SCRIPT: Self-Critic Pretraining of Transformers

Erik Nijkamp[*]
UCLA
enijkamp@ucla.edu

Bo Pang
UCLA
bopang@ucla.edu

Ying Nian Wu
UCLA
ywu@stat.ucla.edu

Caiming Xiong
Salesforce Research
cxiong@salesforce.com

Abstract

We introduce Self-CRItic Pretraining Transformers (SCRIPT) for representation learning of text. The popular masked language modeling (MLM) pretraining methods like BERT replace some tokens with [MASK] and an encoder is trained to recover them, while ELECTRA trains a discriminator to detect replaced tokens proposed by a generator. In contrast, we train a language model as in MLM and further derive a discriminator or critic on top of the encoder without using any additional parameters. That is, the model itself is a critic. SCRIPT combines MLM training and discriminative training for learning rich representations and compute- and sample-efficiency. We demonstrate improved sample-efficiency in pretraining and enhanced representations evidenced by improved downstream task performance on GLUE and SQuAD over strong baselines. Also, the self-critic scores can be directly used as pseudo-log-likelihood for efficient scoring.

1 Introduction

In natural language processing, the landscape of unsupervised learning methods is dominated by masked language modeling (MLM) for bi-directional encoders, such as BERT (Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019; Joshi et al., 2020; Lan et al., 2019; Lewis et al., 2020; Jiao et al., 2019), and causal masking for uni-directional auto-regressive decoders (Radford et al., 2018, 2019; Brown et al., 2020; Raffel et al., 2020; Lewis et al., 2019) such as GPT. In MLM an encoder is pre-trained on a generic corpus of text with the hope of learning universal contextual embeddings, which, then, are fine-tuned on a specific down-stream task. Whereas recent developments in causal masking aim to learn a large-scale model once and define the down-stream task as an auto-regressive manner in the form of few-shot evaluation (Brown et al., 2020). In practice, while an universal auto-regressive neural backbone model without the need for fine-tuning such as GPT-3 is desirable, the computational complexity at inference time remains an open problem. While the two-stage approach of MLM of smaller models is computationally convenient, the pretraining still incurs a substantial computational cost. Hence, in this work, we focus on learning contextual bi-directional representations with the goal of improving upon sample efficiency.

In MLM, the input sequence of tokens is perturbed by randomly masking out a small subset of the identities of tokens (Devlin et al., 2018) or attention scores to those tokens (Yang et al., 2019). Then, the generative model is learned as a denoising auto-encoder (Vincent et al., 2008) which recovers the masked out tokens. While the learned contextual representations achieve remarkable performance on down-stream tasks, the pretraining requires substantial compute. This is mainly due to learning from gradients from the restricted subset of tokens (Clark et al., 2020).

In ELECTRA (Clark et al., 2020), the input sequence is perturbed by replacing a subset of tokens by sampled tokens drawn from an auxiliary generator model in the form of a bi-directional encoder, which itself is learned by MLM. Then, the discriminative model is learned by a binary classification task which detects whether a token is unperturbed or has been replaced. This approach enjoys remarkable sample efficiency, which, we believe, stems primarily from reducing the complexity of the classification task from *masked token prediction* over a large set of classes (i.e., a typical vocabulary size of $30,522$ classes) to *replaced token detection* (i.e., 2 classes).

Despite it being less efficient, MLM training guides the model to learn rich representations. ELECTRA uses MLM only in learning the auxiliary generator which is discarded after pretraining. We propose to combine MLM and discriminative

[*] Research conducted at Salesforce Einstein

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5196–5202
June 6–11, 2021.

