ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS

Kevin Clark, *Minh-Thang Luong*, *Quoc V. Le*, *Christopher D. Manning*

International Conference on Learning Representations (ICLR), 2020

Presented by: Richard

Department of Computer Science

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

What ELECTRA can do:

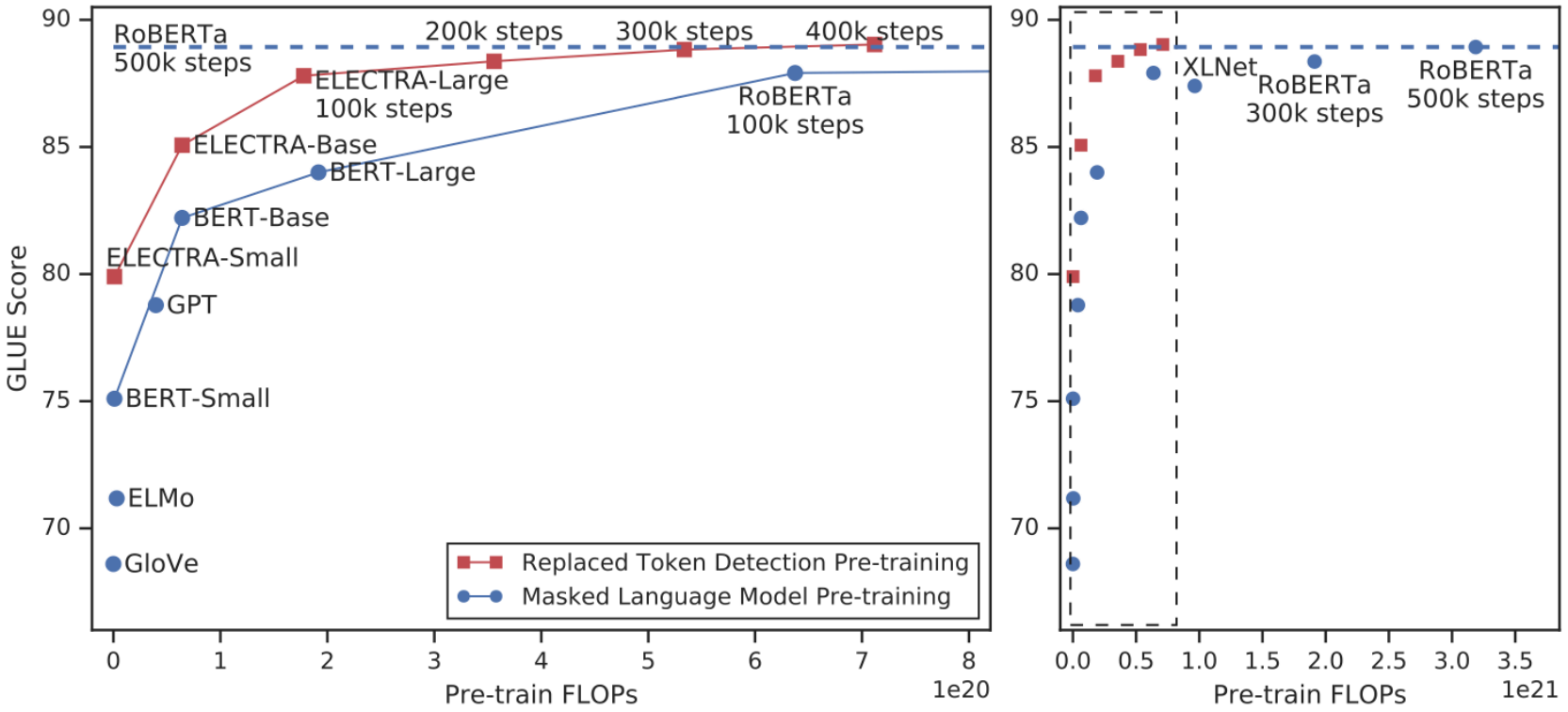


Figure 1: Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.

