ELECTRA: Pre-Training Text Encoders as Discriminators Rather than Generator

Kevin Clark
Stanford University
kevclark@cs.stanford.edu

Minh-Thang Luong
Google Brain
thangluong@google.com

Quoc V. Le
Google Brain
qvl@google.com

Christopher D. Manning
Stanford University & CIFAR Fellow
manning@cs.stanford.edu

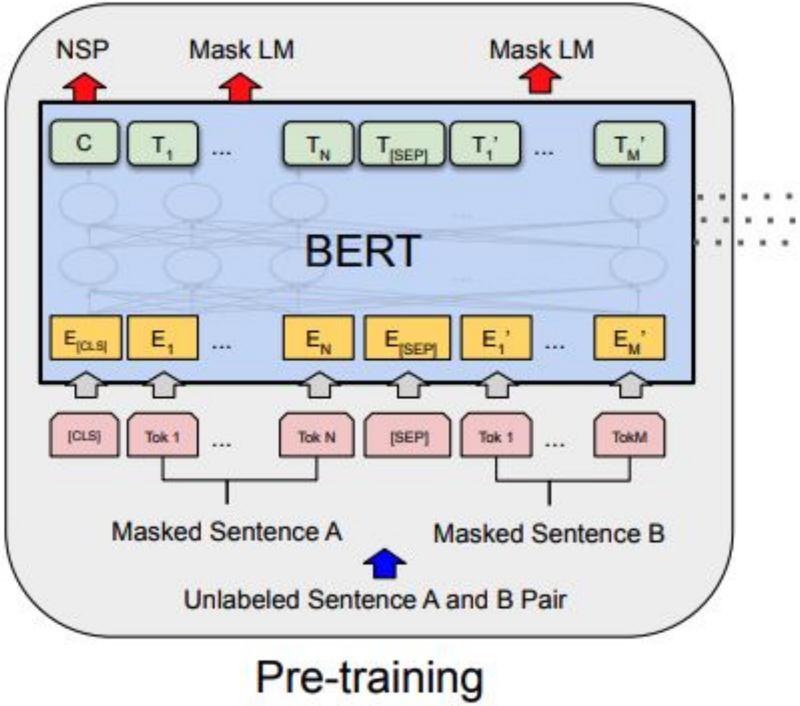
Outline

- Introduction
- Method
- Model Extensions
- Experiment Results

Mask Language Model

MLM learns from 15% of the tokens per example.

Mismatch in BERT: The artificial [MASK] tokens are used during pre-training but are not used when being fine-tuned on downstream tasks.



Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171-4186).