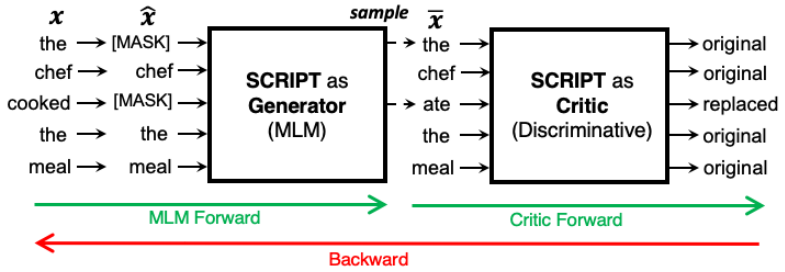

Figure 1: An overview of SCRIPT. We combine MLM and discriminative training in a single transformer encoder, exploiting the rich representations extracted through MLM training and the compute- and sample-efficiency though discriminative training, resulting in a simple yet effective pretraining approach for representation learning. Pretraining starts with replacing a small portion of tokens (e.g., 15%) in a text sequence $\boldsymbol{x}$ with [MASK], yielding $\hat{\boldsymbol{x}}$. The architecture of SCRIPT is a transformer encoder with a softmax output layer, producing a distribution over tokens, same as any MLM models like as BERT. In the MLM forward pass, SCRIPT takes $\hat{\boldsymbol{x}}$ as input and outputs a distribution for each token. This distribution is first used to compute the MLM loss, $\mathcal{L}_{MLM}$, the negative log-likelihood of recovering the masked token. It is then used to construct a Gumbel-Softmax distribution, from which $\bar{\boldsymbol{x}}$ is sampled (indicated by the broken arrows in the figure). The critic forward pass takes $\bar{\boldsymbol{x}}$ as input and goes through the same model. The output softmax distribution is used to construct a binary classifier to discriminate an original versus a replaced token. And the discriminative training loss, $\mathcal{L}_{Disc}$, is simply cross-entropy of the derived binary classifier. Finally, a single backward pass is guided by the combination of $\mathcal{L}_{MLM}$ and $\mathcal{L}_{Disc}$.

training. The resulting model thus has the rich representations from both MLM and discriminative learning and enjoys compute and sample efficiency from its discriminative learning. Furthermore, instead of learning an auxiliary model in addition to the main encoder, our approach learns a single model which is leveraged to recover masked tokens, propose token replacements, and detect replaced tokens. Hence the encoder itself is also a critic, giving the name of our model, Self-CRItic Pretraining Transformers (SCRIPT). Our experiments show that SCRIPT has improved compute and sample efficiency in pretraining and enhanced representations, hence outperforming strong baselines in fine-tuning on downstream tasks.

Contributions. (1) We propose a novel pretraining approach in which the model acts as a self-critic. (2) We demonstrated improved downstream task performance over state-of-the-art under computational constraints. (3) We show the self-critic scores may serve as computationally efficient pseudo-log-likelihood for scoring tasks.

2 Method

We propose a pretraining approach which combines masked token recovery and replaced token detection and does not introduce any additional parameters compared to a regular BERT. In the following sections, we first introduce MLM training which is the same as that in BERT, and then present self-critic training.

Suppose $\boldsymbol{x} = [x_1, ..., x_t, ..., x_T]$ is a text sequence where x_t is the tth token. In MLM training, a portion of tokens (e.g., 15%) are replaced with a special token [MASK]. Let $\hat{\boldsymbol{x}}$ be the sequence after the mask replacement and $e(\hat{\boldsymbol{x}}) = \{e_t \in \mathbb{R}^d\}_{t=1}^T$ be the contextual representations computed by the transformer. Let $W \in \mathbb{R}^{V \times d}$ be the weight matrix of a softmax layer where V is the vocabulary size. The logit or score for token t is $s_t = We_t \in \mathbb{R}^V$. Then the log-likelihood of the sequence $\boldsymbol{x}$ is,

$$\log p_\theta(\boldsymbol{x}|\hat{\boldsymbol{x}}) = \sum_{t=1}^{T} m_t \log p_\theta(x_t|\hat{\boldsymbol{x}}) \quad (1)$$

$$= \sum_{t=1}^{T} m_t \log \frac{\exp(s_{tv})}{\sum_{v'=1}^{V} \exp(s_{tv'})} \quad (2)$$

where $m_t \in \{0, 1\}$ indicates whether x_t is a masked token, [MASK]. The loss function for MLM is the negative log-likelihood $\mathcal{L}_{MLM}(\theta) = -\mathbb{E}_{p_{data}(\boldsymbol{x})} \log p_\theta(\boldsymbol{x}|\hat{\boldsymbol{x}})$ where p_{data} is the empirical data distribution.

