MC-BERT: Efficient Language Pre-Training via a Meta Controller

Zhenhui Xu[*]
Peking University
zhenhui.xu@pku.edu.cn

Linyuan Gong[*]
Peking University
gonglinyuan@hotmail.com

Guolin Ke
Microsoft Research
zhenhui.xu@pku.edu.cn

Di He
Peking University
dihe@microsoft.com

Shuxin Zheng
Microsoft Research
Shuxin.Zheng@microsoft.com

Liwei Wang
Peking University
wanglw@cis.pku.edu.cn

Jiang Bian
Microsoft Research
jiang.bian@microsoft.com

Tie-Yan Liu
Microsoft Research
Tie-Yan.Liu@microsoft.com

Abstract

Pre-trained contextual representations (e.g., BERT) have become the foundation to achieve state-of-the-art results on many NLP tasks. However, large-scale pre-training is computationally expensive. ELECTRA, an early attempt to accelerate pre-training, trains a discriminative model that predicts whether each input token was replaced by a generator. Our studies reveal that ELECTRA's success is mainly due to its reduced complexity of the pre-training task: the binary classification (replaced token detection) is more efficient to learn than the generation task (masked language modeling). However, such a simplified task is less semantically informative. To achieve better efficiency and effectiveness, we propose a novel meta-learning framework, MC-BERT. The pre-training task is a multi-choice cloze test with a reject option, where a meta controller network provides training input and candidates. Results over GLUE natural language understanding benchmark demonstrate that our proposed method is both efficient and effective: it outperforms baselines on GLUE semantic tasks given the same computational budget.

1 Introduction

In natural language processing, pre-trained contextual representations are widely used to help downstream tasks without sufficient labeled data. Previous works [15, 22, 4, 12] train contextual language representations on self-supervised generation tasks. For example, BERT [4] randomly masks[2] a small subset of the unlabeled input sequence and trains a generator to recover the original input. Such tasks require only unlabeled free texts, and Raffel et al. [16] shows that a large dataset is crucial to a pre-trained model's performance. Pre-training over such large-scale data consumes huge computational resources, which raises a critical concern in terms of high energy cost [18].

ELECTRA [3] is a successful attempt to boost the efficiency of pre-training. The learning framework of ELECTRA consists of a discriminator and a generator. Given a sentence, it corrupts the sentence by replacing some words with plausible alternatives sampled from the generator. Then, the discriminator

[*]Equal contribution. Works done while interning at Microsoft Research Asia.

[2]In BERT, among all tokens to be predicted, 80% of tokens are replaced by the `[MASK]` token, 10% of tokens are replaced by a random token, and 10% of tokens are unchanged.

Preprint. Under review.

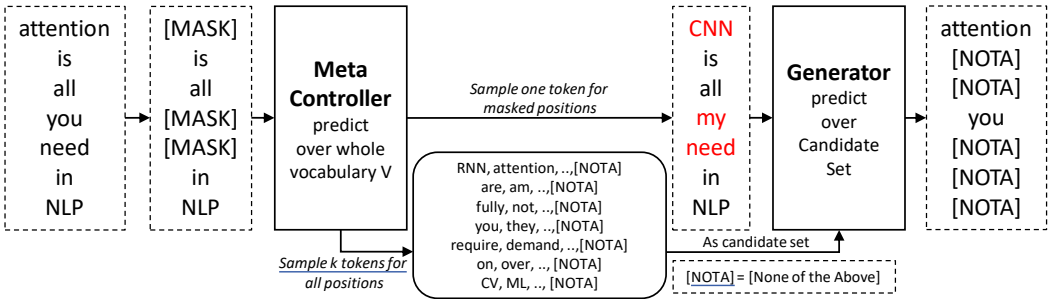

Figure 1: The learning framework of MC-BERT. Given a sentence, the meta controller first corrupts the sentence by replacing a small subset of tokens with sampled plausible alternatives. It then creates k token candidates for each position. The generator uses the corrupted sentence as input and learns to correct each word by predicting over the k candidates [19].

is trained to predict whether a word in the corrupted sentence was replaced by the generator. Finally, the learned discriminator will be used in downstream tasks. Unlike previous generation tasks where the model makes predictions only on a small number (e.g., 15% in BERT) of masked positions, the discriminative task proposed in ELECTRA is defined over all input tokens. According to Clark et al. [3], this approach has better sample efficiency and, consequently, accelerated training.

In Section 3, we provide empirical studies on ELECTRA, showing that ELECTRA has a vital advantage of reducing the complexity of the pre-training task: the *replaced token detection* task of ELECTRA is a simple binary classification. It is easier to learn than generation tasks (i.e., predicting one word from the entire vocabulary), such as *masked language modeling (MLM)* used by BERT. We trained two variants of ELECTRA. We first replace the simple discriminative task by a more complex task, and this modification significantly slows down the convergence. Then, we train discriminator only on a sampled subset of positions, and the convergence is not impacted significantly. These empirical studies show that for efficient training, reduced task complexity is much more important than sample efficiency. Still, the replaced token detection task of ELECTRA is less informative than generation tasks. The semantic information required to detect replaced tokens is not as much as recovering the original input. Detailed analysis on GLUE natural language understanding benchmark shows that ELECTRA's advantage over BERT is less significant on semantic-related tasks than on syntax-related tasks.

