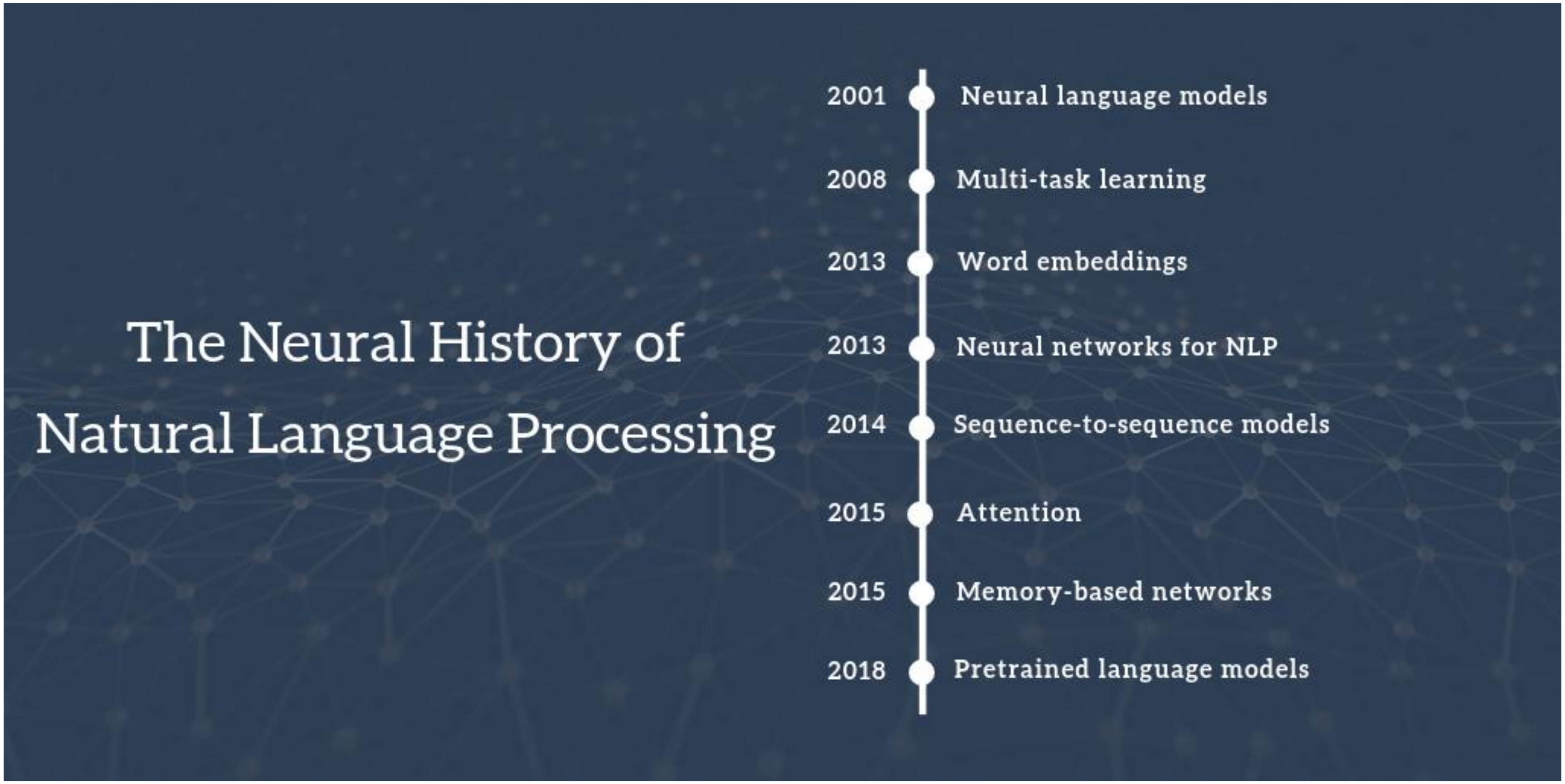
ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS
(ICLR'20 papers)