Replace Token Detection

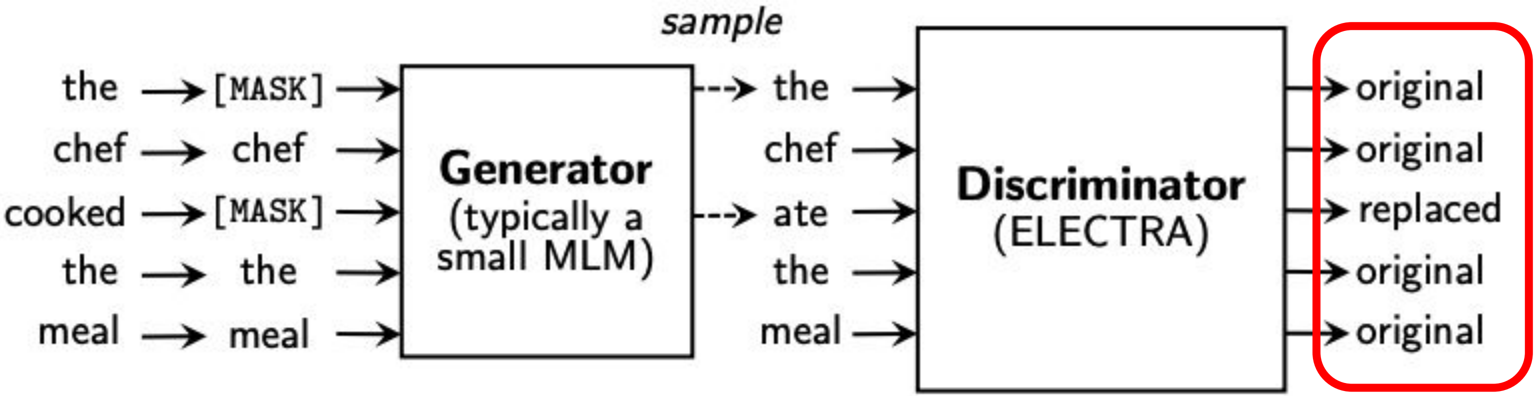
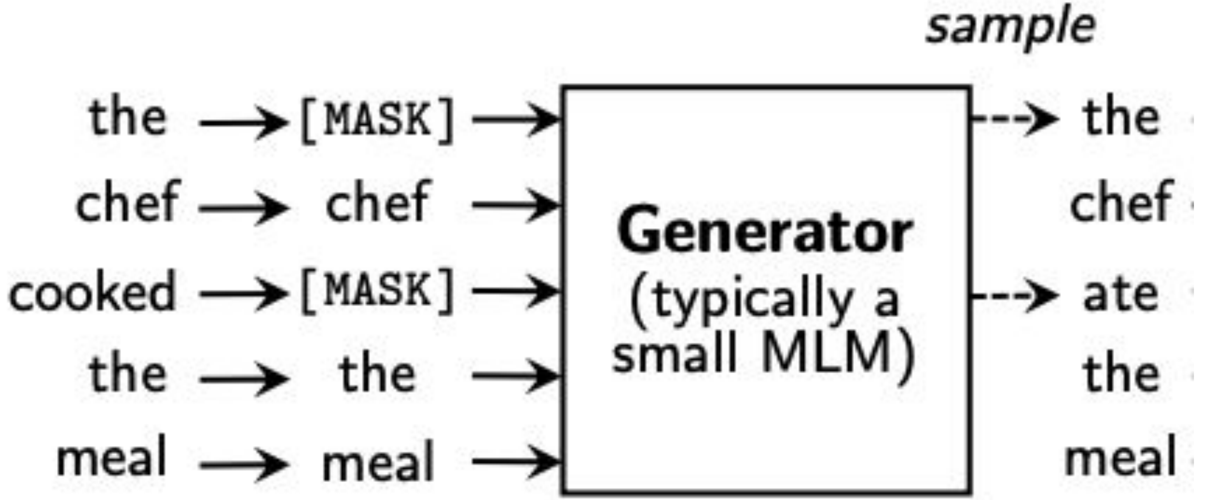


Figure 2: An overview of replaced token detection. The generator can be any model that produces an output distribution over tokens, but we usually use a small masked language model that is trained jointly with the discriminator. Although the models are structured like in a GAN, we train the generator with maximum likelihood rather than adversarially due to the difficulty of applying GANs to text. After pre-training, we throw out the generator and only fine-tune the discriminator (the ELECTRA model) on downstream tasks.