In Section 4, we propose **MC-BERT**, a novel language pre-training method using a **M**eta **C**ontroller to manage the training of a generator, as shown in Figure 1. This pre-training task is comparable to multiple-choice cloze tests. Unlike BERT, the MC-BERT generator only needs to make a k-way classification, which reduces the task complexity. Unlike ELECTRA, MC-BERT still trains a generator, learning more semantic information.

In Section 5, to compare with other models, we conduct experiments and evaluate them over GLUE natural language understanding benchmark [21]. Results show that MC-BERT is more efficient and achieves better accuracy than other baselines on most of the semantic understanding tasks.

2 Background

Current state-of-the-art natural understanding systems learn pre-trained contextual representations by encoding the word's surrounding context. The encoders are trained by self-supervised tasks using large-scale unlabeled corpora. For instance, Peters et al. [13], Radford et al. [14] train language models using LSTMs [9] or Transformer decoders [20], and use the hidden states in the networks as the contextual representation. Devlin et al. [4], Liu et al. [12] use the *masked language modeling* task and achieve state-of-the-art performance on natural language understanding tasks. Alternatively, XLNet [22] and UniLM models [5] design permuted and bidirectional language modeling tasks.

The exploding demand of computations, together with the resulting massive energy cost [18], has become an obstacle to the application of pre-training. Unfortunately, to the best of our knowledge, there is a limited number of works aiming at improving the training efficiency of such models. You

et al. [23] attempts to accelerate BERT pre-training, but it has to pay back with massive computational resources. Gong et al. [8] observes that parameters of BERT in different layers have structural similarity and reduce training time using implicit parameter sharing. A notable improvement is ELECTRA [3], the starting point of our work. We will discurss ELECTRA in detail in Section 3.

3 A Deep Dive into ELECTRA

ELECTRA consists of a generator network G and a discriminator network D, both of which use Transformer encoders as their backbone. Formally, we use V to denote the vocabulary of tokens; we use $x = (x_1, ..., x_n)$ to denote a sentence of n tokens, where $x_i \in V$, $i = 1, 2, \cdots, n$; $x^M = \text{Mask}(x, p)$ denotes a masked sentence of x in which the MASK operator randomly replaces the token at each position by a mask symbol `[MASK]` with an equal probability p.

Let x^M be the input, at each masked position, the generator G learns to predict the correct token from the vocabulary: for any masked position i in x^M, let $P(v|x^M, i; G, V)$ be the probability that G predicts $v \in V$ as the missing token, satisfying $\sum_{v \in V} P(v|x^M, i; G, V) = 1$. We use $P(\cdot|x^M, i; G, V)$ to denote this probability distribution over V. The generator G is trained to minimize the MLM loss as

$$L_{\text{MLM}}(x; G) = \mathbb{E}\Big(\sum_{i: x_i^M = \texttt{[MASK]}} -\log P(x_i|x^M, i; G, V)\Big), \tag{1}$$

where the expectation is taken over the random draw of masked positions. Other details of the generator G can be referred to in Devlin et al. [4], Clark et al. [3].

In ELECTRA, the generator G predicts the missing tokens and fill the corresponding masked positions, but the predictions may differ from the original sentence. We denote the sentence generated by G as $x^R = \text{Replace}(x^M, G)$, in which each token $x_i^R, (i = 1, \cdots, n)$ is defined as

$$x_i^R = \text{Replace}(x^M, G) = \begin{cases} x_i^M, \text{if } x_i^M \neq \texttt{[MASK]}. \\ v \sim P(\cdot|x^M, i; G, V), \text{if } x_i^M = \texttt{[MASK]}. \end{cases} \tag{2}$$

The discriminator D learns to classify whether each token in x^R is the same as the original one. To achieve this, D uses Transformer encoder to get the contextual representations $(h_1, \cdots, h_n)$, where $h_i \in R^d$ is a d-dimension contextual embedding for position i. Then, D introduces a binary classifier with parameters $w \in R^d$, to decide the probability of whether x_i^R is the same as the original one, i.e.,

$$P(x_i^R \text{ is original}; D) = \text{sigmoid}(w^T h_i), \tag{3}$$

The learning objective of D is to minimize the classification error, formally

$$\begin{aligned} & L_{\text{DISC}}(x, x^R; D) \\ =& \mathbb{E}(-\sum_{i=1}^{n}[\mathbf{1}(x_i^R = x_i) \log P(x_i^R \text{ is original}; D) \\ & + \mathbf{1}(x_i^R \neq x_i) \log(1 - P(x_i^R \text{ is original}; D))]) \end{aligned} \tag{4}$$

The generator G and discriminator D are jointly optimized according to Eq. 1 and 4. After training, the discriminator D will be used in downstream tasks.