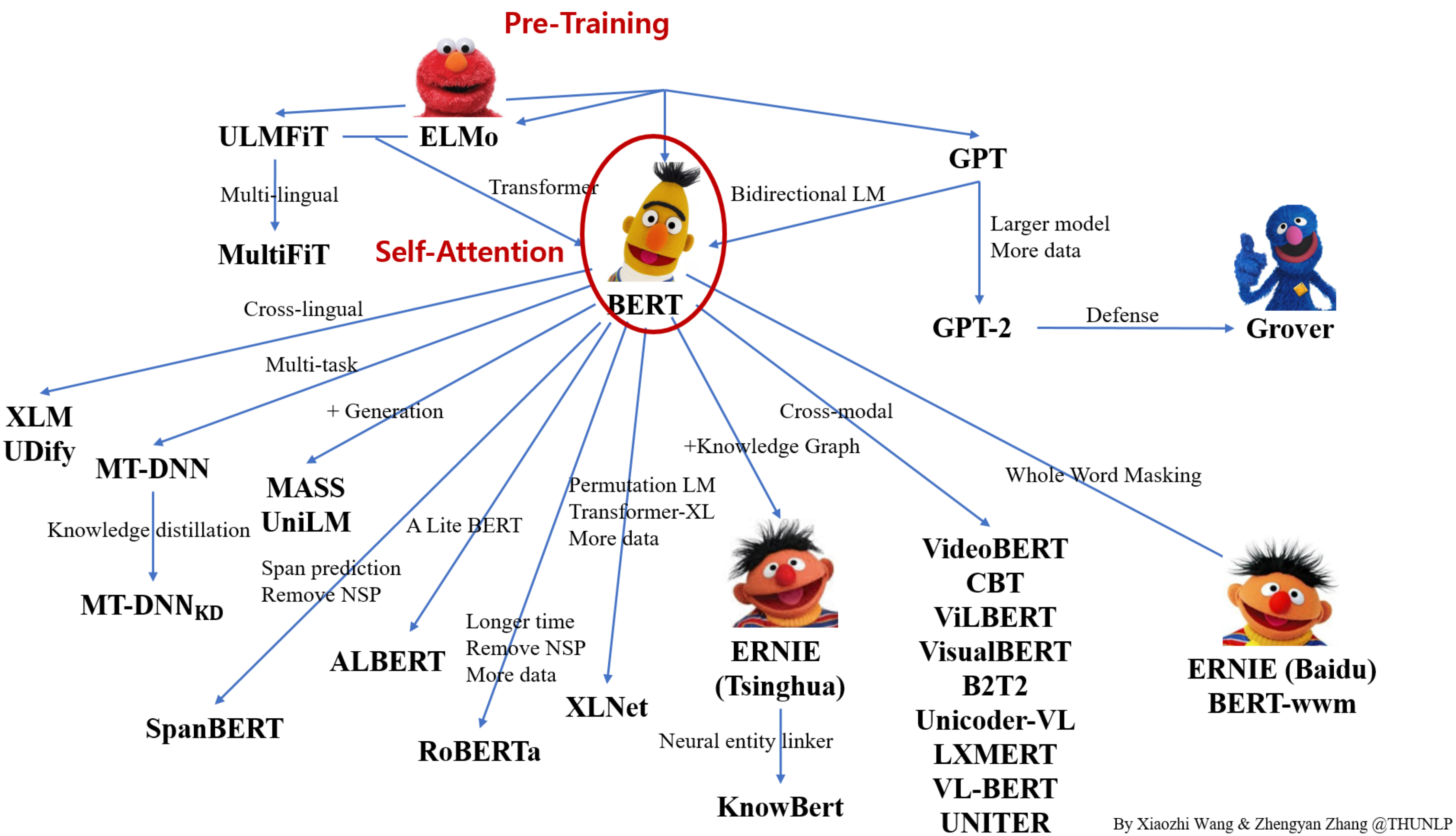
21th Feb, 2022

JongHyeon Kim

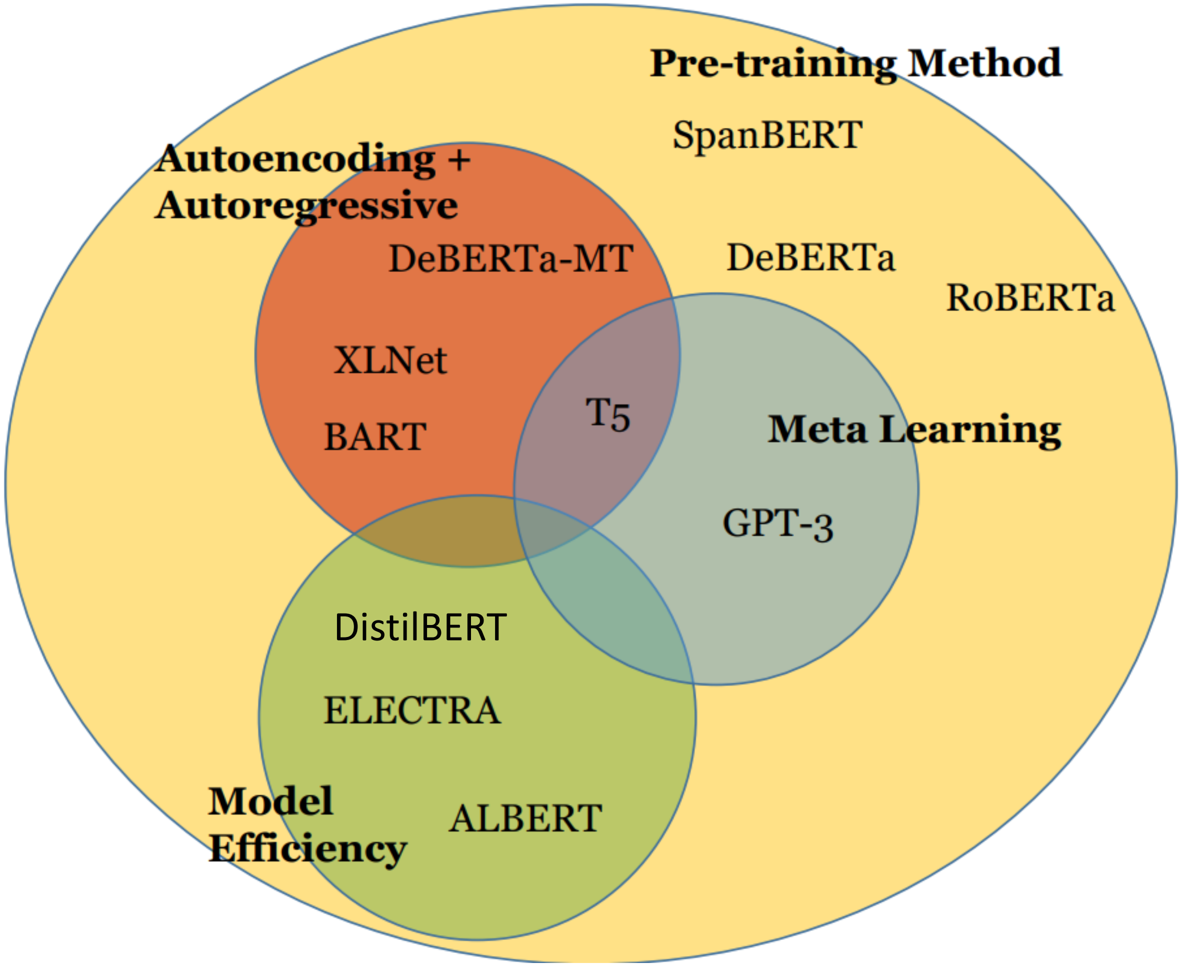
github.com/bellhyeon
bellhyeon@naver.com



Reference: https://ruder.io/a-review-of-the-recent-history-of-nlp/



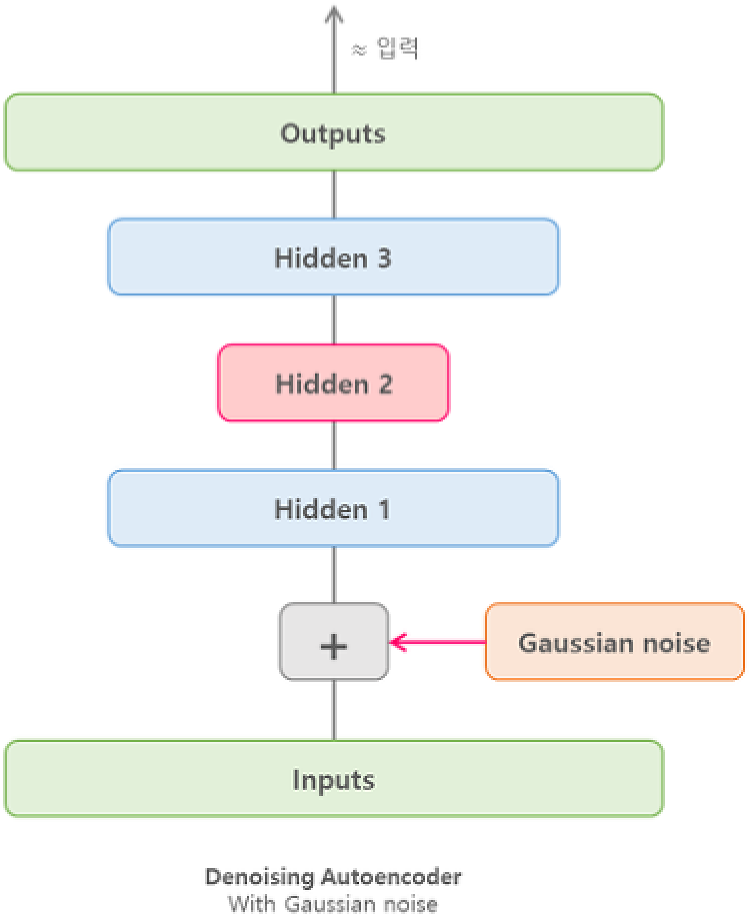
Reference: https://github.com/thunlp/PLMpapers