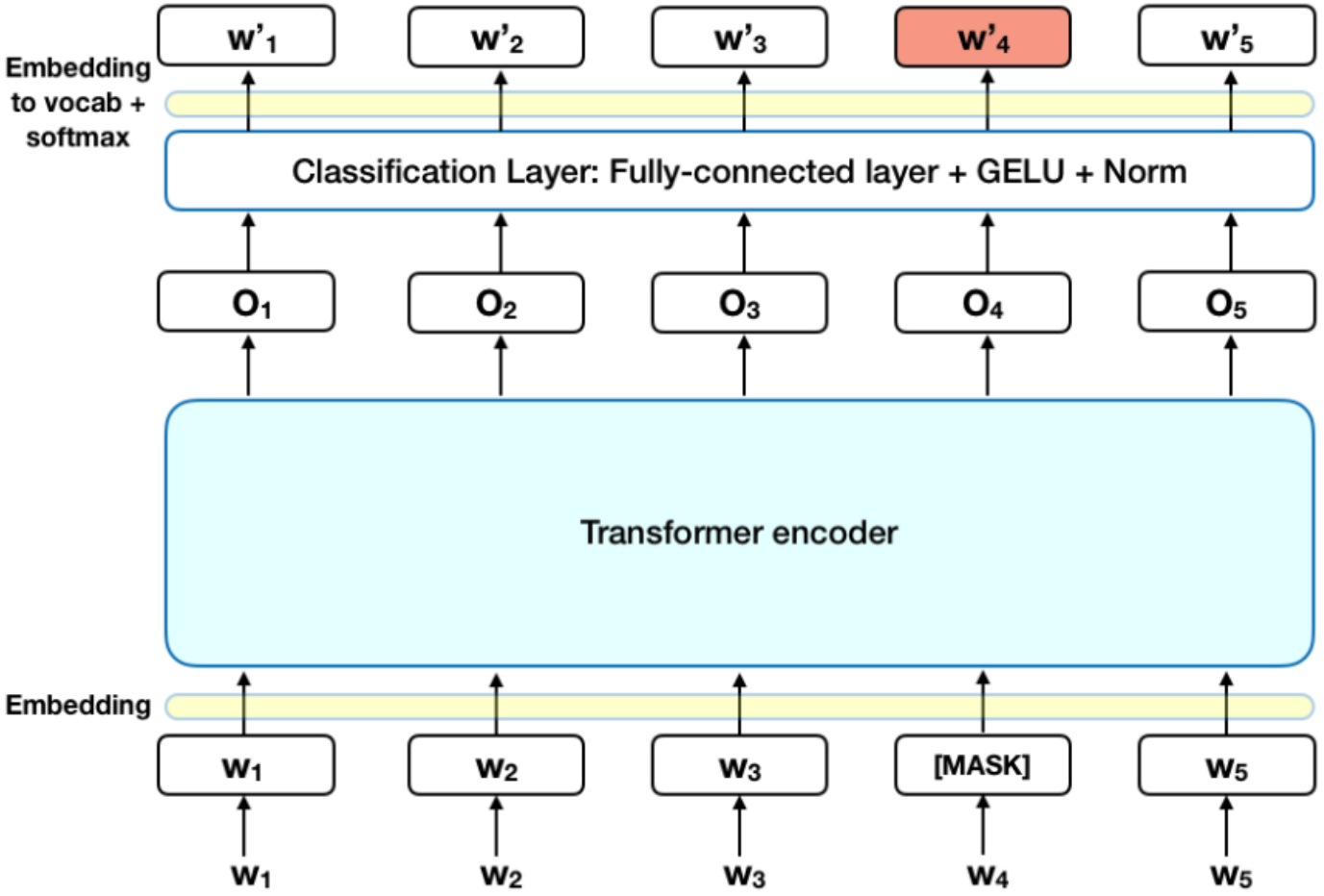
Outline

1. ELECTRA Model
    1. MLM
    2. Replaced Token Detection
    3. Differences with GAN
2. Experimental Results
    1. Model extensions
    2. Small models
    3. Large models
    4. Efficiency Analysis