3.1 The Real Advantage of ELECTRA over BERT

Clark et al. [3] claims that ELECTRA yields higher training efficiency than BERT due to *higher sample efficiency*. While the MLM loss (Eq. 1) of BERT is calculated over a sampled masked subset of positions (e.g., 15%), the loss of the discriminator in ELECTRA (Eq. 4) is calculated over all input positions. Therefore, the learning signals enclosed in more positions could be used to optimize the model parameters, resulting in more efficient training.

However, there is another critical difference between BERT and ELECTRA: BERT learns to predict the correct word from the entire vocabulary V, whose size is tens of thousands. On the contrary,

ELECTRA's discriminator learns from a much simpler pre-training task, i.e., predicting whether each word is replaced or not. The *reduced task complexity* may also lead to training acceleration.

Given the above two crucial differences between ELECTRA and BERT, we conduct controlled experiments to examine which of them is more critical for efficient training.

Experimental setup We conduct experiments to analyze the effects of higher sample efficiency or reduced task complexity on training efficiency. We use the same dataset, model architectures, and other hyperparameters as ELECTRA-Base [3]. The pre-trained models are evaluated on GLUE benchmark (**G**eneral **L**anguage **U**nderstanding **E**valuation) [21]. We leave detailed experiment setups in Section 5.1.

To study the effects of sample efficiency, we design a modified version of ELECTRA, called *ELECTRA-sample*. Unlike ELECTRA that calculates the loss of D over all input positions, ELECTRA-sample only calculates the loss over 50% of input positions (all masked positions plus a sampled subset of non-masked positions). ELECTRA-sample has lower sample efficiency than the original ELECTRA, but it keeps the same task complexity. If sample efficiency is essential to training efficiency, we can expect ELECTRA-sample's worse performance compared to ELECTRA.

To study whether a more complex pre-training task will reduce ELECTRA's training efficiency, we design a modified version of ELECTRA, called *ELECTRA-complex*. Instead of training the model to check whether each word in a corrupted sentence is replaced, ELECTRA-complex learns to predict the correct word from the entire vocabulary at each position. If the task simplification is essential for ELECTRA's success, we can expect much slower training by ELECTRA-complex.

Results As we study training efficiency, we focus on each model's performance of the first several epochs. For all experiments, we dump four checkpoints at 20k, 50k, 100k, 200k steps, corresponding to 2%, 5%, 10%, 20% of all pre-training steps. All checkpoints are then fine-tuned on three downstream tasks, CoLA, RTE, and STS-B.

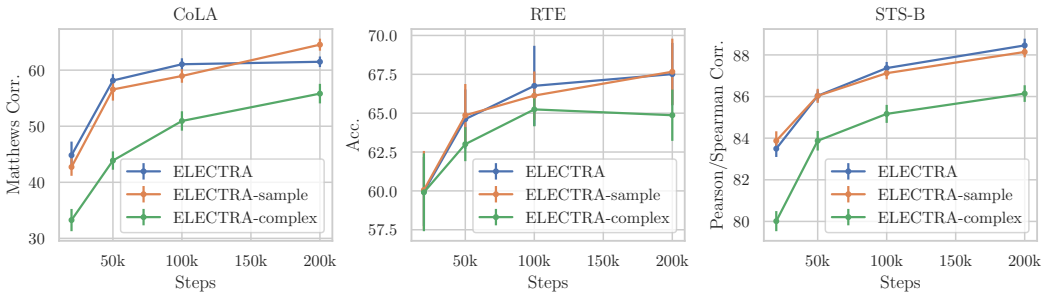

Figure 2: Performance of modified ELECTRA models on downstream tasks.

From Figure 2, we can see that ELECTRA-sample's performance is only slightly worse than ELECTRA in most of the checkpoints, although its sample efficiency is halved. This fact indicates that sample efficiency has little impact on the performance of the model.

However, from Figure 2, we can see that ELECTRA-complex's performance is consistently worse than ELECTRA by a large margin in almost every checkpoint. This fact indicates that reducing task complexity is important to improving pre-training efficiency.

Drawbacks of the discriminative task It is worth noting that the discriminative task is not as informative as the generation task. Formally, we denote random variable X as a sentence with any underlying distribution and X^R as the corrupted sentence. We define a binary vector Z where $Z_i = I[X_i = X_i^R]$. Z, therefore, is the target of the discriminative task. We have the conditional entropy $H(Z|X, X^R) = 0$ since Z is a deterministic function of X and X^R. Then, it is straightforward to see that $H(X|X^R) = H(X, Z|X^R) > H(Z|X^R)$.

Empirical results in Table 3 of Clark et al. [3] and in Table 3 of this paper also show that ELECTRA's advantage over BERT mainly lies in syntactic tasks (CoLA) instead of semantic tasks, which require

the model to capture richer semantic information. These facts inspire us to design more informative pre-training tasks beyond ELECTRA.

4 Pre-training with a Meta Controller

In this section, we introduce a novel pre-training method, *MC-BERT*. We still pre-train a generator (instead of a discriminator) to learn more semantic information, but we use a *meta controller* to improve its training efficiency. We continue to use all notations defined in Section 3 in this section.