Reference: http://dmqm.korea.ac.kr/activity/seminar/309

INTRODUCTION

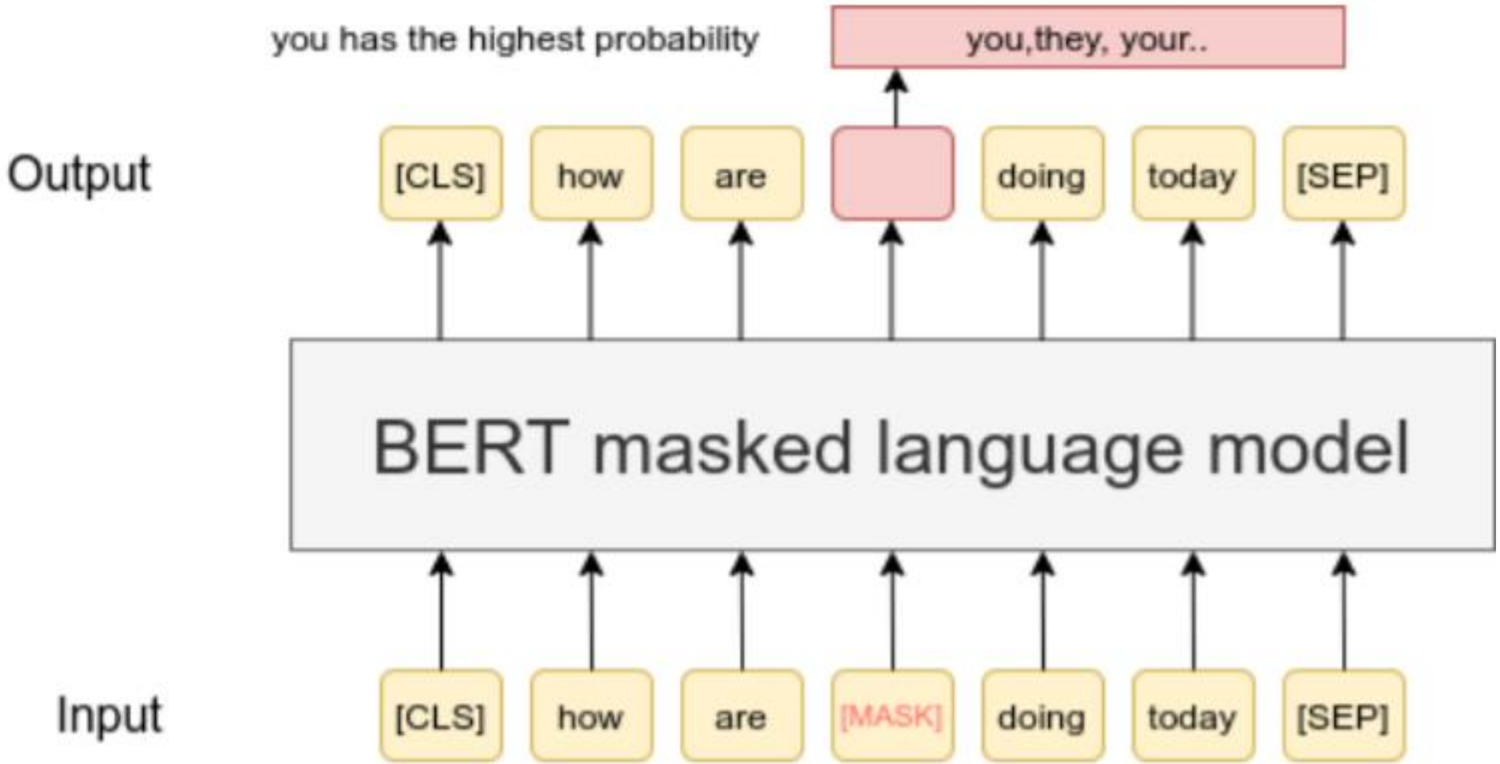
State-Of-The-Art Representation Learning