Generator

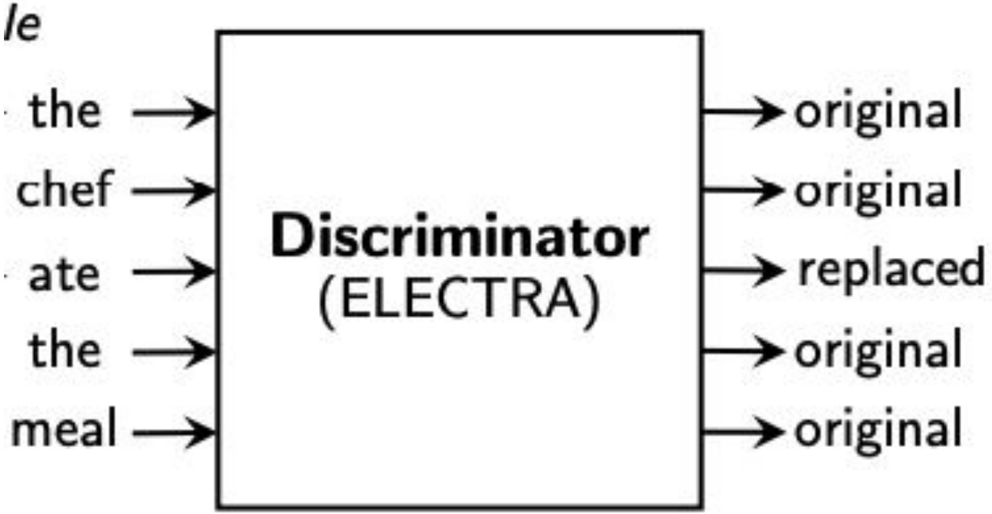


$$m_i \sim \text{unif}\{1, n\} \text{ for } i = 1 \text{ to } k \qquad \boldsymbol{x}^{\text{masked}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \texttt{[MASK]})$$

$$p_G(x_t|\boldsymbol{x}) = \exp\left(e(x_t)^T h_G(\boldsymbol{x})_t\right) / \sum_{x'} \exp\left(e(x')^T h_G(\boldsymbol{x})_t\right)$$

$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i|\boldsymbol{x}^{\text{masked}})\right)$$

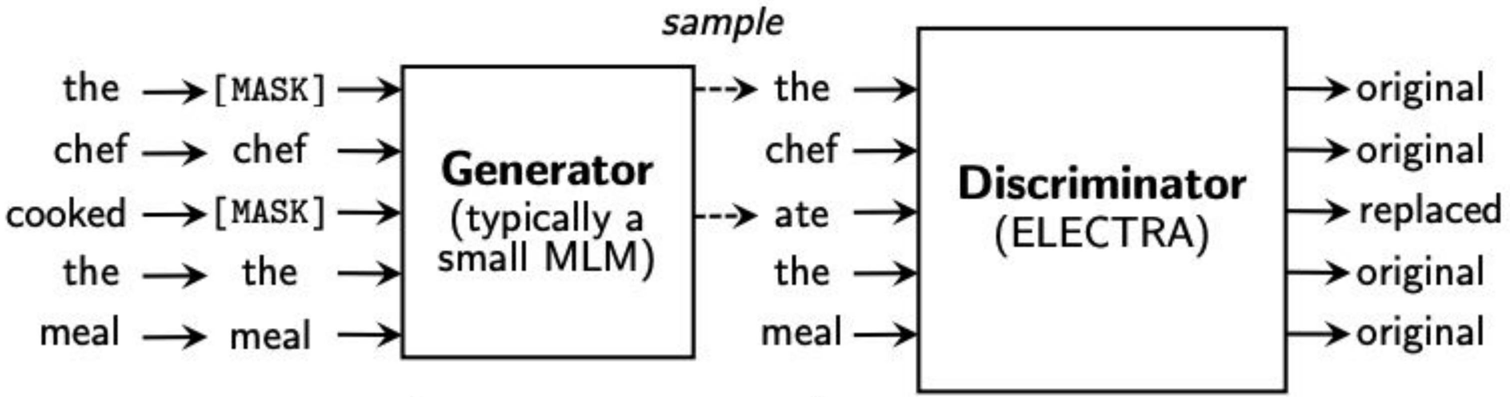
Discriminator



$$\hat{x}_i \sim p_G(x_i|\boldsymbol{x}^{\text{masked}}) \text{ for } i \in \boldsymbol{m} \qquad \boldsymbol{x}^{\text{corrupt}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{x}})$$

$$D(\boldsymbol{x}, t) = \text{sigmoid}(w^T h_D(\boldsymbol{x})_t)$$

$$\mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\mathbb{1}(x_t^{\text{corrupt}} = x_t) \log D(\boldsymbol{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t) \log(1 - D(\boldsymbol{x}^{\text{corrupt}}, t))\right)$$



$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i | \boldsymbol{x}^{\text{masked}})\right)$$

$$\mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\mathbb{1}(x_t^{\text{corrupt}} = x_t) \log D(\boldsymbol{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t) \log(1 - D(\boldsymbol{x}^{\text{corrupt}}, t))\right)$$

$$\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D)$$

Evaluation

Evaluate on the General Language Understanding Evaluation (GLUE) benchmark and Stanford Question Answering (SQuAD) dataset.