4.1 MC-BERT

Our method trains two Transformer encoders, a generator G_{model} and a meta controller G_{ctrl}. The generator G_{model} is served as the primary model and will be further used in the downstream tasks, while the meta controller guides the training of the generator.

The meta controller G_{ctrl} is trained using the MLM loss defined in Eq. 1. Given an input sentence x, the meta controller guides the training of the generator G_{model} in two ways:

- Similar to ELECTRA, G_{ctrl} generates an corrupted sentence $x^R = \text{Replace}(x^M, G_{\text{ctrl}})$ as is shown in Eq. 2
- G_{ctrl} creates a set of token candidates for each position $\bar{V}_i$, and each $|\bar{V}_i| = k$, where k is a small integer.

The generator G_{model} uses x^R as input and learns to correct the sentence using the given candidates $\bar{V}_i$ for each position i. In the following, we denote $\bar{V} = (\bar{V}_1, \bar{V}_2, ..., \bar{V}_n)$, the tuple of all candidate sets.

Label Leaking and Reject Options It is non-trivial to construct a meaningful $\bar{V}_i$ for training G_{model}. First, V_i should contain useful negative candidates, which can provide G_{model} with informative signals for learning. Moreover, the learning process may suffer from *label leaking*. Concretely, if the ground truth token appears in the candidate set $\bar{V}_i$ of every non-replaced position, the generator G_{model} can easily make correct predictions by choosing the input token, since the ground truth is always the same as the input token for a non-replaced position. Because most positions are non-replaced, this problem leads to ineffective training of G_{model}. However, we cannot fix this problem by removing the ground truth token from $\bar{V}_i$, since it will result in an invalid classification task, where no candidate is correct.

To address this problem, we use a novel way to construct $\bar{V}$ motivated by the history of voting [6, 1]. We introduce a special category, "None of the above" (`[NOTA]`), as a reject option. Given a corrupted sentence x^R, for position i, if $x_i^R = x_i$ (when position i is not masked, or the prediction of G_{ctrl} is correct), we sample $k-1$ negative tokens without replacement according to G_{ctrl}, using them together with `[NOTA]` as $\bar{V}_i$. In this case, ~~we hope G_{model} can select `[NOTA]` from $\bar{V}_i$,~~ indicating the input token is correct. If $x_i^R \neq x_i$ (when position i is masked and the prediction of G_{ctrl} is wrong), we sample $k-2$ negative tokens according to G_{ctrl}, using them together with $\{\texttt{[NOTA]}, x_i\}$ as $\bar{V}_i$. In this case, ~~we hope G_{model} can choose x_i from $\bar{V}_i$~~. Formally, we construct $\bar{V}_i$ as is described below.

$$\bar{V}_i = \begin{cases} \{v_i^1, ..., v_i^{k-1}\} \cup \{\texttt{[NOTA]}\}, \text{if } x_i^R = x_i. \\ \{v_i^1, ..., v_i^{k-2}\} \cup \{x_i, \texttt{[NOTA]}\}, \text{if } x_i^R \neq x_i. \end{cases}, \tag{5}$$

All negatives $v_i^j \sim P(\cdot|x^M, i; G_{\text{ctrl}}, V)$ $(j = 1, 2, \cdots)$ are drawn without replacement. We use $P(\cdot|x^R, i; G_{\text{model}}, \bar{V}_i)$ to denote the output distribution of G_{model} over $\bar{V}_i$. Given contextual representations h_i produced by G_{model}, and the token embedding matrix $\hat{\text{Emb}}_i$ (including `[NOTA]`) in $\hat{V}_i$,

$$P(\cdot|x^R, i; G_{\text{model}}, \bar{V}_i) = \text{Softmax}(\hat{\text{Emb}}_i^T h_i), \tag{6}$$

The loss function of G_{model} is defined as the negative log likelihood for a k-class classification problem.

$$\begin{aligned} & L_{\text{model}}(x, x^R; G_{\text{model}}; \bar{V}) \\ =& \mathbb{E}(-\sum_{i=1}^{n}[\mathbf{1}(x_i^R = x_i) \log P(\texttt{[NOTA]}|x^R, i; G_{\text{model}}, \bar{V}_i) \\ +& \mathbf{1}(x_i^R \neq x_i) \log P(x_i|x^R, i; G_{\text{model}}, \bar{V}_i)]). \end{aligned} \tag{7}$$

We optimize a combined loss of Eq. 1 and Eq. 7:

$$\min_{G_{\text{model}}, G_{\text{ctrl}}} L_{\text{MLM}}(x; G_{\text{ctrl}}) + \lambda L_{\text{model}}(x, x^R; G_{\text{model}}; \bar{V}). \tag{8}$$

4.2 Discussions

Table 1: Example of the task comparisons between BERT/ELECTRA and our proposed MC-BERT.