Reference: https://excelsior-cjh.tistory.com/187

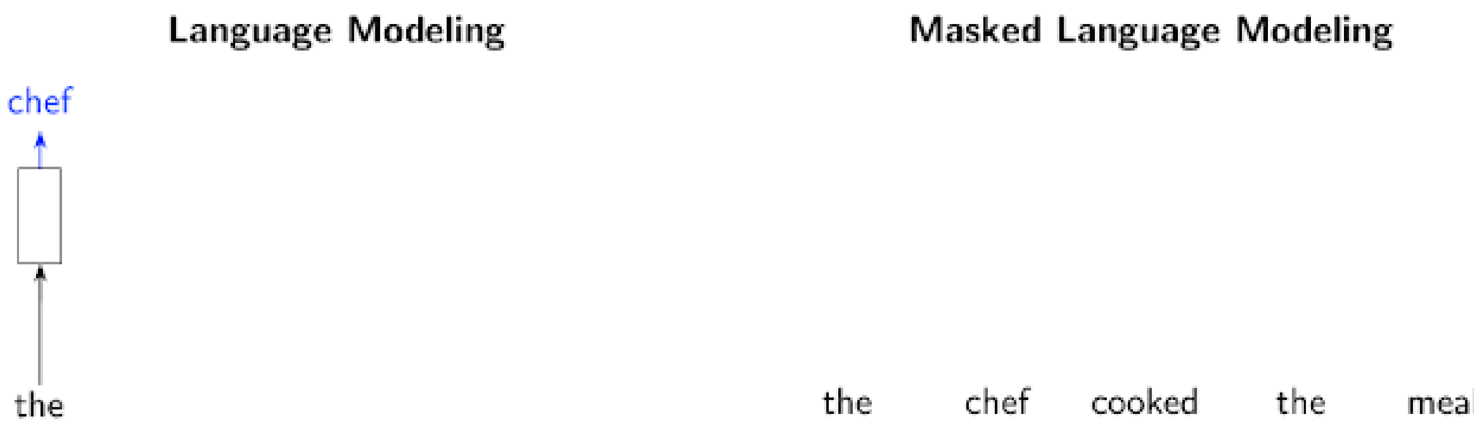
INTRODUCTION

State-Of-The-Art Representation Learning



Reference: https://arxiv.org/pdf/2003.11562.pdf

State-Of-The-Art Representation Learning



Reference: https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html

INTRODUCTION

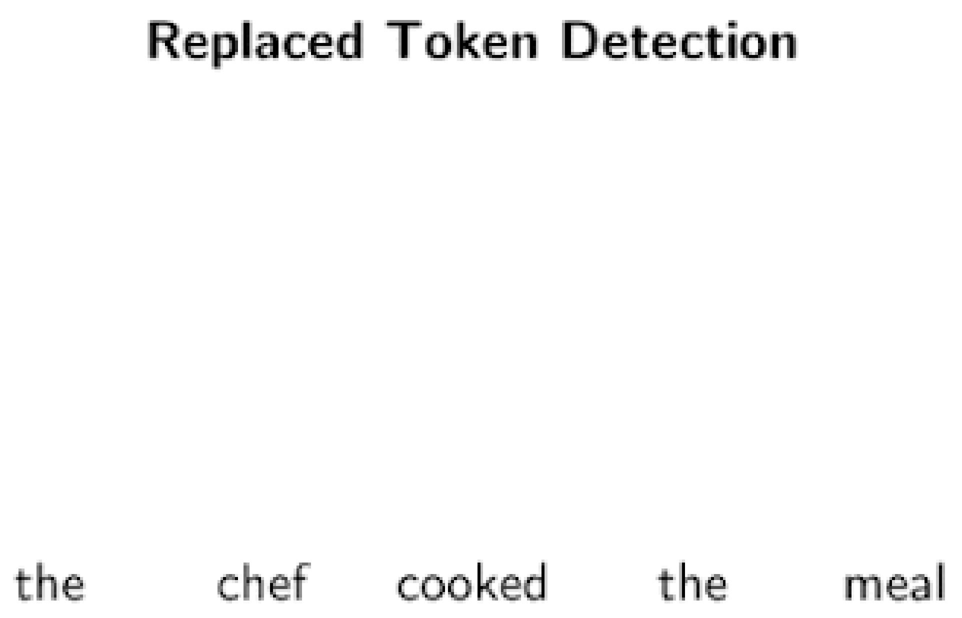
Replaced Token Detection (RTD)

- Masked Language Model trains only 15% of sequences
- Large computation cost
- Network sees [MASK] tokens during pre-training but not when fine-tuning

INTRODUCTION

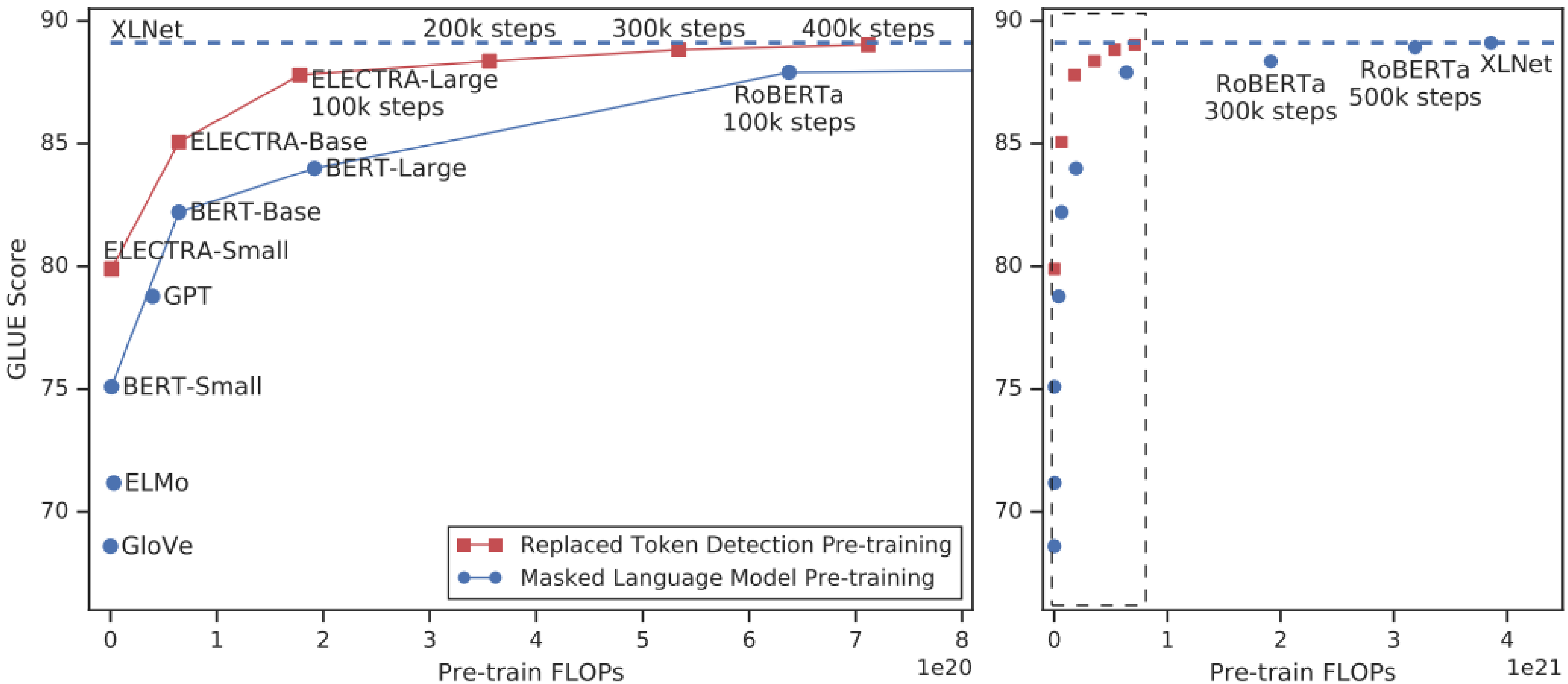
Replaced Token Detection (RTD)

- The model learns to distinguish real input tokens from plausible (but synthetically generated replacements)