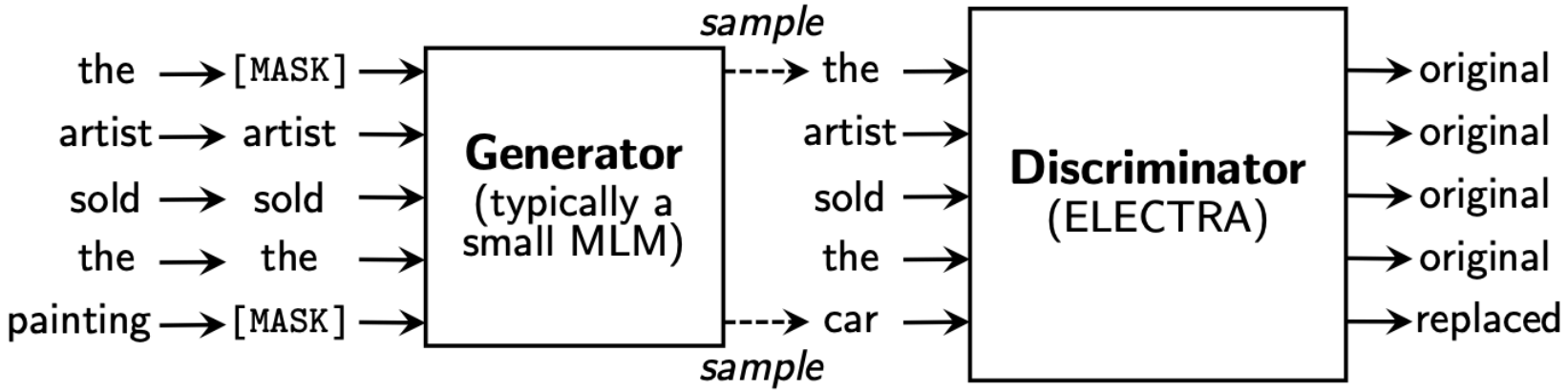
Outline

1. ELECTRA Model
    1. MLM
    2. Replaced Token Detection
    3. Differences with GAN
2. Experimental Results
    1. Model extensions
    2. Small models
    3. Large models
    4. Efficiency Analysis

1. MLM


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

2. Replaced Token Detection



Masking Schem:

$$m_i \sim \text{unif}\{1, n\} \text{ for } i = 1 \text{ to } k$$

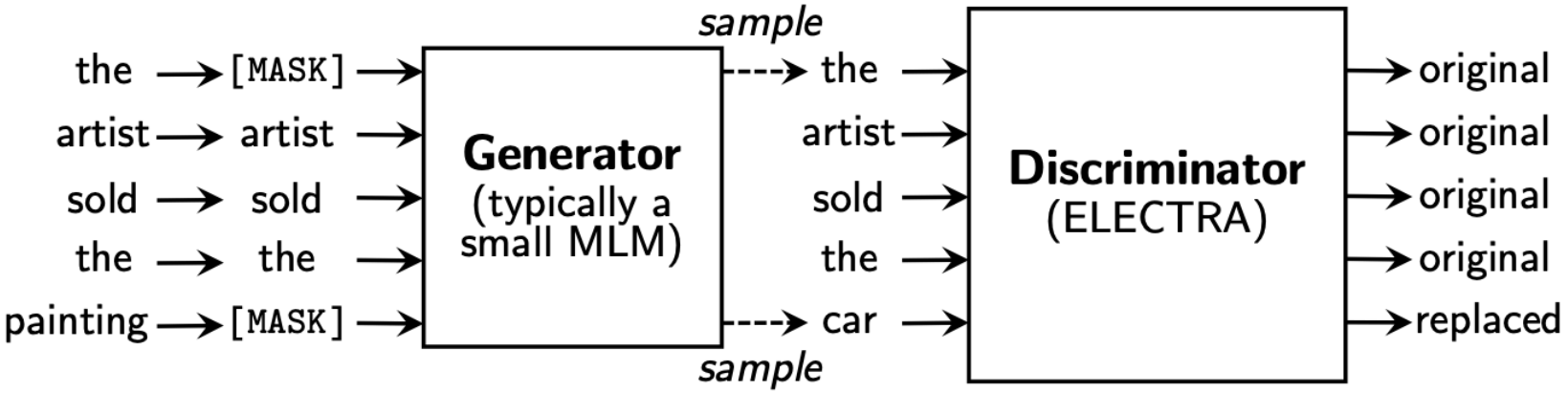
$$\boldsymbol{x}^{\text{masked}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \texttt{[MASK]})$$

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

$$\boldsymbol{x}^{\text{corrupt}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{x}})$$

Typically k = [0.15n], i.e., 15% of the tokens are masked out.