GLUE contains a variety of tasks covering textual entailment (RTE and MNLI) question-answer entailment (QNLI), paraphrase (MRPC), question paraphrase (QQP), textual similarity (STS), sentiment (SST), and linguistic acceptability (CoLA).

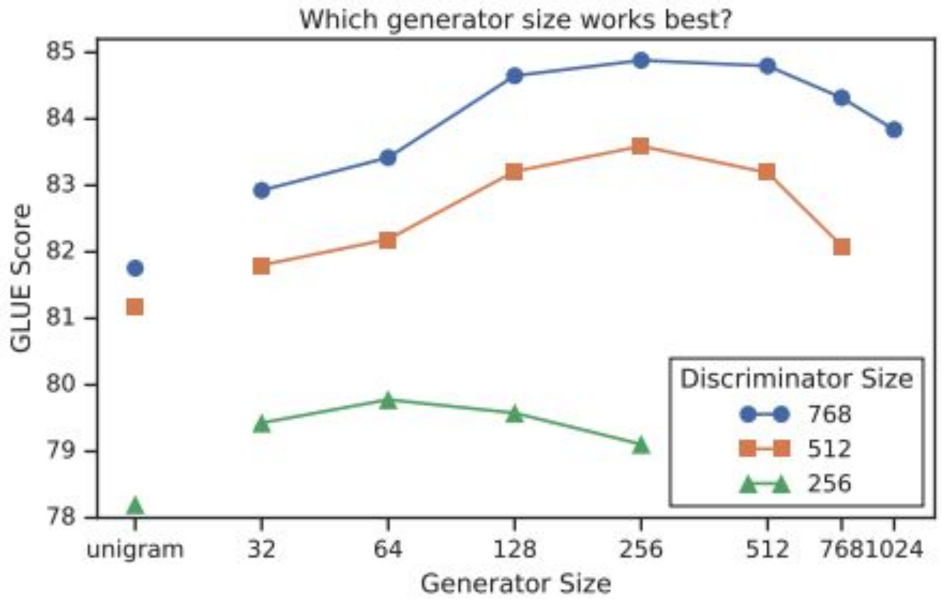
Weight Sharing Embedding Layers

To improve the efficiency of the pre-training by sharing weights between the generator and discriminator.

- When the generator is the ***same size*** as the discriminator, GLUE scores are **83.6** for no weight tying, **84.3** for tying token embeddings, and **84.4** for tying all weights.
- MLM is particularly effective at learning representations.
- Tying all encoder weights caused **little improvement** while incurring the significant disadvantage of requiring the generator and discriminator to be the **same size**.
- Use **tied embeddings** for further experiments in this paper.

Generator and Discriminator Capacity

They suggest using a smaller generator to reduce the model size.



- Make models smaller by **decreasing the layer sizes** while keeping the other hyperparameters constant
- Models work best with generators **1/4-1/2 the size** of the discriminator.
- Having too strong of a generator may pose a **too-challenging task** for the discriminator, preventing it from learning as effectively.

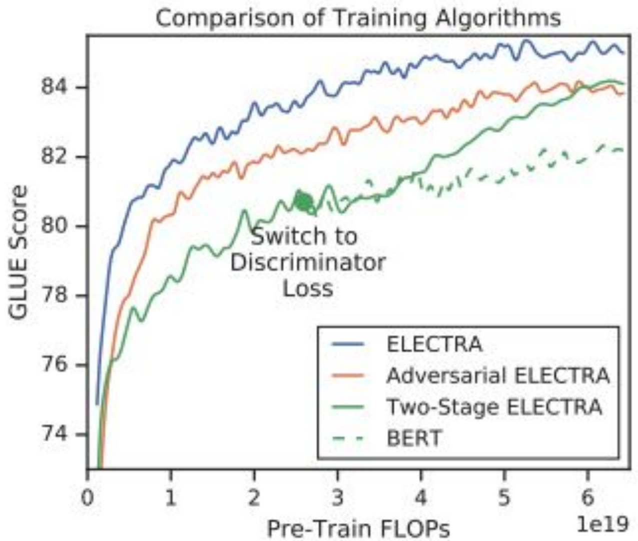
Training Algorithms

$$\min_{\theta_G,\theta_D} \sum_{\boldsymbol{x}\in\mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x},\theta_G) + \lambda\mathcal{L}_{\text{Disc}}(\boldsymbol{x},\theta_D)$$

- Jointly train the generator and discriminator.
- Two-Stage Curriculum Learning

1. Train only the generator with $\mathcal{L}_{\text{MLM}}$ for n steps.
2. Initialize the weights of the discriminator with the weights of the generator. Then train the discriminator with $\mathcal{L}_{\text{Disc}}$ for n steps, keeping the generator's weights frozen.