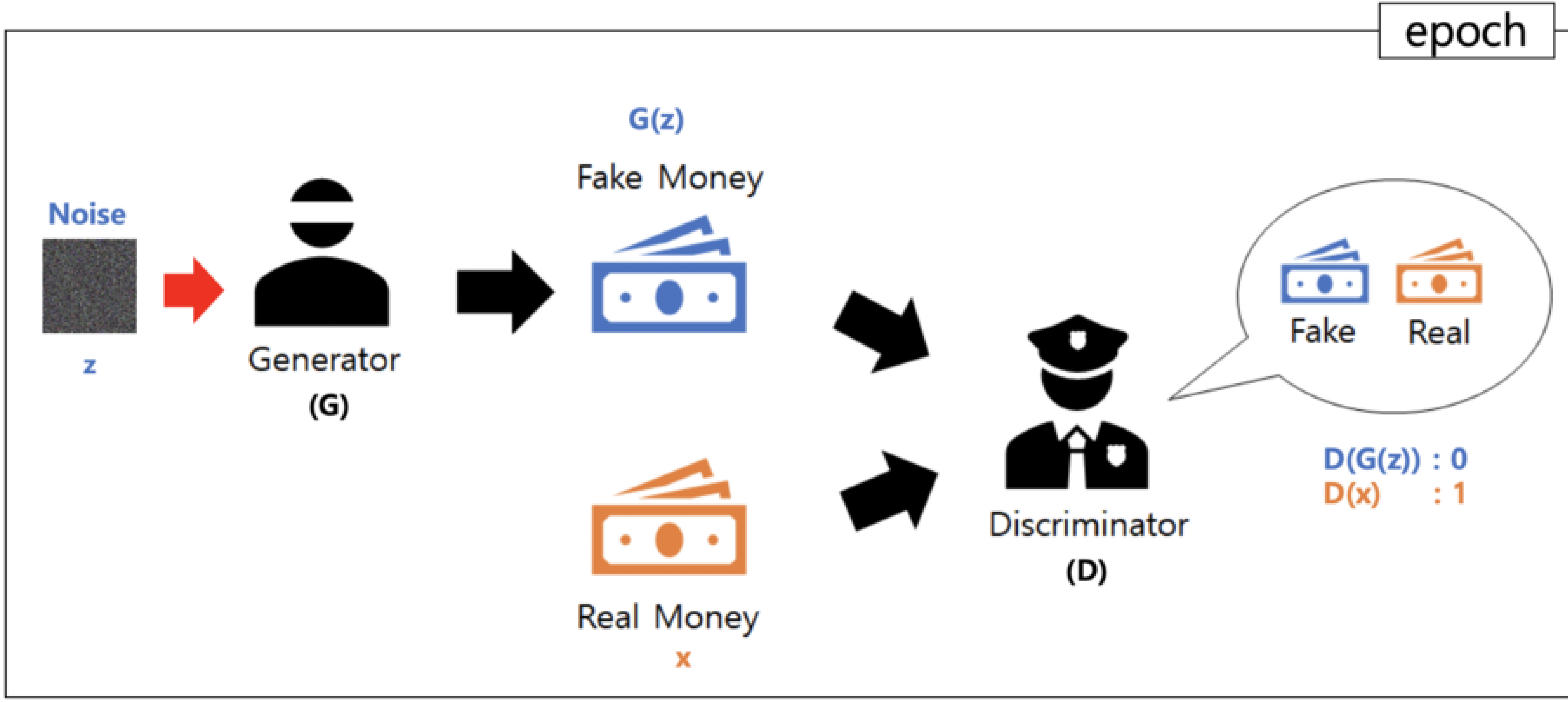
Reference: https://ai.googleblog.com/2020/03/more-efficient-nlp-model-pre-training.html

INTRODUCTION



INTRODUCTION

GAN: Generative Adversarial Networks (Goodfellow et al., 2014)

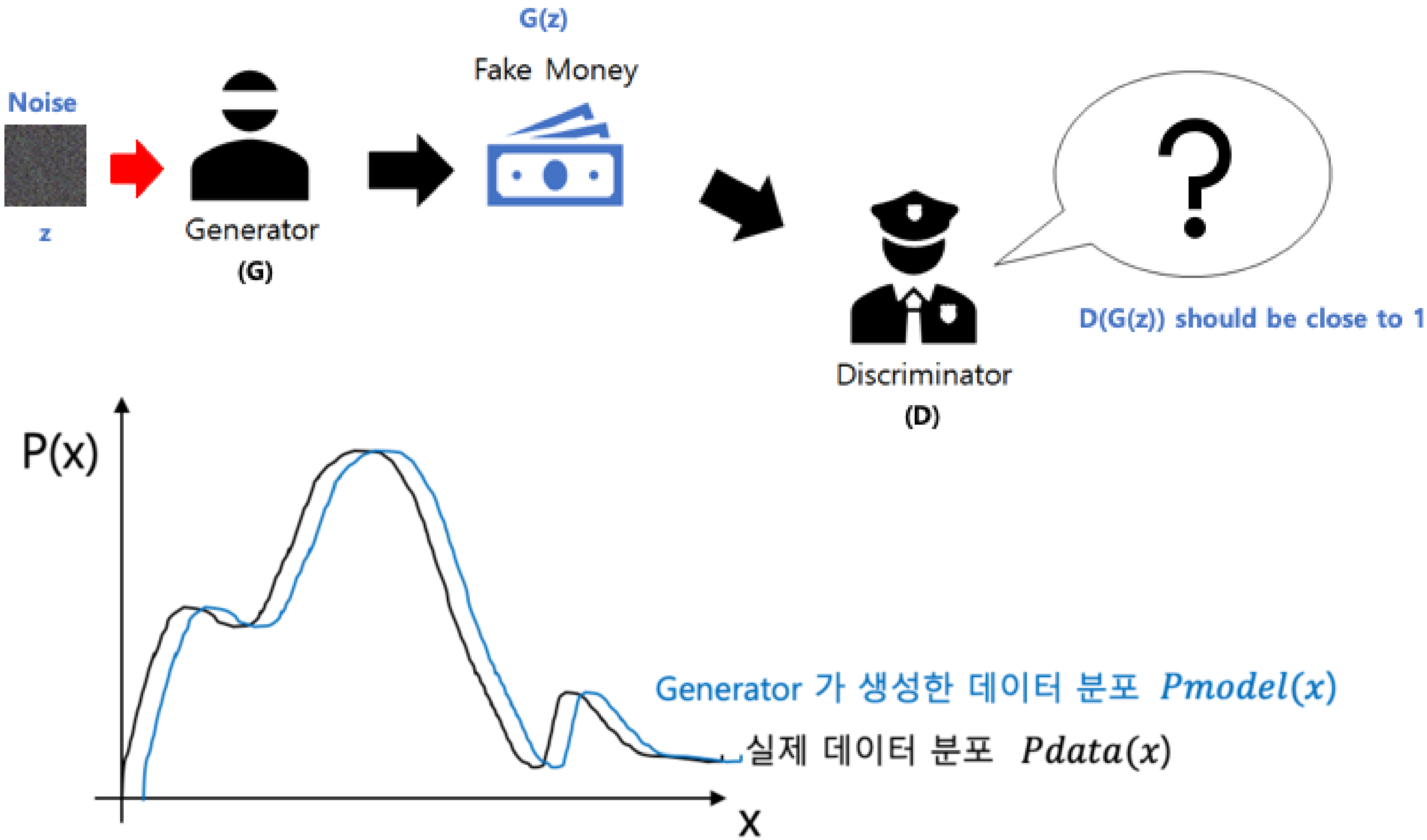


$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim Pdata(x)}[logD(x)] + \mathbb{E}_{z\sim Pz(z)}[\log(1-D(G(z)))]$$