2. Replaced Token Detection

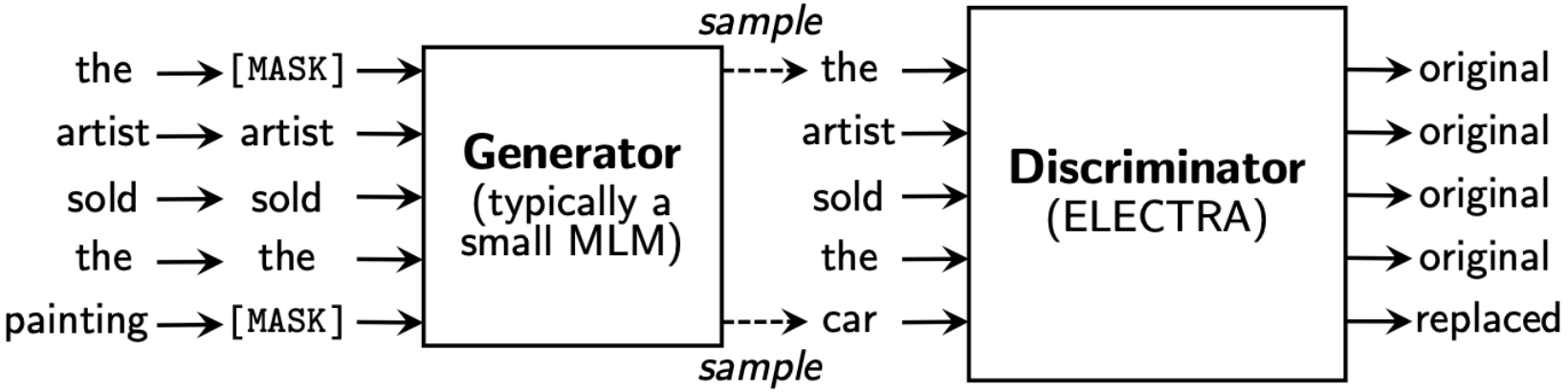


Output:

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

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

2. Replaced Token Detection

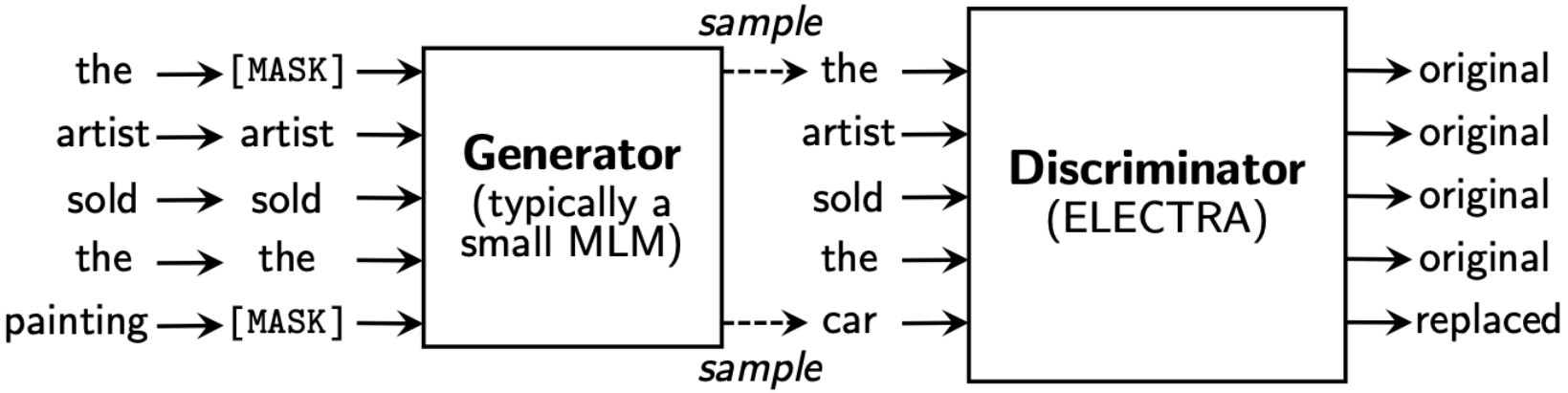


$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i | \boldsymbol{x}^{\text{masked}})\right)$$

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

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

3. Differences with GAN



1. If the generator happens to generate the correct token, that token is considered REAL instead of FAKE.

2. The generator is trained with MML rather than adversairal training. (Sampling)

3. Input of the generator is not a noise vector.

Outline

1. ELECTRA Model
    1. MLM
    2. Replaced Token Detection
    3. Differences with GAN
2. Experimental Results
    1. Model extensions
    2. Small models
    3. Large models
    4. Efficiency Analysis

