of the top systems build upon either BERT (Devlin et al., 2019) or XLNet (Yang et al., 2019), both of which rely on additional external training data. In contrast, our submission does not use any additional data.

Our single RoBERTa model outperforms all but one of the single model submissions, and is the top scoring system among those that do not rely on data augmentation.

5.3 RACE Results

In RACE, systems are provided with a passage of text, an associated question, and four candidate answers. Systems are required to classify which of the four candidate answers is correct.

We modify RoBERTa for this task by concatenating each candidate answer with the corresponding question and passage. We then encode each of these four sequences and pass the resulting *[CLS]* representations through a fully-connected layer, which is used to predict the correct answer. We truncate question-answer pairs that are longer than 128 tokens and, if needed, the passage so that the total length is at most 512 tokens.

Model	Accuracy	Middle	High
Single models on test (as of July 25, 2019)			
$\text{BERT}_{\text{LARGE}}$	72.0	76.6	70.1
$\text{XLNet}_{\text{LARGE}}$	81.7	85.4	80.2
RoBERTa	**83.2**	**86.5**	**81.3**

Table 7: Results on the RACE test set. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Yang et al. (2019).

Results on the RACE test sets are presented in Table 7. RoBERTa achieves state-of-the-art results on both middle-school and high-school settings.

6 Related Work

Pretraining methods have been designed with different training objectives, including language modeling (Dai and Le, 2015; Peters et al., 2018; Howard and Ruder, 2018), machine translation (McCann et al., 2017), and masked language modeling (Devlin et al., 2019; Lample and Conneau, 2019). Many recent papers have used a basic recipe of finetuning models for each end task (Howard and Ruder, 2018; Radford et al., 2018), and pretraining with some variant of a masked language model objective. However, newer methods have improved performance by multi-task fine tuning (Dong et al., 2019), incorporating entity embeddings (Sun et al., 2019), span prediction (Joshi et al., 2019), and multiple variants of autoregressive pretraining (Song et al., 2019; Chan et al., 2019; Yang et al., 2019). Performance is also typically improved by training bigger models on more data (Devlin et al., 2019; Baevski et al., 2019; Yang et al., 2019; Radford et al., 2019). Our goal was to replicate, simplify, and better tune the training of BERT, as a reference point for better understanding the relative performance of all of these methods.

7 Conclusion

We carefully evaluate a number of design decisions when pretraining BERT models. We find that performance can be substantially improved by training the model longer, with bigger batches over more data; removing the next sentence prediction objective; training on longer sequences; and dynamically changing the masking pattern applied to the training data. Our improved pretraining procedure, which we call RoBERTa, achieves state-of-the-art results on GLUE, RACE and SQuAD, without multi-task finetuning for GLUE or additional data for SQuAD. These results illustrate the importance of these previously overlooked design decisions and suggest that BERT's pretraining objective remains competitive with recently proposed alternatives.

We additionally use a novel dataset, CC-NEWS, and release our models and code for pretraining and finetuning at: https://github.com/pytorch/fairseq.

References

Eneko Agirre, Llu'is M'arquez, and Richard Wicentowski, editors. 2007. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. KERMIT: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the International Workshop on Paraphrasing*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. http://web.archive.org/save/http://Skylion007.github.io/OpenWebTextCorpus.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

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

Shankar Iyer, Nikhil Dandekar, and Kornl Csernai. 2016. First quora dataset release: Question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for winograd schema challenge. *arXiv preprint arXiv:1905.06290*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6297–6308.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Sebastian Nagel. 2016. Cc-news. `http://web.archive.org/save/http://commoncrawl.org/2016/10/news-dataset-available`.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. FAIRSEQ: A fast, extensible toolkit for sequence modeling. In *North American Association for Computational Linguistics (NAACL): System Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT)*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Association for Computational Linguistics (ACL)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Association for Computational Linguistics (ACL)*, pages 1715–1725.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning (ICML)*.

Yu Stephanie Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xinlun Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

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*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Association for Computational Linguistics (NAACL)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Reducing bert pre-training time from 3 days to 76 minutes. *arXiv preprint arXiv:1904.00962*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *arXiv preprint arXiv:1905.12616*.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*.

Appendix for "RoBERTa: A Robustly Optimized BERT Pretraining Approach"

A Full results on GLUE

In Table 8 we present the full set of development set results for RoBERTa. We present results for a LARGE configuration that follows $\text{BERT}_{\text{LARGE}}$, as well as a BASE configuration that follows $\text{BERT}_{\text{BASE}}$.

B Pretraining Hyperparameters

Table 9 describes the hyperparameters for pretraining of $\text{RoBERTa}_{\text{LARGE}}$ and $\text{RoBERTa}_{\text{BASE}}$

C Finetuning Hyperparameters

Finetuning hyperparameters for RACE, SQuAD and GLUE are given in Table 10. We select the best hyperparameter values based on the median of 5 random seeds for each task.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS
RoBERTa$_{\text{BASE}}$								
+ all data + 500k steps	87.6	92.8	91.9	78.7	94.8	90.2	63.6	91.2
RoBERTa$_{\text{LARGE}}$								
with BOOKS + WIKI	89.0	93.9	91.9	84.5	95.3	90.2	66.3	91.6
+ additional data (§3.2)	89.3	94.0	92.0	82.7	95.6	**91.4**	66.1	92.2
+ pretrain longer 300k	90.0	94.5	**92.2**	83.3	96.1	91.1	67.4	92.3
+ pretrain longer 500k	**90.2**	**94.7**	**92.2**	**86.6**	**96.4**	90.9	**68.0**	**92.4**

Table 8: Development set results on GLUE tasks for various configurations of RoBERTa.

Hyperparam	**RoBERTa$_{\text{LARGE}}$**	**RoBERTa$_{\text{BASE}}$**
Number of Layers	24	12
Hidden size	1024	768
FFN inner hidden size	4096	3072
Attention heads	16	12
Attention head size	64	64
Dropout	0.1	0.1
Attention Dropout	0.1	0.1
Warmup Steps	30k	24k
Peak Learning Rate	4e-4	6e-4
Batch Size	8k	8k
Weight Decay	0.01	0.01
Max Steps	500k	500k
Learning Rate Decay	Linear	Linear
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Gradient Clipping	0.0	0.0

Table 9: Hyperparameters for pretraining RoBERTa$_{\text{LARGE}}$ and RoBERTa$_{\text{BASE}}$.

Hyperparam	**RACE**	**SQuAD**	**GLUE**
Learning Rate	1e-5	1.5e-5	{1e-5, 2e-5, 3e-5}
Batch Size	16	48	{16, 32}
Weight Decay	0.1	0.01	0.1
Max Epochs	4	2	10
Learning Rate Decay	Linear	Linear	Linear
Warmup ratio	0.06	0.06	0.06

Table 10: Hyperparameters for finetuning RoBERTa$_{\text{LARGE}}$ on RACE, SQuAD and GLUE.