Understand and Modularize Generator Optimization in ELECTRA-style Pretraining

Chengyu Dong [1 2] **Liyuan Liu** [3] **Hao Cheng** [3] **Jingbo Shang** [1] **Jianfeng Gao** [3] **Xiaodong Liu** [3]

Abstract

Despite the effectiveness of ELECTRA-style pre-training, their performance is dependent on the careful selection of the model size for the auxiliary generator, leading to high trial-and-error costs. In this paper, we present the first systematic study of this problem. Our theoretical investigation highlights the importance of controlling the generator capacity in ELECTRA-style training. Meanwhile, we found it is *not* handled properly in the original ELECTRA design, leading to the sensitivity issue. Specifically, since adaptive optimizers like Adam will cripple the weighing of individual losses in the joint optimization, the original design fails to control the generator training effectively. To regain control over the generator, we modularize the generator optimization by decoupling the generator optimizer and discriminator optimizer completely, instead of simply relying on the weighted objective combination. Our simple technique reduced the sensitivity of ELECTRA training significantly and obtains considerable performance gain compared to the original design.

1. Introduction

ELECTRA-style pre-training, as introduced in Clark et al. (2020), has demonstrated significant potential in enhancing the effectiveness and efficiency of training LLMs. In specific, it trains the discriminator model (the main model that is used in downstream tasks) to detect which tokens in an input sequence were replaced by the generator model (the auxiliary model that is not used in downstream tasks). This approach has become increasingly popular among various pre-training settings and downstream applications (Clark et al., 2020; Chi et al., 2021; Kanakarajan et al., 2021; Meng et al., 2021; 2022; Bajaj et al., 2022).

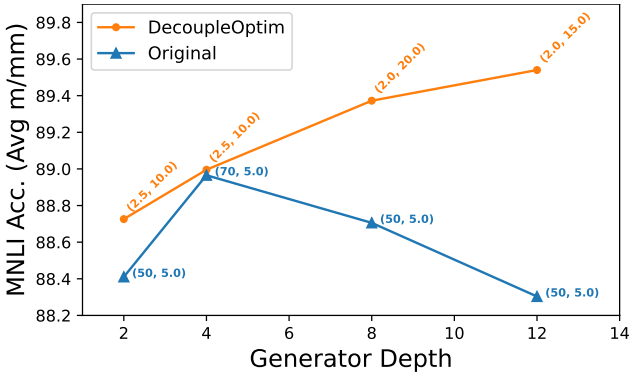

Figure 1. Downstream task performance (MNLI accuracy, Avg m/mm) for models trained with different generator sizes under the "Original" design and our "DecoupledOptim" technique. Each point is annotated by the best hyperparameter setting we found for this experiment, namely (loss weight, learning rate) and (generator learning rate, discriminator learning rate) for "Original" and "DecoupledOptim" respectively (learning rate is scaled by 10^4 for simplicity). More experiment details can be found in Section 6.1.

Despite its effectiveness, the performance of ELECTRA is sensitive to the choice of generator size (Clark et al., 2020). As depicted in Figure 1, variations in generator size can lead to a significant decline in the performance upon fine-tuning the discriminator on downstream tasks [1]. Such sensitivity inevitably demands careful selection of the generator size in real-world practices, which can be time-consuming and resource-intensive. In this research, we investigate the issue of performance degradation in a systematic manner.

First, by carefully evaluating the discriminator's capability of detecting replaced tokens, we confirm that large generator capacity can indeed hurt the effectiveness of pre-training. We also clarify that the performance degradation occurs during the pre-training stage, instead of during the fine-tuning stage. Upon further examination of the optimization in the

[1]University of California, San Diego [2]Work was done during an internship at Microsoft. [3]Microsoft Research. Correspondence to: Chengyu Dong <cdong@ucsd.edu>, Xiaodong Liu <xiaodl@microsoft.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]Following previous works (He et al., 2021; Bajaj et al., 2022), we use the evaluation results on MNLI (Williams et al., 2017) to indicate the performance on downstream tasks

original ELECTRA design, we recognize that it may fail to control the generator capacity effectively in the course of pre-training. The original ELECTRA relies on a weight ratio that combines the training objectives of the generator and the discriminator, in the expectation of balancing their optimization. However, this method is observed to be largely ineffective since a constant scaling of the loss will not affect adaptive optimizers like Adam (Kingma & Ba, 2015), the de facto for LLM pre-training (Liu et al., 2020a). Such a deficiency results in the sensitivity of the original ELECTRA to the generator size.

To regain control over the generator training, we modularize the generator optimization by disentangling the generator optimizer from the discriminator optimizer. This simple technique, dubbed as *DecoupledOptim*, effectively mitigate the sensitivity of ELECTRA-style pre-training to the generator size and regain the performance loss caused by a large generator. Furthermore, our algorithm can foster the flexibility of accelerating discriminator optimization without being impeded by the instability of generator training, thus bringing significant performance gain over strong baselines. We conduct experiments with the standard $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ (Devlin et al., 2019) pre-training setting on the GLUE (Wang et al., 2018) benchmark, and our simple technique consistently outperforms the original ELECTRA design and alternative pre-training specifications that are more recently proposed.

Motivated by the empirical evidence, we turn to explore the underlying mechanism of how the generator and discriminator optimizations impact the ELECTRA-style pre-training. Our theoretical analyses reveal that a well-performed generator will indeed impair discriminator learning, which highlights the importance of controlling generator learning. Our analyses also corroborate the necessity of accelerating discriminator optimization in order to excel in pre-training performance.

To summarize, our main contributions are as follows.

- Our analysis identifies a deficiency in the original ELECTRA optimization that leads to performance degradation during the pre-training stage (Section 3).
- Guided by our analyses, we introduce a simple yet effective method (Section 4) that greatly improves training robustness and downstream performance (Section 6).
- We conduct theoretical analyses to further gain insights into how generator and discriminator optimizations impact the ELECTRA performance (Section 5).

Our source code is publicly available [2].

[2] https://github.com/namisan/DecoupledOptim

2. Background and Related Work

Masked Language Modeling (MLM). MLM methods such as BERT pretrain the language model to predict randomly masked tokens in a sequence. Specifically, given an input sequence $\boldsymbol{x} = [w_1, w_2, \cdots, w_n]$, MLM generates a *masked sequence* $\tilde{\boldsymbol{x}} = [w_1, \cdots, \texttt{[mask]}, \cdots, w_n]$ by randomly selecting a few tokens at positions $\mathcal{M} = [i_1, i_2, \cdots, i_m]$ and replace them with `[mask]` token. The model is then trained to predict the original tokens given the masked sequence $\tilde{\boldsymbol{x}}$. The training objective can be formulated as

$$L(\theta) = \mathbb{E}_{\boldsymbol{x}} \sum_{i \in \mathcal{M}} - \log p_\theta(w_i | \tilde{\boldsymbol{x}})_i,$$

where θ denotes the model parameters and $p_\theta(w_i|\tilde{\boldsymbol{x}})_i$ is the predictive probability of the model at the i-th position on token w_i given the masked sequence $\tilde{x}$.

ELECTRA-style pretraining. Unlike MLM, ELECTRA constructs a pretraining task called Replaced Token Detection (RTD) [3], which involves the joint training of two deep neural models, a generator G (auxiliary model) and a discriminator D (main model). Here the generator is pretrained with MLM as usual, while the discriminator is pretrained to detect tokens in a sequence that are replaced by a generator.