- Adversarially Training Generatore with Reinforcement Learning



Results

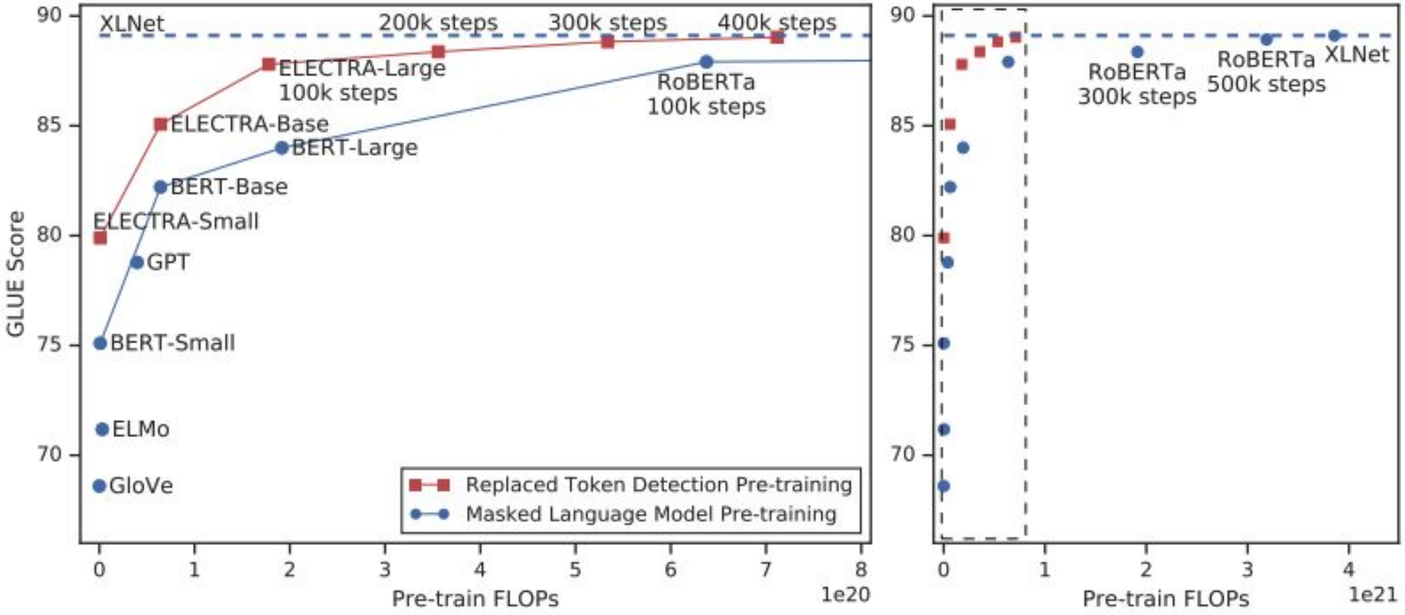


Figure 1: Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.

Results

Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

Table 1: Comparison of small models on the GLUE dev set. BERT-Small/Base are our implementation and use the same hyperparameters as ELECTRA-Small/Base. Infer FLOPs assumes single length-128 input. Training times should be taken with a grain of salt as they are for different hardware and with sometimes un-optimized code. ELECTRA performs well even when trained on a single GPU, scoring 5 GLUE points higher than a comparable BERT model and even outscoring the much larger GPT model.