1. Model extension

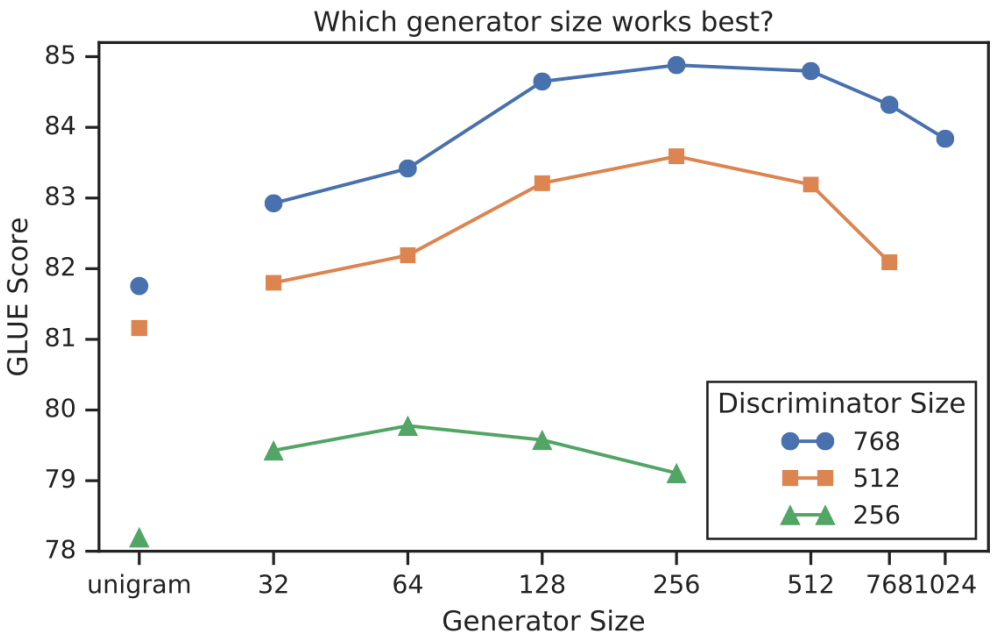
1. After pre-training, throw out the generator and fine-tune the discriminator on downstream tasks.

2. Sharing weights between the generator and discriminator
   83.6 for no weight sharing, 84.3 for sharing embeddings,
   84.4 for sharing all weights

3. Models work best with generators 1/4-1/2 the hidden size of the discriminator.
   Speculation: having too strong of a generator may pose a too-challenging task for the discriminator, preventing it from learning as effectively

4. Joint training better than two stages training/adversarial training:
   Train only the generator for n steps, then initialize the discriminator with the weights of the generator. Then train the discriminator for n steps keeping the generator's weight forzen.

1. Model extension



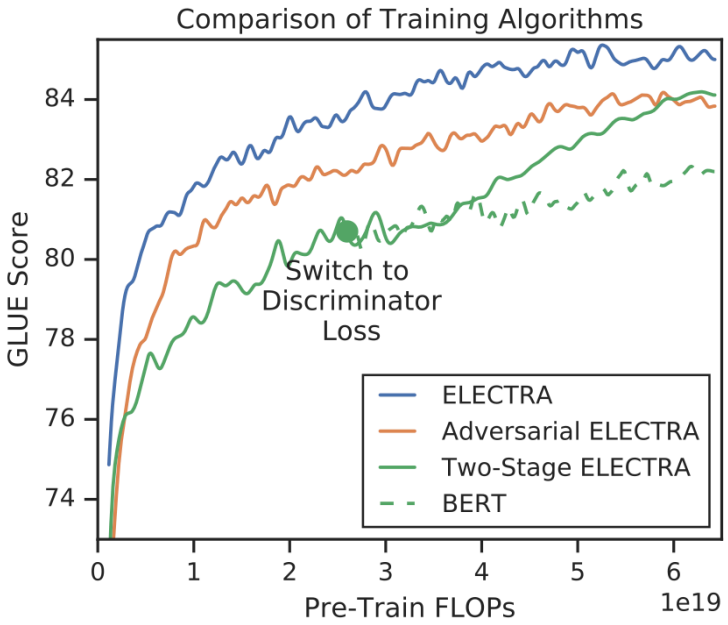


Figure 3: Left: GLUE scores for different generator/discriminator sizes (number of hidden units). Interestingly, having a generator smaller than the discriminator improves results. Right: Comparison of different training algorithms. As our focus is on efficiency, the x-axis shows FLOPs rather than train steps (e.g., ELECTRA is trained for fewer steps than BERT because it includes the generator).

2. Small Models 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
DistilBERT	– / 1.4e10	– / 2x	66M	–	77.8
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4D on 1 V100 GPU	79.9
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.