Specifically, given a masked sequence $\tilde{\boldsymbol{x}}$ constructed for MLM, a *corrupted sequence* $\hat{\boldsymbol{x}}$ is generated by replacing each `[mask]` token in $\tilde{\boldsymbol{x}}$ by a token that is sampled from the generator's predictive distribution at that `[mask]` token, namely $\hat{\boldsymbol{x}} = [w_1, \cdots, \hat{w}_i, \cdots, w_n]$ and $\hat{w}_i \sim p_G(\cdot|\tilde{\boldsymbol{x}})_i$. We refer to those sampled tokens as replaced tokens since they will be different from the original tokens at corresponding positions, as long as the generator does not predict the masked tokens correctly with a one-hot probability distribution. The discriminator is then trained to predict whether the replaced tokens in $\hat{\boldsymbol{x}}$ match the original tokens. The training objective can thus be defined as

$$L_G(\theta_D) = \mathbb{E}_{\boldsymbol{x}} \sum_{i \in \mathcal{M}} \mathbb{E}_{\hat{w}_i \sim p_G} \ell(D(\hat{\boldsymbol{x}})_i, \mathbf{1}_{\hat{w}_i = w_i}) + \mathbb{E}_{\boldsymbol{x}} \sum_{i \in [n] \setminus \mathcal{M}} \ell(D(\hat{\boldsymbol{x}})_i, 1),$$

where $D(\hat{\boldsymbol{x}})_j$ is a scalar score output by the discriminator quantifying the probability of the j-th token being replaced, ℓ is a loss function, typically binary cross-entropy (BCE), and $\mathbf{1}_{\hat{w}=w}$ is the indicator function, namely

$$\mathbf{1}_{\hat{w}=w} = \begin{cases} 1, & \text{if } \hat{w} = w, \\ 0, & \text{if } \hat{w} \neq w. \end{cases}$$

Note that in ELECTRA, the training objective of the discriminator is defined over all input tokens rather than the

[3] We will use ELECTRA-style pretraining and RTD pretraining interchangeably in this paper.

randomly masked subset such as that in MLM.

Analyses of ELECTRA-style pretraining. Extensive analytical efforts have been attracted to understanding the effectiveness of ELECTRA-style pretraining. It was originally believed that the RTD pretraining task gains mostly because of the improved sample efficiency by posing the objective on all tokens, as well as an alleviated pretraining fine-tuning gap (Clark et al., 2020). Recent works also empirically demonstrate that ELECTRA-style pretraining may be advantageous because of the low task complexity of RTD compared to MLM (Xu et al., 2020) or implicit learning curriculum introduced by the generator (Meng et al., 2022),

Variations of ELECTRA-style pretraining. There exist various pretraining methods built on top of ELECTRA. Xu et al. (2020) proposes a pretraining variation alike a multi-choice cloze test, where the main model predicts the original token from a small candidate set. Meng et al. (2021) introduces two additional training objectives including recovery of the original token and alignment between corrupted sequences from the same source. Hao et al. (2021) estimates the discriminator loss on replaced tokens and learns to sample difficult replace tokens from the generator. Meng et al. (2022) proceeds to automatically construct a difficult learning signal by an adversarial mixture of multiple generators. He et al. (2021) argues that embedding sharing may hurt in ELECTRA-style pretraining and suggests preventing the discriminator gradients from back-propagating to the generator embeddings. Bajaj et al. (2022) conducts a comprehensive ablation study and highlights several important improvements of ELECTRA such as large vocabulary size and relative position embedding. Zhang et al. (2022) observes the existence of "false negative" replaced tokens, namely those that are not exactly same but are synonyms to the original ones, and proposes to correct them by synonym look-up and token similarity regularization.

3. Impact of Generator Capacity

In ELECTRA-style pretraining, it has been widely observed that the optimal discriminator performance can only be obtained by a generator that is neither too large nor too small. As shown in Figure 1 for "Baseline", generators with more than 4 layers consistently hurt the discriminator performance on downstream tasks. Here, we conduct systematic analyses to explore the mechanism of this phenomenon.

3.1. Large Generator Capacities Slow Pre-training

Since the performance of the pre-trained model is evaluated in a two-stage setting (i.e., pre-training and fine-tuning), we first aim to understand whether the performance degradation happens in the pre-training stage (i.e., the discriminator is not trained properly) or the fine-tuning stage (i.e., the discriminator is not fine-tuned properly).

Our exploration suggests that performance degradation has already occurred during the pre-training stage. Specifically, we compare the pre-training (RTD) performance of discriminators trained with generators of different depths (4, 8, and 12 layers). As shown in Figure 2, discriminators trained with deeper generators achieve consistently worse RTD performance, echoing their inferior performance on downstream tasks as shown in Figure 1.

Note that, we used the last checkpoint of the 12-layer generator for the evaluation in Figure 2. Rather surprisingly, we observe that to achieve better RTD performance against a deep generator, training the discriminator on a shallow generator can be more effective than training the discriminator on that deep generator itself. As shown in Figure 2, the discriminator trained with the 12-layer generator performed the worst on replaced tokens sampled from this very same generator, compared to other discriminators trained with either the 4-layer or the 8-layer generator. This observation implies that the discriminator trained with a deep generator are not fully optimized in terms of their pre-training objectives and that the performance degradation may be due to a slow convergence in the course of pre-training.

It is worth mentioning that, since the relationship between the generator capacity and the performance degradation resembles overfitting, it may seem reasonable to speculate overfitting plays an important role in this phenomenon. However, in our analyses, we observe the impact of the overfitting to be marginal. We summarized our analyses on the impact of overfitting in Appendix D.

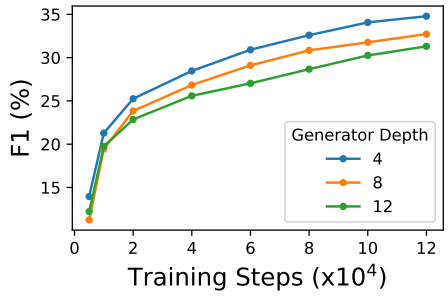

Figure 2. RTD performance (F1-score) of discriminators trained with generators of different depths (4, 8, 12 layers). We measure the RTD performance on replaced tokens generated by the same generator, which is the last checkpoint of the 12-layer generator used to train one of the discriminators.

3.2. Limitations of the Original ELECTRA in Generator Capacity Control

We can see that controlling the generator capacity is critical to the optimization of ELECTRA. Nevertheless, we found

that the original design of ELECTRA may be deficient in controlling the generator capacity.

In ELECTRA, a loss weight λ is originally introduced to balance the generator optimization and discriminator optimization. In specific, the generator and the discriminator are jointly optimized through the following combined training loss (Clark et al., 2020)

$$L = L(\theta_G) + \lambda L_G(\theta_D).$$

Nevertheless, varying the value of λ may not take effect as expected. First, a constant scaling of the loss will not affect adaptive optimizers like Adam (Kingma & Ba, 2015), which is commonly used in pretraining algorithms such as ELECTRA to ensure training stability. In specific, Adam updates a model parameter by the ratio between the first moment and second moment of its gradient, namely [4]

$$\theta := \theta - \eta \cdot \frac{\mathbb{E}[g(\theta)]}{\sqrt{\mathbb{E}[g(\theta)^2]}}, \tag{1}$$

where η is the learning rate, and $g(\theta) = \nabla_\theta L$ is the gradient of the model parameter θ with respect to L.