Besides defining the log-likelihood for MLM training, $p_\theta(x_t|\hat{\boldsymbol{x}})$ naturally provides a conditional distribution of x_t with which we can construct a sampled sequence, $\bar{\boldsymbol{x}} = [\bar{x}_1, ..., \bar{x}_T]$, by replacing x_t with $\bar{x}_t$, a token sampled from $p_\theta(x_t|\hat{\boldsymbol{x}})$. x_t is replaced only if it is masked in $\hat{\boldsymbol{x}}$ (i.e., $m_t = 1$). In particular, the replacement token is sampled from a Gumbel-Softmax distribution (Jang et al., 2016). Let $\pi = \{\pi_v\}_{v=1}^V$ denote $p_\theta(x_t|\hat{\boldsymbol{x}})$ for notational clarity. Then the probability of sampling the vth token in the vocabulary for x_t is,

$$p(\bar{x}_t|\hat{\boldsymbol{x}}) = \frac{\exp[(\log \pi_v + g_v)/\tau]}{\sum_{v'=1}^{V} \exp[(\log \pi_{v'} + g_{v'})/\tau]} \quad (3)$$

where $\{g_{v'}\}_{v'=1}^{V}$ are i.i.d. samples drawn from Gumbel$(0,1)$[1] and τ is the temperature for sampling. The Gumbel-Softmax distribution π approaches one-hot when τ is small (e.g., $\tau = 0.1$) and uniform when τ is large (e.g., $\tau = 10.0$).

To apply discriminative training to the model, we derive a discriminator from the existing model and parameters. $\bar{x}_t$ is considered as a positive token if $\bar{x}_t = x_t$, while deemed a negative token if $\bar{x}_t \neq x_t$. In the MLM training, the last layer defines a V-class classifier with the parameters W. We can augment W with an extra row for computing the score or logit for the negative token class, making it classify $V+1$ classes. Denote the augmented weight matrix as W^+. Then the classification logits are $s_t^+ = W^+ e_t \in \mathbb{R}^{V+1}$. However, it is unnecessary to bring in new parameters and over-parameterization since subtracting an arbitrary function $f(e_t) \in \mathbb{R}$ from all the logits, $s_{tv}^+ - f(e_t)\ \forall v = 1, ..., V+1$, does not change the softmax output. Thus we fix the last row of W^+ to all zeros $\mathbf{0} \in \mathbb{R}^{1\times d}$. Then we have the logit for the tth token,

$$s_t^+ = W^+ e_t = \begin{cases} W e_t = s_t, \text{ for } x_t \in \{1, ..., V\} \\ 0, \text{ otherwise.} \end{cases}$$

Then the probability of the tth token in $\bar{\boldsymbol{x}}$ being a negative token is,

$$p(t^-|\bar{\boldsymbol{x}}) = \frac{1}{\sum_{v'=1}^{V} \exp(s_{tv'}) + 1} \quad (4)$$

while the probability being a positive token is,

$$p(t^+|\bar{\boldsymbol{x}}) = \frac{\sum_{v'=1}^{V} \exp(s_{tv'})}{\sum_{v'=1}^{V} \exp(s_{tv'}) + 1} \quad (5)$$

where t^- and t^+ indicate $\bar{x}_t$ is a positive token and a negative token, respectively. The generator *per se* is thus also a critic or discriminator for replaced token detection, giving the name of our model, self-critic. The loss of discriminative training is simply the cross-entropy loss,

$$\mathcal{L}_{Disc}(\theta) = -\mathbb{E}_{p_{data}}[\sum_{t=1}^{T} \mathbb{1}(t^+) \log p(t^+|\bar{\boldsymbol{x}}) + \mathbb{1}(t^-) \log p(t^-|\bar{\boldsymbol{x}})]. \quad (6)$$

The overall loss function of SCRIPT combines MLM and discriminative training, $\mathcal{L}_\theta = \mathcal{L}_{MLM}(\theta) + \alpha \mathcal{L}_{Disc}(\theta)$, where α is an coefficient determining the strength of discriminative training. The learning of SCRIPT involves two forward passes through a single model, one for MLM with $\hat{\boldsymbol{x}}$ as input, one for discriminative training with $\bar{\boldsymbol{x}}$ as input, and a single backward pass. Figure 1 gives an overview of our model.

[1]The Gumbel$(0,1)$ distribution can be sampled using inverse transform sampling by drawing $u \sim$ Uniform$(0,1)$ and computing $g = -\log(-\log u)$

3 Experiments

In the subsequent empirical evaluations, we shall address the following questions: (1) Does the learning as self-critic lead to competitive down-stream task performance? (2) Can we treat the self-critic scores as pseudo-log-likelihoods? (3) Is the sample efficiency improved over state-of-the-art baselines?

Hence, we train and evaluate two SCRIPT models "small" and "base" with an encoder of the 14M and 110M parameters, respectively. For a direct comparison, the models are trained on the OpenWebText corpus (Gokaslan and Cohen, 2019) with identical pre-processing and optimization procedures as in (Devlin et al., 2018) and (Clark et al., 2020). We refer to the Appendix for details.