3. Large Models 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
XLNet	9.6e20 (1.3x)	360M	63.6	95.6	89.2	91.8	91.8	89.8	93.9	83.8	87.4
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	3.2e21 (4.5x)	356M	68.0	**96.4**	90.9	**92.4**	92.2	90.2	**94.7**	86.6	88.9
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	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	**90.5**	94.5	**86.8**	**89.0**
Test set results for models with standard single-task finetuning (no ensembling, task-specific tricks, etc.)											
BERT	1.9e20 (0.27x)	335M	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	83.3
SpanBERT	7.1e20 (1x)	335M	64.3	94.8	87.9	89.9	89.5	87.7	94.3	79.0	85.9
ELECTRA	7.1e20 (1x)	335M	**68.2**	**96.9**	**89.6**	**91.0**	**90.1**	**90.1**	**95.4**	**83.6**	**88.1**

Table 2: Comparison of large models on the GLUE dev and test sets. BERT dev results are from Clark et al. (2019). See Appendix B for some discussion on GLUE test set comparisons.

3. Large Models Results

Model	Train FLOPs	Params	SQuAD 1.1		SQuAD 2.0	
			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
SpanBERT	7.1e20 (1x)	335M	88.8	**94.6**	85.7	88.7
XLNet-Base	6.6e19 (0.09x)	117M	81.3	–	78.5	–
XLNet	9.6e20 (1.3x)	360M	**89.0**	94.5	86.1	88.8
RoBERTa-100k	6.4e20 (0.90x)	356M	–	94.0	–	87.7
RoBERTa	3.2e21 (4.5x)	356M	88.9	**94.6**	86.5	89.4
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	7.1e20 (1x)	335M	88.7	94.2	**86.9**	**89.6**

Table 3: Results on the SQuAD dev set for single models with no data augmentation or ensembling.

4. Efficiency Analysis

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

Table 4: Compute-efficiency experiments (see text for details).

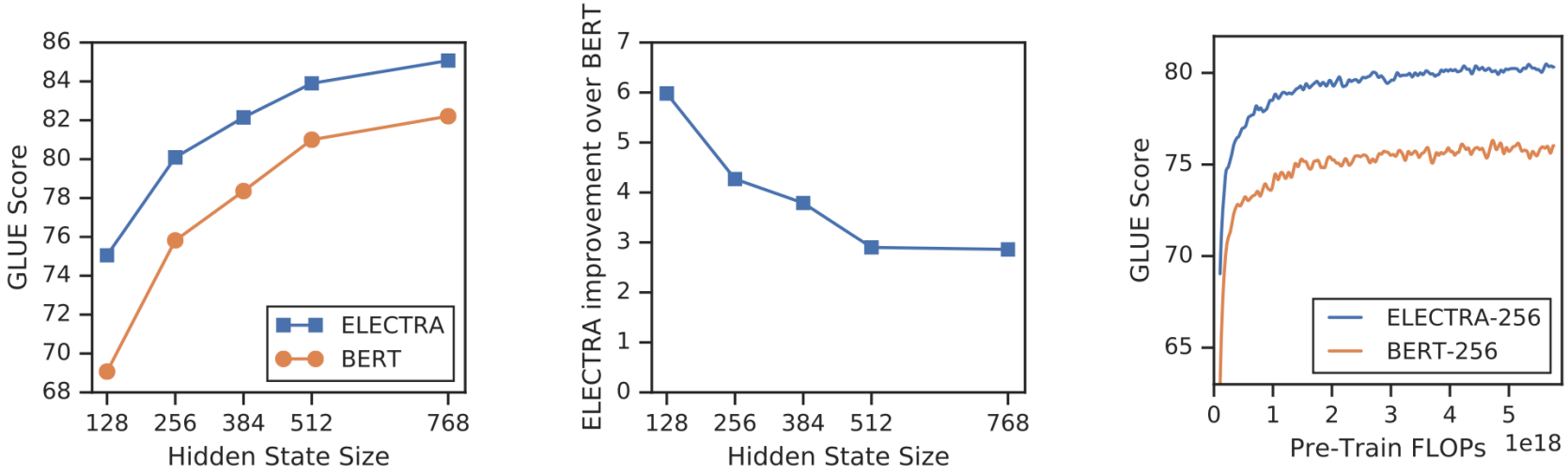


Figure 4: Left and Center: Comparison of BERT and ELECTRA for different model sizes. Right: A small ELECTRA model converges to higher downstream accuracy than BERT, showing the improvement comes from more than just faster training.

Thanks for your attention!

Q&A