Consequently, for all generator parameters that are not shared with the discriminator (denoted as $\hat{\theta}_G$) and all discriminator parameters that are not shared with the generator ($\hat{\theta}_D$), we have [5]

$$g(\hat{\theta}_G) = \nabla L(\theta_G), \quad g(\hat{\theta}_D) = \lambda \cdot \nabla L_G(\theta_D).$$

It is important to note that the loss weight λ does not affect the update rule of these parameters as any constant scaling of the gradients will be canceled out in Equation (1). Therefore, these parameters would always be trained with the same learning rate regardless of the value of λ.

The only parameters in ELECTRA that are affected by the loss weight λ are the embeddings θ_E shared between the generator and the discriminator. The gradients would be

$$g(\theta_E) = \nabla L(\theta_G) + \lambda \cdot \nabla L_G(\theta_D),$$

which means the update rule would become

$$\theta_E := \theta_E - \eta \cdot \frac{\mathbb{E}[\nabla L(\theta_G) + \lambda \cdot \nabla L_G(\theta_D)]}{\sqrt{\mathbb{E}[(\nabla L(\theta_G) + \lambda \cdot \nabla L_G(\theta_D))^2]}}.$$

Therefore, the updates of these embeddings will be contributed by the gradients from the discriminator more if λ is larger.

Since the loss weight λ fails to balance the updates of the majority of model parameters, it cannot control the generator learning effectively. As shown in Figure 3, increasing the loss weight λ has little effect on the generator performance in the original ELECTRA design.

Alternative Ways to Control Generator in the original ELECTRA design. Here we discuss methods other than the loss weight λ to control the generator capacity. One way is to change the learning rate η. However, this would also alter the learning rate for discriminator learning and ultimately results in worse pretraining performance.

Another way is to reduce the model size of the generator, as also shown in Figure 3. This may be the only effective mechanism in the original ELECTRA design that can control generator capacity without affecting discriminator learning. However, it brings about the dependency of the pretraining performance on the careful selection of the generator size, which can be time-consuming and resource-intensive in practice.

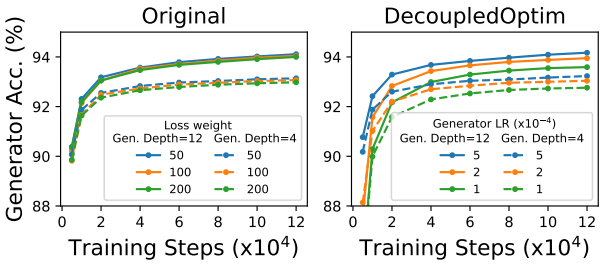

Figure 3. (Left): The effect of tuning the loss weight λ on the generator performance in the original ELECTRA design. (Right): The effect of tuning the generator learning rate η_G on the generator performance with our DecoupledOptim technique. For the original ELECTRA design, the learning rate is fixed as 5×10^{-4} and for our DecoupledOptim technique, the discriminator learning rate is fixed as 1×10^{-3}.

4. *DecoupledOptim*

Decouple the generator optimizer and the discriminator optimizer. To properly control the generator capacity, we simply decouple the generator optimizer and the discriminator optimizer. Specifically, the generator and discriminator parameters are now updated with separate rules, namely

$$\theta_G := \theta_G - \eta_G \cdot \frac{\mathbb{E}[g(\theta_G)]}{\sqrt{\mathbb{E}[g(\theta_G)^2]}},$$

$$\theta_D := \theta_D - \eta_D \cdot \frac{\mathbb{E}[g(\theta_D)]}{\sqrt{\mathbb{E}[g(\theta_D)^2]}}.$$

To control the generator capacity, we can now directly adjust the optimizer dedicated for the generator (*e.g.*, adjusting η_G, the generator learning rate). Figure 3 shows that, for a large

[4] In practice, the first and second moments here are estimated as exponential moving averages.

[5] We neglect the subscript in ∇ for simplicity.

generator, we can simply reduce η_G to effectively control its capacity during the pre-training.

This implies that DecoupledOptim is capable to handle large generators and reduce the sensitivity of ELECTRA-style pretraining on the choice of the generator size. As elaborated in Section 6, our proposed method is simple yet effective, consistently outperforming the original ELECTRA design and its recently proposed variants.

Note that for the simplicity of the implementation, we will not share the embeddings between the generator and the discriminator anymore. Despite the belief that embedding sharing is crucial in ELECTRA-style pretraining since RTD pretraining may not be as effective as MLM in learning token representations (He et al., 2021), we found that with our decoupled optimization, a discriminator learned from randomly initialized embeddings can in fact achieve equivalently good or even better performance.

Improve ELECTRA with Sufficient Discriminator Optimization. With our decoupled-optimizer design, we can in fact not only control generator capacity more easily but also achieve significantly better pretraining performance by optimizing the discriminator more sufficiently.

In the original ELECTRA design, the same learning rate is assigned to the generator and discriminator as mentioned in Section 3.2. Therefore, attempts to speed up discriminator optimization by increasing the learning rate inevitably lead to larger generator capacity, thus yielding worse pretraining performance. Moreover, The increased learning rate may even cause training failure since the generator training in an MLM style can undergo strong instability with a large learning rate. In our experiments, we found that the original ELECTRA design diverges within 25K training steps, despite a proper selection of the loss weight (*e.g.*, 50) and the generator size (*e.g.*, 4 layers).

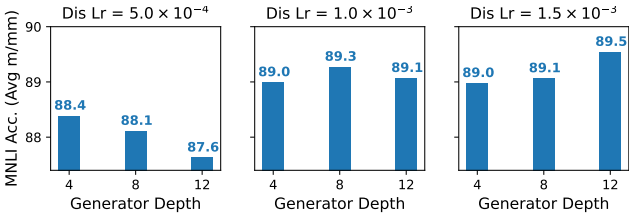

Figure 4. Discriminator performance on downstream tasks (MNLI) with different generator sizes and discriminator learning rates. Here the generator learning rate is fixed as 2×10^{-4}.

However, with the optimizers decoupled in DecoupledOptim, we can now accelerate discriminator optimization without being impeded by generator learning. Empirical experiments in Figure 4 show that, as we increase the discriminator learning rate such that its optimization becomes more sufficient, the pretraining performance is increasingly good. One may notice that the discriminator learning rate can often be as large as 1.5×10^{-3}, which is 3-7 times the learning rate suitable for generator training. Furthermore, we observe that with increasingly sufficient discriminator optimization, the best generator shifts to one with a larger capacity, even as large as the discriminator itself (12 layers).

5. Optimization of ELECTRA-style Methods

Here, we conduct theoretical analyses to gain insight into how generator and discriminator optimizations impact the performance of the ELECTRA-style pretraining.

Problem setup. We consider a simplified RTD task where only one token in an input sequence is replaced. We refer to the rest of those unchanged tokens in this sequence as context. Let w be a word in the sentence, and let c be the remaining context words in the same sentence. The generator is trained to predict the original token given the context, namely

$$L(\theta_G) = \mathbb{E}_{c,w} - \log p_G(w|c). \tag{2}$$

For discriminator training, we focus on the detection of this single replaced token exclusively. The optimization objective of the discriminator D can be thus described as

$$\bar{L}_G(\theta_D) = \mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim p_G}\ell(D(c,\hat{w}), \mathbf{1}_{\hat{w}=w}). \tag{3}$$

Ideal discriminator optimization objective. An ideal discriminator optimization objective should align with the discriminator performance on downstream tasks. However, discriminator performance evaluated against a given replaced token distribution may not always be indicative of the downstream performance (see more in Section D). Ideally, a discriminator should be able to detect any possible tokens replaced in a sequence, regardless of the specific distribution from which such replaced tokens are sampled. To this end, we define the ideal optimization objective of the discriminator as the highest possible discriminator loss achieved by any replaced token distribution, namely the probability distributions from which the replaced tokens are sampled.