Reference: https://blog.naver.com/euleekwon

INTRODUCTION

GAN: Generative Adversarial Networks (Goodfellow et al., 2014)



Reference: https://blog.naver.com/euleekwon

GAN: Generative Adversarial Networks (Goodfellow et al., 2014)

Sample x from real data distribution

Sample latent code z from Gaussian distribution

$$\min_{G} \boxed{\max_{D} V(D,G)} = \mathbb{E}_{\boxed{x\sim Pdata(x)}}[log\boxed{D(x)}] + \mathbb{E}_{\boxed{z\sim Pz(z)}}[\log(1-\boxed{D\big(G(z)\big)})]$$

D should maximize V(D,G)

Maximum when D(x) = 1

Maximum when D(G(z)) = 0

Reference: https://blog.naver.com/euleekwon

INTRODUCTION

GAN: Generative Adversarial Networks (Goodfellow et al., 2014)

$$\boxed{\min_{G}}\,\max_{D} V(D,G) = \cancel{\mathbb{E}_{x\sim Pdata(x)}[\log D(x)]} + \mathbb{E}_{z\sim Pz(z)}\left[\log\left(1-\boxed{D\big(G(z)\big)}\right)\right]$$

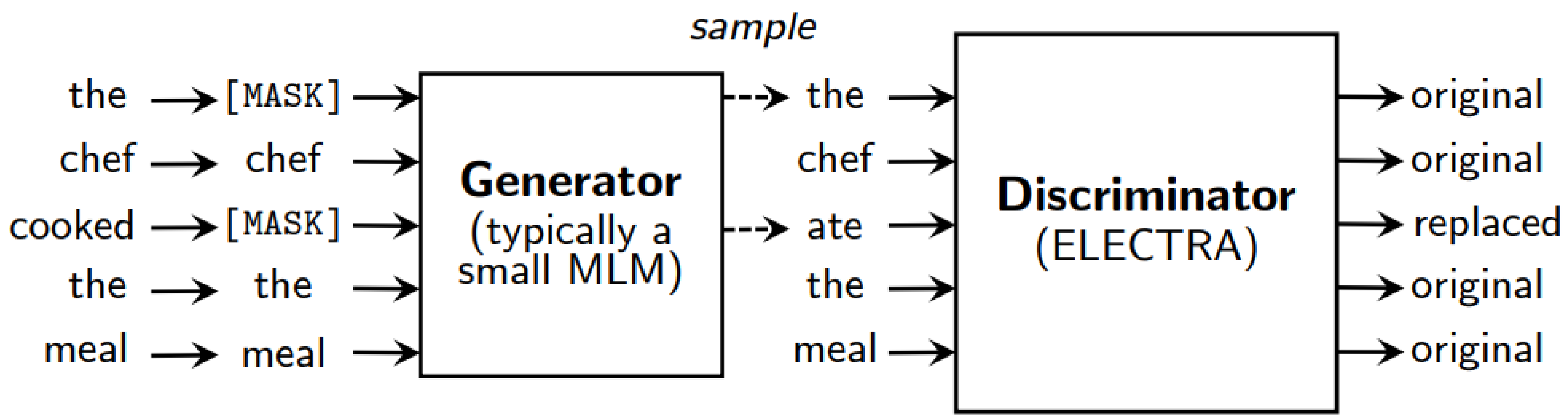
G should minimize V(D,G)

G doesn't care

Minimum when D(G(z)) = 1

Reference: https://blog.naver.com/euleekwon

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately)



METHOD

Generator

- The generator can be any model that produces an output distribution over tokens
- ELECTRA uses a small Masked Language Model
- Train with maximum likelihood estimation
- In downstream tasks, fine-tune only discriminator