3.1 Transfer to Downstream Tasks

We evaluate the efficacy of our method on the GLUE natural language understanding benchmark (Wang et al., 2018) and the SQuAD 1.1 and 2.0 question answering dataset (Rajpurkar et al., 2016a). We report mean scores of GLUE tasks over 8 fine-tuning runs with varying random seed. For the evaluation on SQuAD, we re-trained the "small" models with a sequence length of 512 tokens. Table 1 depicts improved scores across the benchmarks. The task specific GLUE scores are shown in Table 2.

	GLUE	**SQuAD 1.1**		**SQuAD 2.0**	
Model	Mean	EM	F1	EM	F1
ELECTRA-small	80.38	74.13	81.65	65.91	68.59
SCRIPT-small	**81.32**	**74.84**	**82.43**	**67.03**	**69.81**
ELECTRA-base	85.06	84.57	90.72	80.86	83.52
SCRIPT-base	**85.76**	**85.43**	**91.56**	**81.74**	**84.25**

Table 1: GLUE and SQuAD dev-set scores for models pretrained on OpenWebText with identical pre-processing and optimization.

3.2 Efficient Pseudo-Log-Likelihood Scoring

In contrast to MLM and ELECTRA pretraining, SCRIPT allows for efficient computation of

Model	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Mean
BERT-small	14M	38.40	88.99	84.55	84.20	87.67	78.07	85.75	61.01	76.08
ELECTRA-small	14M	56.82	88.37	87.41	86.82	88.30	78.94	87.92	**68.51**	80.38
SCRIPT-small (ours)	14M	**59.46**	**89.56**	**88.23**	**87.16**	**89.38**	**80.30**	**88.04**	68.42	**81.32**
BERT-base	110M	51.72	92.83	83.93	83.9	88.75	84.55	89.91	65.98	80.19
ELECTRA-base	110M	64.36	91.03	88.23	**90.18**	91.33	86.21	92.01	77.16	85.06
SCRIPT-base (ours)	110M	**65.04**	**93.09**	**90.08**	90.01	**91.43**	**86.88**	**92.29**	**77.23**	**85.76**

Table 2: Comparison of small and base models on the GLUE dev set. The models were trained on the OpenWebText corpus (Gokaslan and Cohen, 2019) for $1,000,000$ and $766,000$ steps, respectively. The GLUE task scores are means of 8 runs over a set of random seeds. SCRIPT outperforms ELECTRA while enjoying a simple architecture and learning algorithm.

a pseudo-log-likelihood (PLL) for a given sequence $\boldsymbol{x}$,

$$\mathrm{PLL}(\boldsymbol{x}) = \sum_{t=1}^{T} \log p(t^{+}|\boldsymbol{x}). \tag{7}$$

The PLL allows for the re-ranking of a set of sequences produced by a NMT or ASR system. While language models seem a natural fit for a ranking problem, Salazar et al. (2019) show improved performance when ranking is based on the PLL. However, for a sequence with T tokens, this would require T forward passes as each token has to be masked out. Instead, we propose to recruit (7) as a measure of PLL. Table 3 compares the word error rates (WER) on the LibriSpeech dataset after rescoring. SCRIPT performs competitively while (7) is computed as a single forward pass.

	dev		test	
Model	clean	other	clean	other
baseline (1-best)	7.17	19.79	7.26	20.37
oracle (100-best)	2.85	12.21	2.81	12.85
uni-SANLM	6.08	17.32	6.11	18.13
bi-SANLM	5.52	16.61	5.65	17.44
BERT-small	5.65	16.97	5.80	17.70
SCRIPT-small	5.79	17.02	6.12	17.83

Table 3: WERs on LibriSpeech after rescoring. Baseline, SANLM, and oracle numbers are from Shin et al. (2019).

3.3 Computational Efficiency

Wall-clock time. We compare the number of training steps per second. For direct comparison, we modify the ELECTRA reference code[2]. For TPU v3 with 8 TPU cores, ELECTRA and SCRIPT achieve 31.3 and 22.7 training iterations per second with a mean MXU utilization of 14.93% and 17.91% for small models, respectively.

GLUE. Figure 2 depicts the improvement in the mean GLUE scores for ELECTRA-small and SCRIPT-small over the number of training steps. While the wall-clock time per computational training step of SCRIPT is increased over ELECTRA, the sample-efficiency of SCRIPT in terms of the mean GLUE score over training steps is higher. Hence, the efficiency of both methods may be comparable, however, SCRIPT achieves improved overall performance on GLUE.