Definition 1 (Ideal optimization objective of the discriminator)**.** *Let $\mathcal{P}$ be a family of replaced token distributions. The ideal optimization objective of the discriminator D can be defined as*

$$L^*(\theta_D) = \mathbb{E}_{c,w} \sup_{p\in\mathcal{P}} \mathbb{E}_{\hat{w}\sim p}\ell(D(c,\hat{w}), \mathbf{1}_{\hat{w}=w}). \tag{4}$$

Practical optimization of the discriminator. In practice, such an ideal optimization objective cannot be used as a loss function for training the discriminator since it is not feasible to enumerate all possible replaced token distributions.

Fortunately, if we have an assumption on the "difficulty" of the replaced token distributions, we can bound the highest discriminator loss over all possible replaced token distributions. This upper bound can further be approached by the discriminator loss on the generator's predictive distribution as a replaced token distribution. Hence, the optimization objective defined by a generator (*i.e.*, Equation (3)) can be a fair surrogate of the ideal objective.

Assumption 1 (Difficulty of the replaced token distribution)**.** *We assume a discriminator is more likely to make detection errors if the sampled replaced tokens recover the original token more frequently, namely*

$$\mathbb{E}_{\hat{w}\sim p}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}) = F_D(\mathbb{E}_{\hat{w}\sim p}[\mathbf{1}_{\hat{w}=w}]) \quad (5)$$

where $F_D : [0,1] \to \mathbb{R}^+$ is a concave and monotonically increasing function that is dependent on the discriminator D.

We can now show that the ideal discriminator objective is bounded by the surrogate objective defined by a generator. The intuition here is that, given Assumption 1, the predictive distribution of the generator should approximate the most difficult replaced token distribution, and the approximate error happens to be bounded by the performance of the generator.

Lemma 1 (The discriminator objective defined by a generator is a surrogate of the ideal objective)**.** *Let $m_D = \max_{c,w} \|\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w})\|_2$ be the upper bound on the discriminator loss given any context-token pairs. Then we have*

$$L^*(\theta_D) \le \bar{L}_G(\theta_D) + 2^{-1/2} m_D L(\theta_G)^{1/2}. \quad (6)$$

Lemma 1 implies that training towards a surrogate objective defined by a generator $L_G(\theta_D)$ can indeed optimize the ideal discriminator objective. This justifies the basic ELECTRA design which employs a generator to sample replaced tokens for discriminator training. It also implies that an under-performed generator may not be as effective for optimizing the ideal objective since the distribution approximation error (the 2-nd term) would be much higher.

Large generator capacity may hurt optimization. However, well-performed generators may be less effective for optimizing the ideal objective as well. This is because well-performed generators will approach the most difficult replace token distribution based on Assumption 1, thus creating significantly higher discriminator loss $L_G(\theta_D)$ in Equation (6). We have the following Lemma to demonstrate this.

Lemma 2 (Dependence of the discriminator loss on generator performance)**.** *Let $V_G = \mathbb{E}_{c,w}[(-\log p_G(w|c) - L(\theta_G))^2]$ be the variance of the generator loss, we have*

$$\bar{L}_G(\theta_D) \le F_D\left((1+V_G/2)\, e^{-\bar{L}(\theta_G)} + V_G/2\, e^{\varepsilon}\right). \quad (7)$$

Lemma 2 shows that the discriminator loss given a generator is inversely correlated with the generator loss. This means strong generators may create significantly higher discriminator loss. If such a high discriminator loss cannot be sufficiently reduced through the optimization process, which is likely since the training budget is always limited, the ideal objective cannot be sufficiently optimized as well.

Modeling the generator and discriminator optimization. To further illustrate the effects of both the generator and discriminator optimizations, we introduce a simplified modeling of the optimization process, which is based on trajectory analysis of gradient descent for deep linear neural networks (Arora et al., 2018).

Proposition 1 (Gradient descent trajectory of deep linear neural networks (informal))**.** *In gradient descent, let $\theta(t)$ be the model parameters after t updates, η be the learning rate that meets certain regularities, and N be the number of layers in the model, then we have*

$$\mathcal{L}(\theta(t)) \le \mathcal{L}(\theta(0)) \cdot (1 - \eta \cdot c^{\frac{2(N-1)}{N}})^t, \quad (8)$$

where c is a positive constant.

We can now derive a complete picture of the effect of the generator performance on the optimization of the discriminator. Together with Lemmas 1 and 2, we have the following result.

Theorem 1 (Optimization of the ideal discriminator objective)**.** *Consider the discriminator optimization after the generator is trained with several updates. Let η_G be the generator's learning rate and N_G be the generator depth. The generator loss after t_G updates is*

$$L(\theta_G(t_G)) = L(\theta_G(0))\left(1 - \eta_G \cdot \xi_G\right)^{t_G}, \quad (9)$$

where $\xi_G = c^{2(N_G-1)/N_G}$. Subsequently, let η_D be the discriminator's learning rate and N_D be the discriminator depth. Then after t_D discriminator updates, we have

$$\begin{aligned} L^*(\theta_D(t_D)) \le & F\left(e^{-L(\theta_G(t_G))}\right) \cdot (1 - \eta_D \cdot \xi_D)^{t_D} \\ & + 2^{-1/2} m_D \left[L(\theta_G(t_G))\right]^{1/2}, \end{aligned} \quad (10)$$

where we have neglected some constants in Equation (7) for simplicity.

One can see that, in terms of generator optimization, increasing the generator learning rate first helps and then hurts the discriminator performance for a given discriminator setting, as also illustrated in Figure 5. This results in a sweet spot where the generator learning rate is the best for the discriminator performance. Similarly, since $\xi \propto N$, increasing the generator depth would also first help and then hurt the discriminator performance.

In terms of discriminator optimization, increasing the discriminator learning rate or depth can almost always help the discriminator performance. In an ideal case, if the discriminator optimization is sufficient, for example, by letting $\eta_D \cdot \xi_D \approx 1$ or $t_D \approx \infty$ in Equation (10), then the discriminator performance will improve monotonically with the generator performance, as also illustrated in Figure 5 where the discriminator learning rate is sufficiently large. Unfortunately, it is not possible to always sufficiently optimized the discriminator in practice given training instability and/or limited training budgets.

Interestingly, Equation (10) shows that, with increasingly sufficient discriminator optimization, the best generator should shift to one with a larger capacity, as also illustrated in Figure 5. One may find this echoes the empirical observation in Figure 4.

Finally, Equation (10) also reflects the limitation of the original ELECTRA design mentioned in Section 3.2. As also illustrated in Figure 5, since the generator and discriminator are assigned almost the same learning rate, the original design can only reach a line (1-D subspace) in the entire optimization space. Further, due to the training instability of the generator, this line is truncated where the discriminator learning rate is still small and thus the discriminator performance is still suboptimal. In contrast, DecoupledOptim can increase the discriminator learning rate without being affected by the generator learning thus achieving better performance by exploring the entire optimization space.