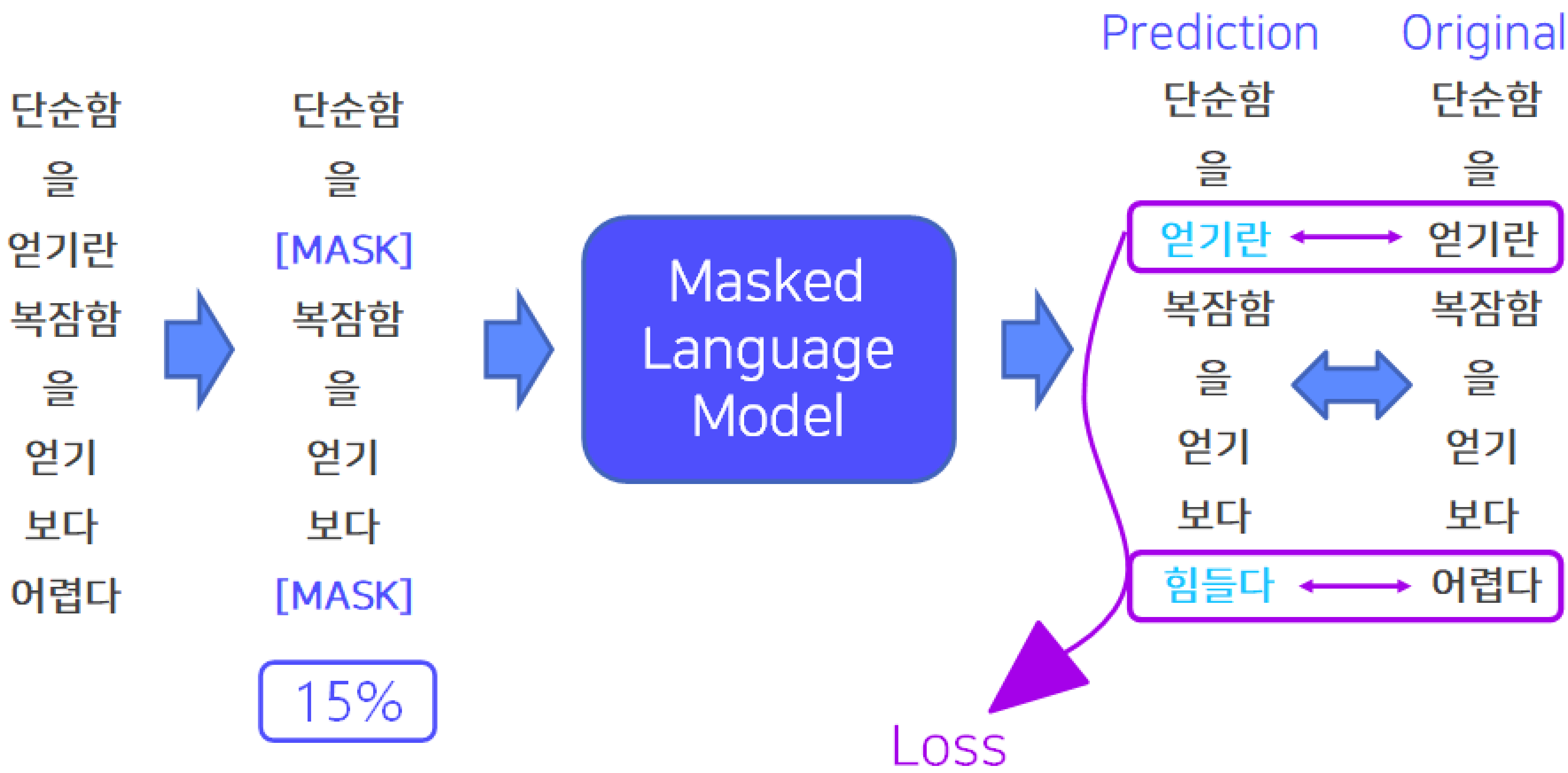
METHOD

Generator

- input $x = [x_1, x_2 \cdots, x_n]$
  contextualized vector representations $h(x) = [h_1, \cdots, h_n]$
- MLM select a random set of positions to mask out $[m_1, \cdots, m_k]$
  m_i ~ unif $\{1, n\}$ for i = 1 to k. usually k = $0.15n$ (15% mask)
- x^{masked} = REPLACE(x, m, [MASK])
- The generator outputs a probability for generating particular token x_t with a softmax layer
- $pG\left(x_t \middle| x^{masked}\right) = \exp(e(x_t)^T h_G\left(x^{masked}\right)_t) \,/\, \sum_{x'} \exp(e(x')^T h_G(x^{masked})_t)$

METHOD

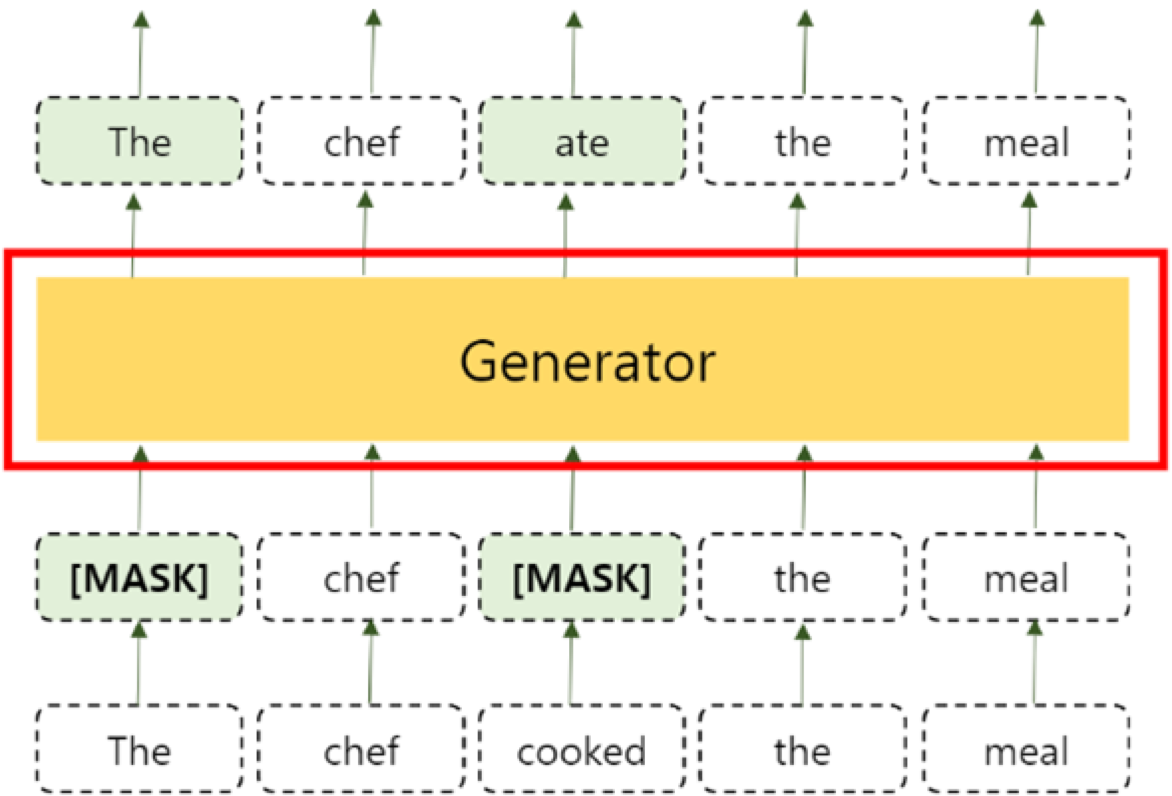
Generator



Reference: https://ydy8989.github.io/2021-07-11-electra/

METHOD

Generator



Reference: https://github.com/jiphyeonjeon/season2

METHOD

Discriminator

- Make input sampled from generator
  [MASK] $\rightarrow pG\left(x_t \middle| x^{masked}\right)$ (corrupt)