Ground Truth: *He is overweight as he eats a lot.*			
Model	**Question**	**Choices**	**Answer**
BERT	*He is ____ as he eats a lot.*	All tokens: *abandon, able, about, ...*	*overweight*
ELECTRA	*He is a as he eats a lot.*	Right, Wrong	Wrong
MC-BERT	*He is tiny as he eats a lot.*	A. *overweight* B. *healthy* C. *smart* D. *None of the above*	A

The example in Table 1 illustrates the difference between MC-BERT and BERT/ELECTRA in terms of their pre-training tasks. From Table 1, we can see that BERT solves a general cloze problem: it masks some tokens and requires the learner to pick correct tokens from the entire vocabulary. The task is very complex. ELECTRA learns from detecting replaced tokens, which is a binary classification problem similar to grammar checking. This task is less complex, but the learning signal of ELECTRA is less informative.

Our MC-BERT is similar to multi-choice cloze tests that have been widely used in real practices, such as the GRE verbal test. Moreover, the input sequence and the candidates are given by the meta controller network, which gradually increases the difficulty of the generator's pre-training task. In the beginning, the meta controller is not well-trained, so it provides the generator with easy multi-choice questions. Therefore, the generator can learn from these questions efficiently. As the meta controller outputs more meaningful token alternatives and negative candidates, the generator will be forced to make predictions relying on deep semantic information from contexts. In conclusion, MC-BERT strikes a good balance between training efficiency and the richness of semantic information learned by the model.

Our method is related to curriculum learning [2]. Curriculum learning suggests that some instances are easier to learn, and the model training should first focus on easy instances and then on the hard ones. Our work is different from curriculum learning in that we consider the complexity of the self-supervised tasks rather than the difficulty of instances.

Note that our methodology is quite general. As the main idea is to simplify the generation task using a meta controller, it is easy to be extended to help a broad class of self-supervised pre-training methods, such as XLNet and UniLM [5].

5 Experiments

In this section, we evaluate our proposed MC-BERT with BERT and ELECTRA on a wide range of tasks. We implement all methods based on *fairseq* [7] in PyTorch [3]. For BERT, we use the

[3] Codes have been anonymously released to `https://github.com/MC-BERT/MC-BERT` for review.

Table 2: Hyperparameter search spaces for fine-tuning. Other hyperparameters are kept the same as pre-training.

Batch size	{16, 32}
Maximum epoch	10
Learning rate	{1e-5, ..., 8e-5}
Warm-up ratio	0.06
Weight decay	0.1

implementation of RoBERTa (an optimized version of BERT) [12] in *fairseq*. We will use RoBERTa to refer to as BERT in the following of this section.

5.1 Experimental Setup

Model architecture We use the same architecture for RoBERTa, the discriminator D of ELECTRA, and the generator G_{model} of MC-BERT, where we set all hyperparameters to be the same as BERT-Base (110M parameters). The only difference between these three models lies in the number of output categories of the output layer. Clark et al. [3] recommends using a small-size generator for better efficiency. For a fair comparison, we set the architecture of our meta controller G_{ctrl} to be the same as the ELECTRA generator G.

Pre-training We use the same pre-training corpus as [4], which consists roughly 3400M words from English Wikipedia corpus[4] and BookCorpus[5]. We apply *byte pair encoding* (BPE) [17] with the same vocabulary size as BERT, where $|V| = 32768$.

We construct the inputs of MLM models (RoBERTa, the ELECTRA discriminator, and the meta controller of MC-BERT) in the same way as Devlin et al. [4]. For MC-BERT, we set the number of token candidates $k = 20$ and set the factor of the generator's loss function $\lambda = 20$, unless otherwise specified.

We use the same sequence lengths, batch sizes, and training steps as Devlin et al. [4] for all models. In total, we train each model for 1 million steps. We use the same optimizer configuration as Liu et al. [12] and the same learning rate scheduling scheme as Devlin et al. [4]. We train all models on 8 NVIDIA Tesla V100 GPUs.

Fine-tuning We use the GLUE (**G**eneral **L**anguage **U**nderstanding **E**valuation) benchmark[21] as the downstream tasks to evaluate the performance of the pre-trained models. GLUE consists of nine tasks. CoLA is a syntactic task where the model checks the linguistic acceptability of each sentence. Other tasks, such as SST-2 (sentiment analysis), STS-B (semantic text similarity), and MNLI (natural language inference), are semantic tasks. The detailed description of each task is shown in the supplementary materials.

We run each configuration with ten different random seeds and take the average of these ten scores as the performance of this configuration. We report the best score over all configurations.

5.2 Experiment Results

To compare efficiency fairly, we define a list of computational costs (in terms of FLOPs). For each experiment, we dump the checkpoint trained with respective computational cost and then fine-tune it in downstream tasks. All corresponding results are shown in Table 3.

Syntactic tasks Table 3 shows that our proposed MC-BERT is significantly better than RoBERTa under different computational constraints, which indicates that MC-BERT is much more efficient than RoBERTa. On the other hand, for this particular task, CoLA, ELECTRA outperforms both RoBERTa and MC-BERT, because the pre-training task of ELECTRA is more aligned with CoLA. As we discussed in Section 3 and Section 4, the *replaced token detection* pre-training task of ELECTRA

[4] `https://dumps.wikimedia.org/enwiki`

[5] As the dataset BookCorpus [24] is no longer freely distributed, we follow the suggestions from Devlin et al. [4] to crawl from `smashwords.com` and collect BookCorpus by ourselves.