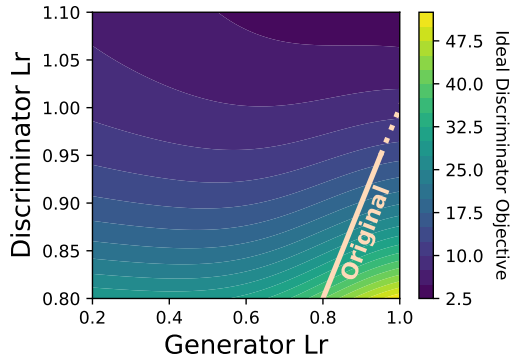

Figure 5. Illustration of the discriminator performance in terms of the ideal objective (denoted by the color, and the darker color corresponds to better performance) with respect to the generator learning rate (x-axis) and discriminator learning rate (y-axis). The line represents the space that the original ELECTRA design can possibly reach by modulating its learning rate in an ideal scenario. The dashed line represents the space that the original ELECTRA fails to reach in practice due to the training instability of the generator.

6. Experiments

6.1. Experiment Setup

Pretraining Setup. We conduct experiments with two standard settings, *Base* and *Large*, following previous works (Devlin et al., 2019; Meng et al., 2021; Bajaj et al., 2022). Specifically, we employ Wikipedia and BookCorpus (Zhu et al., 2015) (16 GB of texts, 256M samples) for pretraining with sequence length as 512. We use a cased sentence piece BPE vocabulary of 128K tokens following (He et al., 2020), since larger vocabulary size improves LLMs without significant additional training and inference cost (Bao et al., 2020).

We conduct pretraining for 125K updates with a batch size of 2048. For our DecoupledOptim, we use the same hyperparameter combination in both Base and Large settings, namely the generator learning rate is set as 2×10^{-4} and the discriminator learning rate is set as 1.5×10^{-3}. Detailed hyperparameter settings can be found in Appendix B.

Model Architecture. Our main model (discriminator) in the Base setting follows the $\text{BERT}_{\text{base}}$ architecture (Devlin et al., 2019), namely a 12-layer transformer with 768 hidden dimensions plus T5 relative position encoding (Raffel et al., 2019) with 32 bins. We employ Admin (Liu et al., 2020a; 2021) for model initialization to stabilize the training. Our main model in the Large setting follows $\text{BERT}_{\text{Large}}$, namely a 24-layer transformer with 1024 hidden dimensions and 128 relative position encoding bins. Our auxiliary model (generator) in Base has the same architecture as the main model, which is larger than the recommended size in previous works (Clark et al., 2020; Meng et al., 2021) (typically 4 layers), but yields significantly better results. Our auxiliary model in Large has 8 layers with other settings same as the main model.

Downstream evaluation setup. We conduct the evaluation on downstream tasks following the setup in previous works (Meng et al., 2021; Bajaj et al., 2022). Specifically, we evaluate on GLUE (Wang et al., 2018) language understanding benchmark with a single-task, single-model finetuning setting following previous works. We employ the suggested training hyperparameters such as the AdaMax optimizer (Kingma & Ba, 2015) from Liu et al. (2019a; 2020b). We report Spearman correlation on STS-B, Matthews correlation on CoLA, and accuracy on the rest of the datasets. Detailed hyperparameter settings can be found in Appendix B.

Baselines. We compare with various pretrained models that are consistent with our basic settings (dataset and training steps) (see Table 1). For the Large setting, we also compare with pretrained models that consume similar computation costs, for example, in terms of the total number of processed tokens. We report the results of baseline models from the corresponding papers and their follow-up works,

Table 1. Results on the GLUE development set. "†" indicates the model is pretrained for 1M updates with batch size of 256. "‡" indicates the model is pretrained for 100K updates with batch size of 8K. "-" indicates that no public reports are available.

Model	MNLI-(m/mm) (Acc.)	QQP (Acc.)	QNLI (Acc.)	SST-2 (Acc.)	CoLA (Mat. Corr.)	RTE (Acc.)	MRPC (Acc.)	STS-B (Spear. Corr.)	AVG
Base Setting									
BERT (Devlin et al., 2019)	84.5/ -	91.3	91.7	93.2	58.9	68.6	87.3	89.5	83.1
RoBERTa (Liu et al., 2019b)	85.8/85.5	91.3	92.0	93.7	60.1	68.2	87.3	88.5	83.3
XLNet (Yang et al., 2019)	85.8/85.4	-	-	92.7	-	-	-	-	-
DeBERTa (He et al., 2020)	86.3/86.2	-	-	-	-	-	-	-	-
TUPE (Ke et al., 2020)	86.2/86.2	91.3	92.2	93.3	63.6	73.6	89.9	89.2	84.9
ELECTRA (Clark et al., 2020)	86.9/86.7	91.9	92.6	93.6	66.2	75.1	88.2	89.7	85.5
MC-BERT (Xu et al., 2020)	85.7/85.2	89.7	91.3	92.3	62.1	75.0	86.0	88.0	83.7
COCO-LM (Meng et al., 2021)	88.5/88.3	92.0	93.1	93.2	63.9	84.8	91.4	90.3	87.2
AMOS (Meng et al., 2022)	88.9/88.7	92.3	93.6	94.2	**70.7**	86.6	90.9	**91.6**	88.6
DeBERTaV3 (He et al., 2021)	89.3/89.0	-	-	-	-	-	-	-	-
METRO (Bajaj et al., 2022)	89.0/88.8	92.2	93.4	**95.0**	70.6	86.5	91.2	91.2	88.6
$\text{METRO}_{\text{ReImp}}$	89.0/88.9	92.0	93.4	94.4	70.1	86.3	91.4	91.2	88.5
DecoupledOptim	**89.4/89.7**	**92.4**	**93.6**	94.7	70.6	**88.8**	**92.2**	91.1	**89.1**
Large Setting									
BERT†	86.6/ -	-	-	-	-	-	-	-	-
RoBERTa‡	89.0/ -	91.9	93.9	95.3	66.3	84.5	90.2	91.6	87.8
XLNet†	88.4/ -	91.8	93.9	94.4	65.2	81.2	90.0	91.1	87.0
TUPE†	88.2/88.2	91.7	93.6	95.0	67.5	81.7	90.1	90.7	87.3
$\text{METRO}_{\text{ReImp}}$	89.9/90.2	**92.5**	94.5	94.3	69.7	**88.8**	**91.9**	91.6	89.2
DecoupledOptim	**90.5/90.6**	92.4	**94.7**	**96.1**	**72.1**	88.4	91.2	**92.2**	**89.7**

whichever are higher. We also reimplement METRO as our baselines [6].

6.2. Results

Main Results. Table 1 lists the downstream evaluation results of DecoupledOptim and competitive baselines under the Base and Large setting. DecoupledOptim outperforms previous state-of-the-art by notable margins in terms of both the overall GLUE score and specific results on large datasets, which are considered to be more reliable.

Robustness to Generator Setting. We experiment with a wide variety of pretraining hyperparameters in the Base setting to validate the robustness of DecoupledOptim with respect to the change of generator capacity, since it is the main focus of this paper. We use our reimplemented METRO as a strong baseline. As shown in Figure 6, DecoupledOptim yields more stable downstream performance when the generator capacity varies/the hyperparameter changes. DecoupledOptim also achieves consistently better performance when employing generators with larger sizes.

Pretraining Efficiency. In each pretraining step, DecoupledOptim introduces no additional model parameters or computation cost compared to the original ELECTRA

[6]Both the re-implemented METRO and our method are implemented within the same codebase, which is built on top of FAIRSEQ, a popular open-sourced package (Ott et al., 2019).