Results

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

Table 2: Comparison of large models on the GLUE dev set. ELECTRA and RoBERTa are shown for different numbers of pre-training steps, indicated by the numbers after the dashes. ELECTRA performs comparably to XLNet and RoBERTa when using less than 1/4 of their pre-training compute and outperforms them when given a similar amount of pre-training compute. BERT dev results are from Clark et al. (2019).

Results

Model	Train FLOPs	Params	SQuAD 1.1 dev		SQuAD 2.0 dev		SQuAD 2.0 test	
			EM	F1	EM	F1	EM	F1
BERT-Base	6.4e19 (0.09x)	110M	80.8	88.5	–	–	–	–
BERT	1.9e20 (0.27x)	335M	84.1	90.9	79.0	81.8	80.0	83.0
SpanBERT	7.1e20 (1x)	335M	88.8	94.6	85.7	88.7	85.7	88.7
XLNet-Base	6.6e19 (0.09x)	117M	81.3	–	78.5	–	–	–
XLNet	3.9e21 (5.4x)	360M	**89.7**	**95.1**	87.9	**90.6**	87.9	90.7
RoBERTa-100K	6.4e20 (0.90x)	356M	–	94.0	–	87.7	–	–
RoBERTa-500K	3.2e21 (4.5x)	356M	88.9	94.6	86.5	89.4	86.8	89.8
ALBERT	3.1e22 (44x)	235M	89.3	94.8	87.4	90.2	88.1	90.9
BERT (ours)	7.1e20 (1x)	335M	88.0	93.7	84.7	87.5	–	–
ELECTRA-Base	6.4e19 (0.09x)	110M	84.5	90.8	80.5	83.3	–	–
ELECTRA-400K	7.1e20 (1x)	335M	88.7	94.2	86.9	89.6	–	–
ELECTRA-1.75M	3.1e21 (4.4x)	335M	**89.7**	94.9	**88.0**	**90.6**	**88.7**	**91.4**

Table 4: Results on the SQuAD for non-ensemble models.

Efficiency Analysis

Compare a series of other pre-training objectives that are designed to be a set of "stepping stones" between BERT and ELECTRA

- ELECTRA 15%
- Replace MLM
- All-Tokens MLM

Model	ELECTRA	All-Tokens MLM	Replace MLM	ELECTRA 15%	BERT
GLUE score	85.0	84.3	82.4	82.4	82.2

Table 5: Compute-efficiency experiments (see text for details).

Efficiency Analysis

- The small models are trained fully to convergence.
- ELECTRA is more parameter-efficient than BERT

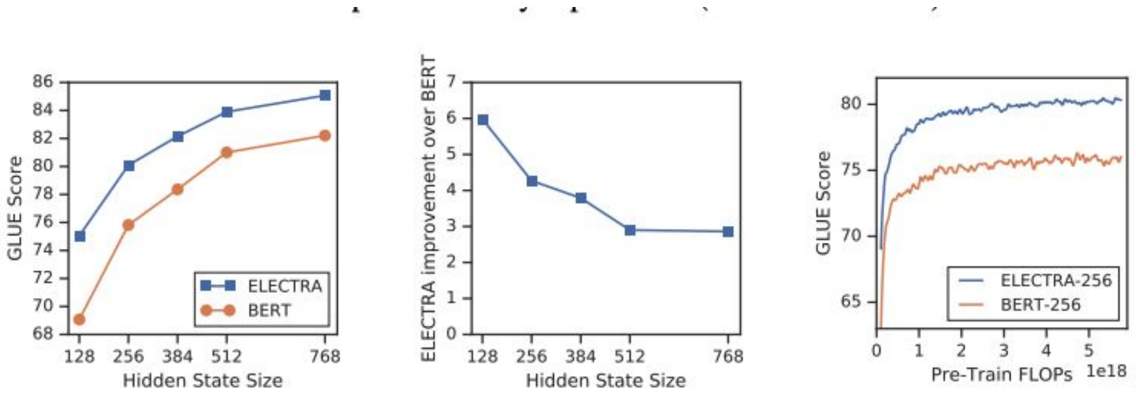


Figure 4: Left and Center: Comparison of BERT and ELECTRA for different model sizes. Right: A small ELECTRA model converges to higher downstream accuracy than BERT, showing the improvement comes from more than just faster training.

Questions?