Table 3: The results on the GLUE benchmark. The percentage numbers of the FLOPs denote the progress of pre-training.

Task	Model	Pre-train FLOPs					
		4%	8%	16%	32%	64%	100%
Syntactic (CoLA)	RoBERTa	27.21	42.23	47.00	50.69	57.40	57.41
	ELECTRA	**44.83**	**58.15**	**61.05**	**61.49**	**65.72**	**64.34**
	MC-BERT	39.20	53.27	57.96	59.20	62.05	62.10
Semantic (8 tasks)	RoBERTa	76.40	80.06	81.83	82.85	84.41	84.65
	ELECTRA	79.57	82.76	84.22	85.23	86.15	86.52
	MC-BERT	**79.78**	**83.23**	**84.28**	**85.46**	**86.63**	**86.82**

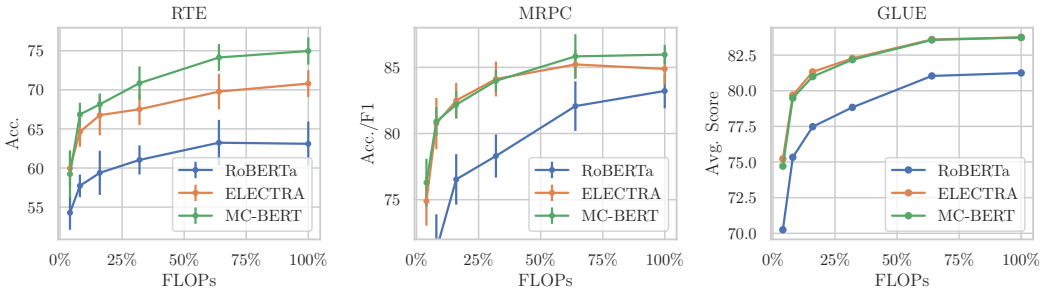

Figure 3: Left: Model performances on RTE; Middle: Model performances on MRPC; Right: GLUE scores.

mainly provides the model with syntactic information, making it do particularly well in syntactic acceptability tasks. Therefore, we would like to focus on the comparison of different models on the other eight tasks, which are semantic tasks that require deeper semantic understanding.

Semantic tasks We report the average performance of each checkpoint on the eight tasks. As shown in Table 3, MC-BERT consistently outperforms RoBERTa and ELECTRA in almost all checkpoints, which indicates that MC-BERT is more efficient than RoBERTa and ELECTRA in learning semantic information from texts. In Figure 3 (Left and Middle), we show the learning curves of two semantic tasks, RTE and MRPC. For both tasks, MC-BERT achieves higher performance than ELECTRA and RoBERTa under the same computational budgets. For tasks that require deeper semantic understanding, our proposed MC-BERT has more significant advantages in terms of efficiency and effectiveness than the baselines do.

Discussion The above experimental results show that MC-BERT outperforms BERT on all the tasks, indicating the effectiveness of using a meta controller to help the generator's training. They also suggest that the generator-discriminator framework in ELECTRA is not the only way to achieve better efficiency.

We plot the final GLUE scores of all model checkpoints in Figure 3 (Right). Our MC-BERT is competitive to ELECTRA in terms of the average performance of the nine tasks. However, MC-BERT does better in eight semantic tasks but performs worse on the one syntactic task.

5.3 Effect of hyper-parameters

We examine the effect of hyper-parameters used in MC-BERT. We follow the experimental settings described above and assess the pre-trained models' performance on the RTE task. The experimental results are shown in Figure 4.

Effect of varying k If k is very large, e.g., $k \approx |V|$, MC-BERT will be comparable to ELECTRA-complex (see Section 3), so degraded performance is expected. In Figure 4, we compare the performance given reasonable smaller values of k, specifically $k = 10$ and $k = 100$. The models

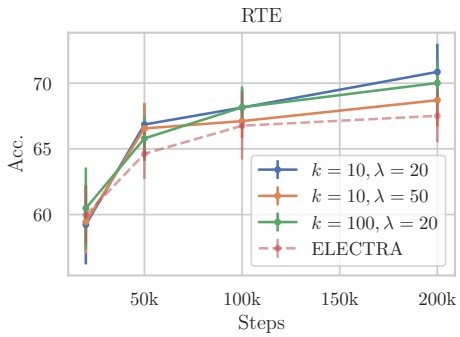

Figure 4: Effect of hyper-parameters in MC-BERT.

trained with $k = 10$ perform slightly better than those trained with $k = 100$, but the difference is insignificant.

Effect of varying λ Since λ serves as a trade-off between learning the meta controller and the generator, a larger λ indicates a greater focus on optimizing the generator rather than optimizing the meta controller. To check the effects of varying λ, Figure 4 compares the models trained with $\lambda = 20$ and $\lambda = 50$. We can see that the models trained with $\lambda = 20$ is consistently better than the model trained with $\lambda = 50$, which implies that too large λ may hurt the performance due to the less optimized meta controller.