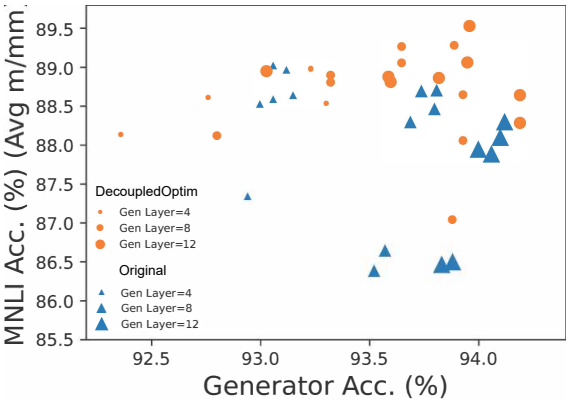

Figure 6. Discriminator performance on downstream tasks (MNLI Avg m/mm) versus the generator performance for a variety of hyperparameter combinations that can affect the generator training. For the original ELECTRA design (denoted by "▲") specifically, we modulate the loss weight λ in $\{50, 70, 100, 200\}$ and the learning rate in $\{10, 5, 2, 1\} \times 10^{-4}$. For our DecoupledOptim technique (denoted by "●") specifically, we modulate the generator learning rate also in $\{10, 5, 2, 1\} \times 10^{-4}$. For both, we modulate the generator depth in $\{4, 8, 12\}$. For diverged training, the combination of hyperparameters is not included in the figure.

design. Note that separate generator and discriminator embeddings in DecoupledOptim require exactly the same amount of operations as the shared embeddings in the original ELECTRA design, since the gradients of the embeddings have to be back-propagated from the generator and discriminator loss separately in both scenarios. DecoupledOptim does induce additional memory consumption due to separate embeddings. To mitigate this, one can maintain the shared embedding while employing separate optimizers, by merging the embedding gradients in a custom manner, which we would like to leave as a future work.

7. Conclusion

In this study, we conduct systematic analyses on the impact of generator capacity on the performance of the discriminator in ELECTRA-style pretraining. Our investigation begins with the observation that using a large auxiliary generator often results in a degradation of the downstream performance of the main discriminator model. Our analyses suggest that such performance degeneration is due to inadequate control of the generator capacity during pretraining, highlighting a long-overlooked issue in ELECTRA-style training. Based on our findings, we propose a simple-yet-effective method that greatly improves the training robustness and downstream performance, which further verified our intuition.

References

Arora, S., Cohen, N., Golowich, N., and Hu, W. A convergence analysis of gradient descent for deep linear neural networks. *ArXiv*, abs/1810.02281, 2018.

Bajaj, P., Xiong, C., Ke, G., Liu, X., He, D., Tiwary, S., Liu, T.-Y., Bennett, P., Song, X., and Gao, J. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *ArXiv*, abs/2204.06644, 2022.

Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Piao, S., Gao, J., Zhou, M., and Hon, H.-W. Unilmv2: Pseudo-masked language models for unified language model pre-training. *ArXiv*, abs/2002.12804, 2020.

Chi, Z., Huang, S., Dong, L., Ma, S., Singhal, S., Bajaj, P., Song, X., and Wei, F. Xlm-e: Cross-lingual language model pre-training via electra. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

Hao, Y., Dong, L., Bao, H., Xu, K., and Wei, F. Learning to sample replacements for electra pre-training. In *FINDINGS*, 2021.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654, 2020.

He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *ArXiv*, abs/2111.09543, 2021.

Kanakarajan, K. R., Kundumani, B., and Sankarasubbu, M. Small-bench nlp: Benchmark for small single gpu trained models in natural language processing. *ArXiv*, abs/2109.10847, 2021.

Ke, G., He, D., and Liu, T.-Y. Rethinking positional encoding in language pre-training. *ArXiv*, abs/2006.15595, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. Understanding the difficulty of training transformers. *ArXiv*, abs/2004.08249, 2020a.

Liu, L., Liu, J., and Han, J. Multi-head or single-head? an empirical comparison for transformer training. *ArXiv*, abs/2106.09650, 2021.

Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. In *Annual Meeting of the Association for Computational Linguistics*, 2019a.

Liu, X., Wang, Y., Ji, J., Cheng, H., Zhu, X., Awa, E., He, P., Chen, W., Poon, H., Cao, G., et al. The microsoft toolkit of multi-task deep neural networks for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 118–126, 2020b.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019b.

Meng, Y., Xiong, C., Bajaj, P., Tiwary, S., Bennett, P., Han, J., and Song, X. Coco-lm: Correcting and contrasting text sequences for language model pretraining. *ArXiv*, abs/2102.08473, 2021.

Meng, Y., Xiong, C., Bajaj, P., Tiwary, S., Bennett, P., Han, J., and Song, X. Pretraining text encoders with adversarial mixture of training signal generators. *ArXiv*, abs/2204.03243, 2022.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Raffel, C., Shazeer, N. M., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2019.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018.

Wei, F., Gao, Y., Wu, Z., Hu, H., and Lin, S. Aligning pretraining for detection via object-level contrastive learning. In *Neural Information Processing Systems*, 2021.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Chapter of the Association for Computational Linguistics*, 2017.

Xu, Z., Gong, L., Ke, G., He, D., Zheng, S., Wang, L., Bian, J., and Liu, T.-Y. Mc-bert: Efficient language pre-training via a meta controller. *ArXiv*, abs/2006.05744, 2020.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, 2019.

Zhang, Z., Zhao, H., Utiyama, M., and Sumita, E. Language model pre-training on true negatives. *ArXiv*, abs/2212.00460, 2022.

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 19–27, 2015.

Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. V. Rethinking pre-training and self-training. *ArXiv*, abs/2006.06882, 2020.

A. Proof

Bound the ideal discriminator objective. Given Assumption 1, it is easy to see that the ideal discriminator objective can be bounded as $L^*(\theta_D) \leq \tilde{L}^*(\theta_D)$, where

$$\tilde{L}^*(\theta_D) := \mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim\delta_{w|c}(\hat{w})}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}), \tag{11}$$

where $\delta_{w|c}(\hat{w})$ is a Dirac measure defined by the original token, namely the discriminator loss is maximized if the original token is always sampled.

We can now proceed to prove the two Lemmas in Section 5.

Lemma 1.

Proof. We mainly utilize the Cauchy-Schwartz inequality, the equivalence between p-norms and Pinsker's inequality in this proof.