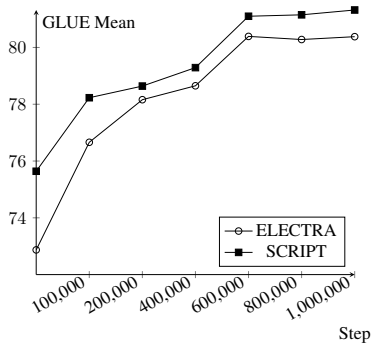

Figure 2: Comparison between ELECTRA-small and SCRIPT-small on the GLUE mean score over training steps on the OpenWebText corpus.

4 Conclusion

This work presents SCRIPT for representation learning. It is a transformer encoder like BERT. In pretraining, it recovers masked tokens, proposes negative samples, and acts as a self-critic, discriminating between sampled and original tokens. The joint MLM and discriminative learning improves sample efficiency in pretraining and enhances representation learning, leading to improved performance over strong baselines on various downstream tasks. It also provides an efficient way for computing pseudo-log-likelihood for scoring tasks and achieves competitive performance.

[2] https://github.com/google-research/electra

References

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*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

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. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B. Dolan. 2007. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL@ACL*.

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

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First Quora dataset release: Question pairs.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

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. In *International Conference on Learning Representations*.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

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*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016a. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy S. Liang. 2016b. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. 2019. Masked language model scoring. *arXiv preprint arXiv:1910.14659*.

Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. 2019. Effective sentence scoring method using bert for speech recognition. In *Asian Conference on Machine Learning*, pages 1081–1093.

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 *EMNLP*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

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. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*.

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*, pages 5753–5763.

Appendix

4.1 Experiment Details

We describe the configuration used for pretrainined and fine-tuning below.

Pre-Training Hyperparameters. We largely use the same hyperparameters as BERT and ELECTRA. The coefficient for discriminative learning, α, is set to be 50. We use dynamic token masking with the masked positions decided on-the-fly. Among the 15% tokens selected for masking, 80% are replaced with [MASK], 10% are kept to be the same, 10% are replaced with a random token. The full set of hyperparameters are displayed in Table 4.

Fine-Tuning Hyperparameters. We follow the fine-tuning hyperparameters used in ELECTRA. The full set of hyperparameters is listed in Table 5.

4.2 GLUE Description

Each subtask of GLUE is described below.

MNLI. Multi-genre Natural Language Inference (Williams et al., 2018). Given a pair of sentences, the task is to predict whether whether the second sentence is an entailment, contradiction, or neutral with respect to the first one.

QQP. Quora Question Pairs (Iyer et al., 2017). The task is to determine whether a pair of questions asked on Quora are semantically equivalent.

QNLI. Question Natural Language Inference. It is a binary classification task constructed from SQuAD (Rajpurkar et al., 2016b). The task is to predict whether a context sentence contains the answer to a question sentence.

SST. Stanford Sentiment Treebank (Socher et al., 2013). This task is binary task to determine if a sentence is positive or negative in sentiment.

STS. Semantic Textual Similarity (Cer et al., 2017). The tasks is to predict how similar two sentences are on a 1-5 scale in terms of semantic meaning.

CoLA. Corpus of Linguistic Acceptability (Warstadt et al., 2018). The task is to determine whether a given sentence is linguistically "acceptable".

MRPC. Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005). The task is to predict whether two sentences are semantically equivalent.

RTE. Recognizing Textual Entailment (Giampiccolo et al., 2007). Given a premise and a hypothesis, the task is to predict whether the premise entails the hypothesis.

Hyperparameter	**Small**	**Base**
Number of layers	12	12
Hidden Size	256	768
FFN inner hidden size	1024	3072
Attention heads	4	12
Attention head size	64	64
Embedding Size	128	768
Mask percent	15	15
Learning Rate Decay	Linear	Linear
Warmup steps	10000	10000
Learning Rate	5e-4	2e-4
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
Attention Dropout	0.1	0.1
Dropout	0.1	0.1
Weight Decay	0.01	0.01
Batch Size	128	256

Table 4: Pre-train hyperparameters.

Hyperparameter	Value
Learning Rate	3e-4 for Small, 1e-4 for Base
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Layerwise LR decay	0.8
Learning rate decay	Linear
Warmup fraction	0.1
Attention Dropout	0.1
Dropout	0.1
Weight Decay	0
Batch Size	32
Train Epochs	10 for RTE and STS, 2 for SQuAD, 3 for other tasks

Table 5: Fine-tune hyperparameters.