6 Conclusion and Future Work

In this work, we propose MC-BERT, which uses a meta controller to manage the complexity of the pre-training task. The pre-training task is a multi-choice cloze test with a reject option, "None of the above". Extensive experiments demonstrate MC-BERT is more efficient than BERT, and it learns deeper semantic information than ELECTRA does. It outperforms several baselines on semantic understanding tasks given the same computational budget. We will continue exploring more roles of the meta controller, e.g., how to mask positions and select batched sentences smartly.

References

[1] Attila Ambrus, Ben Greiner, and Anne Sastro. The case for nil votes: Voter behavior under asymmetric information in compulsory and voluntary voting systems. *Journal of Public Economics*, 154:34–48, 2017.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[3] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2019.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13042–13054, 2019.

[6] Timothy J Feddersen and Wolfgang Pesendorfer. Abstention in elections with asymmetric information and diverse preferences. *American Political Science Review*, 93(2):381–398, 1999.

[7] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional Sequence to Sequence Learning. In *Proc. of International Conference on Machine Learning*, 2017.

[8] Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of bert by progressively stacking. In *International Conference on Machine Learning*, pages 2337–2346, 2019.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL*, 2007.

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

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

[13] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[14] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*, 2018.

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

[16] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv:1910.10683 [cs, stat]*, Oct 2019. URL `http://arxiv.org/abs/1910.10683`. arXiv: 1910.10683.

[17] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL `http://arxiv.org/abs/1508.07909`.

[18] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[19] Wilson L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953. doi: 10.1177/107769905303000401. URL `https://doi.org/10.1177/107769905303000401`.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[21] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018. URL `http://arxiv.org/abs/1804.07461`.

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

[23] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 1(5), 2019.

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

We released the source code of MC-BERT at: `https://github.com/MC-BERT/MC-BERT`. Our code is based on PyTorch, Fairseq[6], and RoBERTa [12].

We run all our experiments on NVIDIA Tesla V100. For pre-training, we use mixed-precision floating-point arithmetic (FP16+FP32) to accelerate training; for fine-tuning, we train with single-precision floating-point arithmetic (FP32). Besides, we measure the FLOPs in the same way as Clark et al. [3] do.

A Model Details

The architecture settings for RoBERTa, ELECTRA and MC-BERT are listed in Table 4. For the encoder served for downstream tasks, we use the same architecture for three models. As for the generator of ELECTRA, we use the same model size as Clark et al. [3]. We also set the size of the meta controller of MC-BERT to be the same as the size of the ELECTRA generator.

Table 4: Model specifications. The "Encoder" denotes the transformer served for downstream tasks in each model, with the same base architecture for all the models. The "G_{ctrl} / G" denotes the controller of MC-BERT and the generator of ELECTRA, respectively, and they are also set to be the same.

Hyperparameter	**Encoder**	$G_{\textbf{ctrl}}$ **/** G
Number of layers	12	12
Hidden size	768	256
FFN inner hidden size	3072	1024
Attention heads	12	4
Attention head size	64	64
Embedding size	768	768

B Pre-Training Details

B.1 Dataset

We use the same dataset as the one in BERT [4], which includes BooksCorpus and Wikipedia. After concatenating these two datasets, we obtain a corpus with roughly 3400M words in total. Following the practices of Devlin et al. [4], we first segment documents into sentences with Spacy[7]; then, we normalize, lower-case, and tokenize texts using Moses decoder [10]; next, we apply *byte pair encoding* (BPE) [17]. We randomly split documents into one training set and one validation set, where the training-validation ratio for pre-training is 199:1. The vocabulary consists of 32,768 tokens. Following Liu et al. [12], we pack each input with full sentences sampled contiguously from the corpus, such that the total length is at most 512 tokens.

B.2 Hyperparameters

The pre-training hyperparameters are set mostly the same as the ones in BERT [4]. However, as are suggested by recent works [22] [12] [11], we remove the next sentence prediction (NSP) pre-training task. The details are listed in Table 5.

C Down-Stream Details

C.1 GLUE Tasks

We use the GLUE (**G**eneral **L**anguage **U**nderstanding **E**valuation) dataset [21] as the downstream tasks to evaluate the performance of the pre-trained models. Particularly, there are nine tasks within the GLUE dataset that have been widely used for evaluation, including CoLA, RTE, MRPC, STS-B, SST-2, QNLI, QQP, and MNLI-m/mm. The specifications of these tasks are listed in Table 6. Especially, we follow BERT [4] and ELECTRA [3] to skip WNLI in our experiments, because few submissions on the leaderboard[8] do better than predicting the majority class for this task.

Notably, we strictly adopt official metrics to evaluate the performance on GLUE tasks. However, the scores reported in ELECTRA [3] are not. Their evaluation metrics are Spearman correlation for STS-B (instead of the

[6] `https://github.com/pytorch/fairseq`

[7] `https://spacy.io`

[8] `https://gluebenchmark.com/leaderboard`

Table 5: Pre-training hyperparameter settings.