First, we note that,

$$L^*(\theta_D) \leq \tilde{L}^*(\theta_D) = \bar{L}_G(\theta_D) + (\tilde{L}^*(\theta_D) - \bar{L}_G(\theta_D)),$$

where the second term, the difference between two objectives, can be further bounded as

$$\begin{aligned}
&\tilde{L}^*(\theta_D) - \bar{L}_G(\theta_D)\\
=&\mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim\delta_{w|c}}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w}) - \mathbb{E}_{c,w}\mathbb{E}_{w\sim p_G}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w})\\
=&\mathbb{E}_{c,w}\sum_{\hat{w}}\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w})(\delta_{w|c}(\hat{w}) - p_G(\hat{w}|c))\\
\leq&\mathbb{E}_{c,w}M_D(c,w)\left\|\delta_{w|c}(\hat{w}) - p_G(\hat{w}|c)\right\|_2\\
\leq&\mathbb{E}_{c,w}M_D(c,w)\left\|\delta_{w|c}(\hat{w}) - p_G(\hat{w}|c)\right\|_1\\
\leq&2^{-1/2}\mathbb{E}_{c,w}M_D(c,w)\left[D_{\mathrm{KL}}(\delta_{w|c}(\hat{w})\|p_G(\hat{w}|c))\right]^{1/2}\\
=&2^{-1/2}\mathbb{E}_{c,w}M_D(c,w)\left[H(\delta_{w|c}(\hat{w}),p_G(\hat{w}|c))\right]^{1/2}\\
\leq&2^{-1/2}m_D\left[\mathbb{E}_{c,w}H(\delta_{w|c}(\hat{w}),p_G(\hat{w}|c))\right]^{1/2}\\
=&2^{-1/2}m_D\left[\mathbb{E}_{c,w}H(\delta_{\hat{w}}(w|c),p_G(\hat{w}|c))\right]^{1/2}\\
=&2^{-1/2}m_D\left[\mathbb{E}_{c}H(q(\hat{w}|c),p_G(\hat{w}|c))\right]^{1/2}\\
=&2^{-1/2}m_D L(\theta_G)^{1/2}
\end{aligned} \tag{12}$$

where $M_D(c,w) = \|\ell(D(c,\hat{w}),\mathbf{1}_{\hat{w}=w})\|_2$ and $m_D = \max_{c,w} M_D(c,w)$. □

Lemma 2.

Proof. We mainly utilize a corollary of Jensen's inequality on concave functions, as well as a simple trick that expands

$\mathbb{E}[\log x]$ around $\log \mathbb{E}[x]$.

$$
\begin{aligned}
&\bar{L}_G(\theta_D) \\
&= \mathbb{E}_{c,w}\mathbb{E}_{\hat{w}\sim p_G}\ell(D(c,\hat{w}), \mathbf{1}_{\hat{w}=w}) \\
&\le \mathbb{E}_{c,w}\, F_D(\mathbb{E}_{\hat{w}\sim p_G}\mathbf{1}_{\hat{w}=w}) \\
&\le \mathbb{E}_{c,w}\, F_D(p_G(w|c)) \\
&= \mathbb{E}_{c,w}\, F_D(e^{\log P_G(w|c)}) \\
&\le F_D(\mathbb{E}_{c,w} e^{\log P_G(w|c)}) \\
&= F_D\left(\mathbb{E}_{c,w}\left[e^{\mathbb{E}\log p_G(w|c)} + e^{\mathbb{E}\log p_G(w|c)}(\log p_G(w|c) - \mathbb{E}\log p_G(w|c)) + \right.\right. \\
&\qquad\qquad \left.\left. \frac{1}{2}e^{\mathbb{E}\log p_G(w|c)+\varepsilon}(\log p_G(w|c) - \mathbb{E}\log p_G(w|c))^2\right]\right) \\
&= F_D\left(e^{\mathbb{E}\log p_G(w|c)} + \frac{V_G}{2}e^{\mathbb{E}\log p_G(w|c)+\varepsilon}\right) \\
&= F_D\left(\left(1+\frac{V_G}{2}\right)e^{-L(\theta_G)} + \frac{V_G}{2}e^{\varepsilon}\right),
\end{aligned}
$$

where $V_G = \mathbb{E}(\log p_G(w|c) - \mathbb{E}\log p_G(w|c))^2$.

□

Theorem 1.

Proof. This is a direct consequence of Lemma 1 and Lemma 2. We can write

$$
\begin{aligned}
L^*(\theta_D) \le \tilde{L}^*(\theta_D) &= \bar{L}_G(\theta_D) + (\tilde{L}^*(\theta_D) - \bar{L}_G(\theta_D)) \\
&\le F_D\left((1+V_G/2)\, e^{-\bar{L}(\theta_G)} + V_G/2\; e^{\varepsilon}\right) + 2^{-1/2} m_D L(\theta_G)^{1/2}.
\end{aligned} \tag{13}
$$

Note that the discriminator optimization only applies on the first term. Using the existing result Proposition 1 on both the generator and discriminator optimization yields the theorem.

□

B. Hyperparameter settings

Our hyperparameter settings follow the standard practice in previous works. For MLM pretraining of the generator, we fix the mask ratio as 15%. When sampling sequences for pretraining, we respect document boundaries and avoid concatenating texts from different documents. We did not mask special tokens following the standard BERT practice. We conduct pretraining on NVIDIA Tesla V100 with 32GB memory and fine-tuning on NVIDIA Tesla P100 with 16GB memory. Table 2 lists the detailed hyperparameters used in pretraining. Table 3 lists the detailed hyperparameters used for fine-tuning.

C. A Roadmap to Hyperparameter Tuning

The separate generator and discriminator optimizers introduced by DecoupledOptim are in line with our understanding of the distinct roles of generator optimization and discriminator optimization in ELECTRA-style training. Therefore, DecoupledOptim is much more friendly to hyperparameter tuning, which often requires excessive efforts especially for LLM pre-training. Here we provide a guideline on the hyperparameter selection in DecoupledOptim to achieve the best pretraining performance.

First, since DecoupledOptim is no longer sensitive to the generator size, one can choose a generator as large as possible given hardware constraints. Next, one can find a discriminator learning rate as large as possible with the generator learning rate fixed as any value, provided that the training is stable. This is based on our understanding that a larger discriminator learning rate can almost always benefit the pretraining. Finally, one can locate the best generator learning rate through an

Table 2. Hyperparameter settings used in pretraining.

Hyperparameters	**Base**	**Large**
Max Steps	125K	125K
Optimizer	Adam	Adam
Peak Learning Rate (Generator)	2×10^{-4}	2×10^{-4}
Peak Learning Rate (Discriminator)	1.5×10^{-3}	1.5×10^{-3}
Batch Size	2048	2048
Warm-Up Steps	10K	10K
Sequence Length	512	512
Relative Position Encoding Buckets	32	128
Relative Position Encoding Max Distance	128	256
Adam ϵ	1e$-$6	1e$-$6
Adam (β_1, β_2)	$(0.9, 0.98)$	$(0.9, 0.98)$
Clip Norm	2.0	2.0
Dropout	0.1	0.1
Weight Decay	0.01	0.01

Table 3. Hyperparameter search space in fine-tuning.

Hyperparameters	**Base**	**Large**
Sequence Length	256	256
Optimizer	AdaMax	AdaMax
Peak Learning Rate	{5e-5,1e-4, 3e-4}	{5e-5,1e-4, 3e-4}
Max Epochs	{2,3,5,10, 20}	{2,3,5,10, 20}
Batch size	{16, 32, 64, 128}	{8, 16, 32, 64}
Learning rate decay	Linear	Linear
Weight Decay	{0, 0.01}	{0, 0.01}
Warm-up Proportion	{6 %, 10 %, 30%}	{6 %, 10 %, 30%}
Adam ϵ	1e-6	1e-6
Adam (β_1, β_2)	$(0.9, 0.98)$	$(0.9, 0.98)$
Gradient Clipping	1.0	1.0
Dropout	0.1	0.1

efficient binary search, since increasing the generator learning rate first helps and then hurts the training. In Figure 7, we show our practice of finding the best hyperparameter combination for the Base setting.