Hyperparameter	Pre-training Value
Learning rate	1e-4
Learning rate decay	Linear
Decay steps	1,000,000
Warmup steps	10,000
Adam ϵ	1e-6
Adam $(\beta 1, \beta 2)$	(0.9, 0.98)
Batch size	256
Dropout	0.1
Attention dropout	0.1
Weight decay	0.01
ELECTRA λ	50
MC-BERT λ	20
MC-BERT k	10

average of Spearman correlation and Pearson correlation), Matthews correlation for CoLA, and accuracy for all the other GLUE tasks (instead of the average of F1-score and accuracy for MRPC and QQP).

Table 6: Specification of GLUE tasks.

Corpus	Size	Task	#Class	Metric(s)	Domain
Syntactic Tasks					
CoLA	8.5k	Acceptibility	2	Matthews correlation	Misc.
Semantic Tasks					
RTE	2.5k	Inference	2	Accuracy	Misc.
MRPC	3.7k	Paraphrase	2	Accuracy/F1	News
STS-B	5.7k	Similarity	-	Pearson/Spearman corr.	Misc.
SST-2	67k	Sentiment	2	Accuracy	Movie reviews
QNLI	108k	QA/Inference	2	Accuracy	Wikipedia
QQP	364k	Similarity	2	Accuracy/F1	Social QA questions
MNLI-m/mm	393k	Inference	3	Accuracy	Misc.

C.2 Fine-Tuning Details

For fine-tuning, most hyperparameters are also the same as in BERT [4]. We design an exhaustive search for batch size, learning rate to get reasonable performance numbers. The details of the search space has been shown in the paper. Our search space is much larger than the setting in both BERT [4] and ELECTRA [3], with higher confidence. Except for the hyperparameters listed in space, the other parameters are all set the same as in pre-training.

D Detailed Results

Due to the space limitation, the detailed experimental results are listed here in Table 7. The scores on all the tasks are listed, including the scores on the nine tasks for each checkpoint. The “Avg." is for the semantic tasks. The number below each task denotes the number of training examples. The metrics for these tasks are mentioned above. Following the standard practice for computing GLUE scores, we report the arithmetic average of all metrics for tasks with multiple metrics (MRPC, QQP, STS-B), and we average the score of MNLI-m and MNLI-mm to get the final score of MNLI.

It can be seen from the table, when the data size of the downstream task is large, e.g., in QNLI, MNLI and QQP, the performance of both RoBERTa/ELECTRA and our proposed method are similar. However, when the data size of the downstream task is small, ELECTRA and ours are significantly better than RoBERTa.

Table 7: The detailed results on the GLUE benchmark (except WNLI).

FLOPs	Model	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 5.7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	Avg. -
4% (2e18)	RoBERTa	27.21	87.58	63.04	80.62	86.76	76.93/77.69	85.22	54.28	76.40
	ELECTRA	44.83	87.72	74.89	83.50	87.38	77.48/78.08	85.75	59.96	79.57
	MC-BERT	39.20	88.50	76.29	83.54	87.31	77.68/78.27	85.63	59.23	79.89
8% (4e18)	RoBERTa	42.23	90.70	71.22	85.08	88.15	79.71/79.78	87.82	57.72	80.06
	ELECTRA	58.15	90.15	80.75	86.04	88.64	80.63/80.93	88.32	64.62	82.76
	MC-BERT	53.28	91.11	80.91	86.11	88.51	80.80/80.95	88.23	66.85	83.23
16% (8e18)	RoBERTa	47.00	91.56	76.53	86.22	88.68	81.36/81.55	88.98	59.38	81.83
	ELECTRA	61.05	91.56	82.50	87.37	89.14	82.32/82.28	89.90	66.76	84.22
	MC-BERT	57.96	91.96	82.18	86.93	89.11	82.04/82.30	89.46	68.14	84.28
32% (1.6e19)	RoBERTa	50.69	92.22	78.30	86.72	89.13	82.61/82.41	90.05	61.03	82.85
	ELECTRA	61.49	92.31	84.12	88.46	89.57	83.85/83.87	90.80	67.51	85.23
	MC-BERT	59.20	92.67	83.98	87.31	89.37	83.69/83.58	90.37	70.85	85.46
64% (3.2e19)	RoBERTa	57.40	93.08	82.07	87.72	89.31	84.40/84.47	91.04	63.24	84.41
	ELECTRA	65.72	92.82	85.22	88.79	89.99	85.42/84.80	91.31	69.77	86.15
	MC-BERT	62.05	92.41	85.83	88.23	89.75	85.11/84.73	91.15	74.13	86.63
100% (5e19)	RoBERTa	57.41	93.15	83.22	88.14	89.24	84.69/84.63	91.02	63.10	84.65
	ELECTRA	64.33	93.38	84.88	89.10	89.96	86.00/85.29	91.85	70.80	86.52
	MC-BERT	62.10	92.34	85.96	88.01	89.65	85.68/85.24	91.34	74.96	86.82