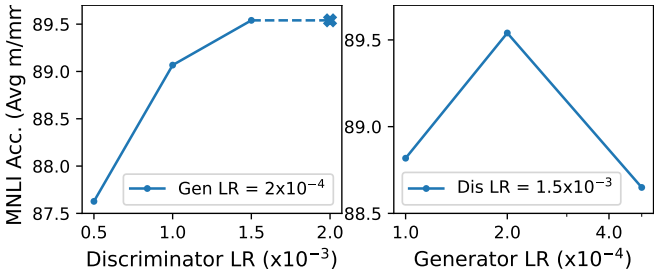

Figure 7. Practice of finding the best hyperparameter combination for the Base setting. (Left): We search the largest possible discriminator learning rate as long as the training is stable. (Right): Upon locate the largest discriminator learning rate, we binary search the best generator learning rate since the its correlation with the performance is an unimodal curve.

D. Understand Generator "Overfitting" in ELECTRA

In ELECTRA-style pretraining, it has been widely observed that a large generator consistently hurts the discriminator performance, as also shown in Figure 1 for "Baseline". It has been speculated that a large generator may prevent the discriminator from learning effectively (Clark et al., 2020), yet the exact reason remains largely unclear.

Does a large generator hurt cross-domain generalization? A natural explanation to the "overfitting" phenomenon in ELECTRA-style pretraining is that a large generator may impair the cross-domain generalization, namely, it hurts the transferability of the discriminator to downstream tasks since the gap between pretraining performance and downstream performance is commonly seen, especially when the pretraining task and the downstream task are significantly different (Zoph et al., 2020; Wei et al., 2021).

However, we show that the generator "overfitting" phenomenon is not, or at least not completely, due to a larger transferring gap between RTD-based pretraining and downstream tasks such as MNLI. A large generator may already hurt the pretraining performance, in that the discriminator becomes less effective on detecting replaced tokens.

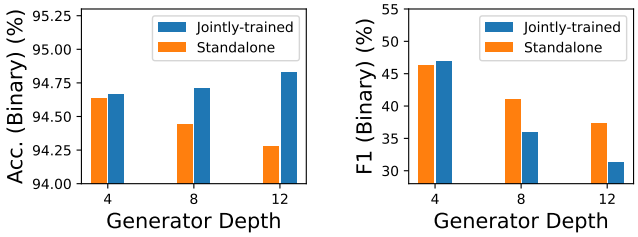

Figure 8. The binary classification performance of the discriminators on detecting replaced tokens generated by their individual jointly-trained generators versus that by a standalone generator. (Left): Accuracy (Right): F1-score. Here the standalone generator has 4 layers and is trained for 125k steps.

Fair evaluation of RTD performance. To demonstrate this, we need to first measure the RTD performance of discriminators jointly trained with different generator sizes in a fair manner. The standard binary classification performance reported in RTD-based pertraining is clearly not a measure that is comparable across different discriminators, as they are evaluated with different jointly-trained generators.

Alternatively, we propose to use a standalone generator to measure the RTD performance fairly. A standalone generator is pretrained and shared across evaluations which means we can maintain the difficulty of the generated replaced tokens for different discriminators. As shown in Figure 8, the binary classification performance of discriminators against such a standalone generator ranks consistently with different metrics, while the jointly-trained generators report mixed rankings. This suggests the standalone generator is more reliable to measure the RTD performance.

Large generators cause inferior RTD performance. With a fair measure of the RTD pretraining performance, we can now observe that a larger generator in fact hurts the pretraining. As shown in Figure 9, when trained with a larger generator,

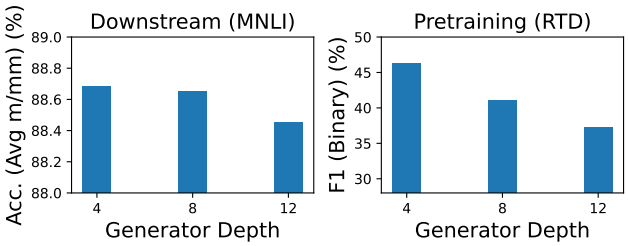

Figure 9. (Left): Averaged matched/mismatched accuracy on the MNLI dataset by fine-tuning discriminators trained along with different generator sizes. (Right): Fair measure of the RTD performance of discriminators jointly trained with generators of different sizes. Here the standalone generator has 4 layers and is trained for 125k steps.

the discriminator is not able to detect the replaced tokens as effectively. Subsequently, the performance on downstream tasks degrades as well. Therefore, to understand and potentially and rectify generator "overfitting", it is necessary to first dig into the pretraining stage of ELECTRA.

D.1. Does a large generator hurt in-domain generalization?

We conjecture that a large generator hurts RTD-based pretraining because the replaced tokens generated by it may lack diversity and be overfitted easily, thus hurting RTD performance.

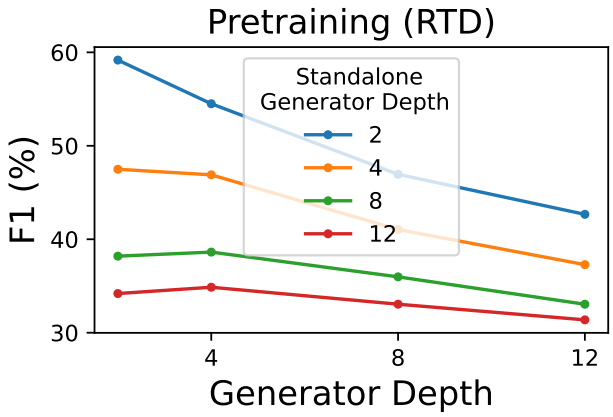

Figure 10. RTD performance (F1-score) of discriminators jointly trained with generators of different sizes (2, 4, 8, 12 layers), measured against standalone generators of different sizes (2, 4, 8, 12 layers)

Overfitting the jointly-trained generator? One possibility is that a large generator may predict the token at some masked positions with a low-entropy distribution, which means the replaced token at this particular position may almost always be the same. This prevents the discriminator from learning to detect other diverse replaced tokens. However, we find that this might not be the case. For example, a BERT-Base discriminator gets an RTD performance of 31.4% in terms of F1-score against its jointly-trained generator with 12 layers. In comparison, when evaluated against a standalone generator sharing the exact same architecture as the jointly-trained generator, but initialized with a different random seed, it gets an F1-score of 30.9%, which shows no significant discrepancy. Therefore, it is not likely that the discriminator is overfitting a jointly-trained large generator itself.

Overfitting the size? It is also possible that large generators may generate replaced tokens with similar properties. For example, the replaced tokens generated by different generators, albeit randomly initialized, are the same or follow similar distributions. In this case, the discriminator may overfit generators of this particular size. However, Figure 10 shows that this may not be the case. A large generator also hurts the RTD performance against a standalone generator of exactly the same size. Furthermore, one may find that a larger generator in fact hurts the RTD performance against standalone generators of multiple different sizes, which suggests that a large generator may hurt RTD-based pretraining "universally".

D.2. How to reliably measure the performance of RTD pretraining?

In the above sections we mentioned that to reliably measure the performance of RTD pretraining, it is better to evaluate the discriminator against a standalone generator. However, we note that not all standalone generators can reliably measure the RTD performance such that it can reflect the downstream performance. As shown in Figure 10, a standalone generator with 2 layers will erroneously report the discriminator trained with a 2-layer generator is better than the one trained with a 4-layer generator, while on downstream tasks the latter is better. Therefore, we speculate that this is because the replaced token distributions generated by a 2-layer generator is too simple in the verge of random, thus is not able to reflect the discriminator's capacity of language understanding. Based on this observation, we believe to reliably measure the RTD performance we should sample replaced tokens from a distribution that is sufficiently difficult, for example, a generator with a reasonably large size.