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

- $\hat{x} \sim pG\left(x_i \middle| x^{masked}\right)$ for $i \in m$

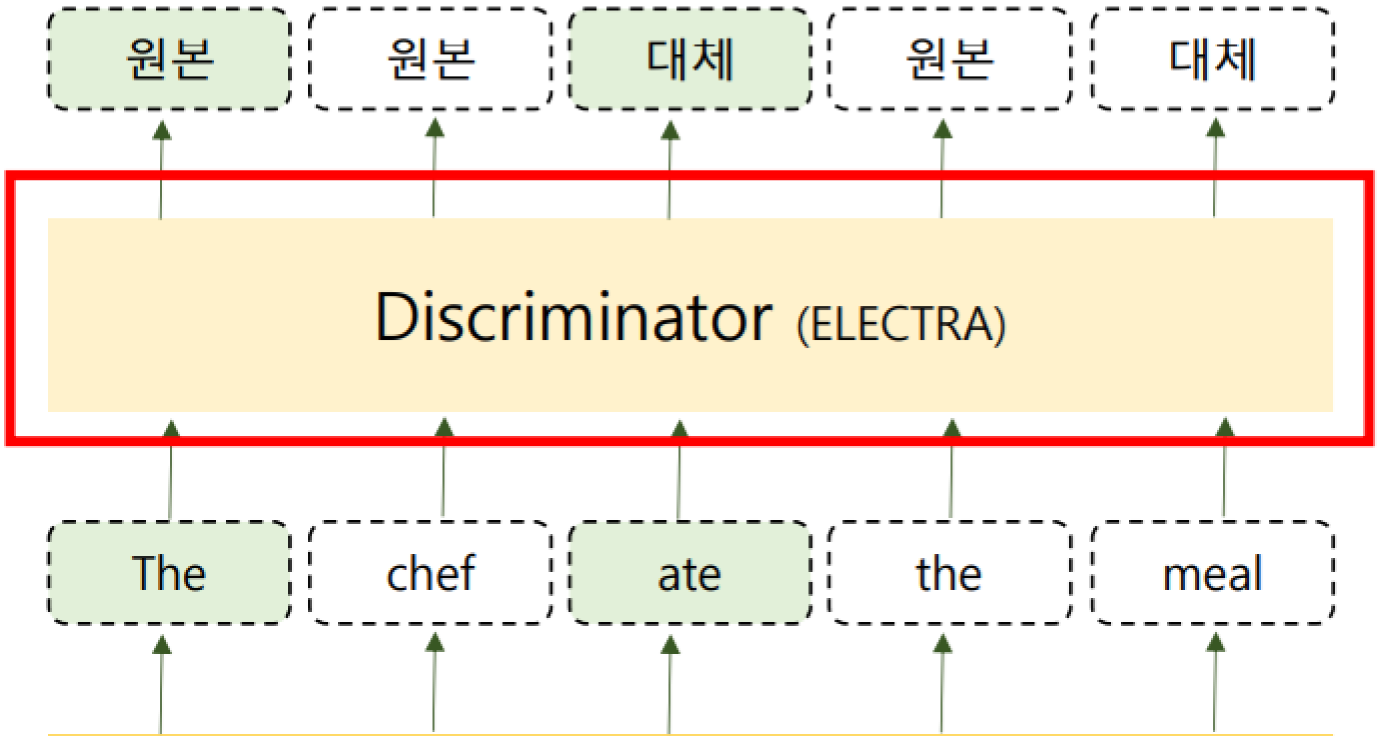
- The discriminator is trained to distinguish tokens in the data from tokens that have been replaced by generator samples

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

- Binary Classification
  - Original(1): Same token as original
  - Replaced(0): Different token as original

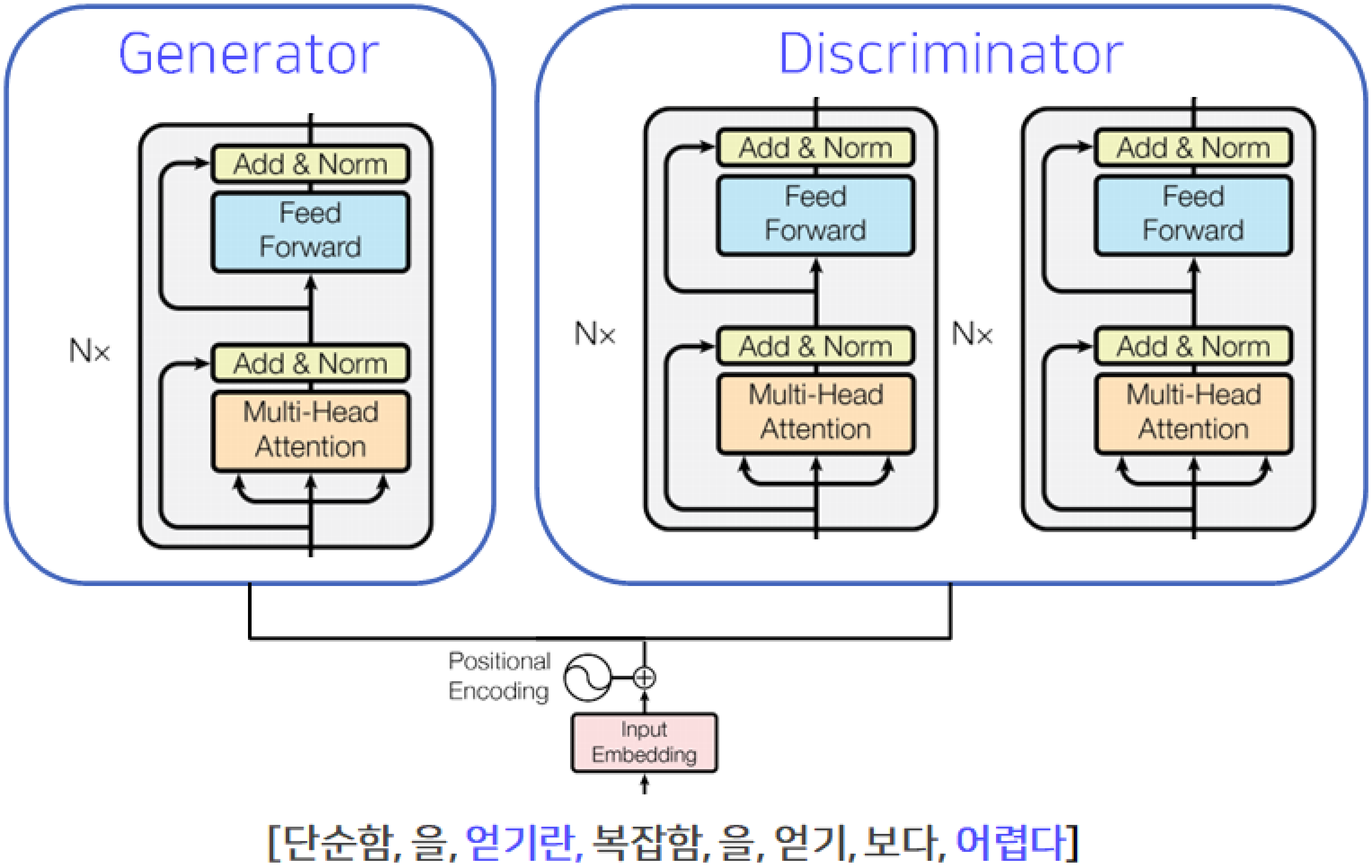
METHOD

Discriminator



METHOD

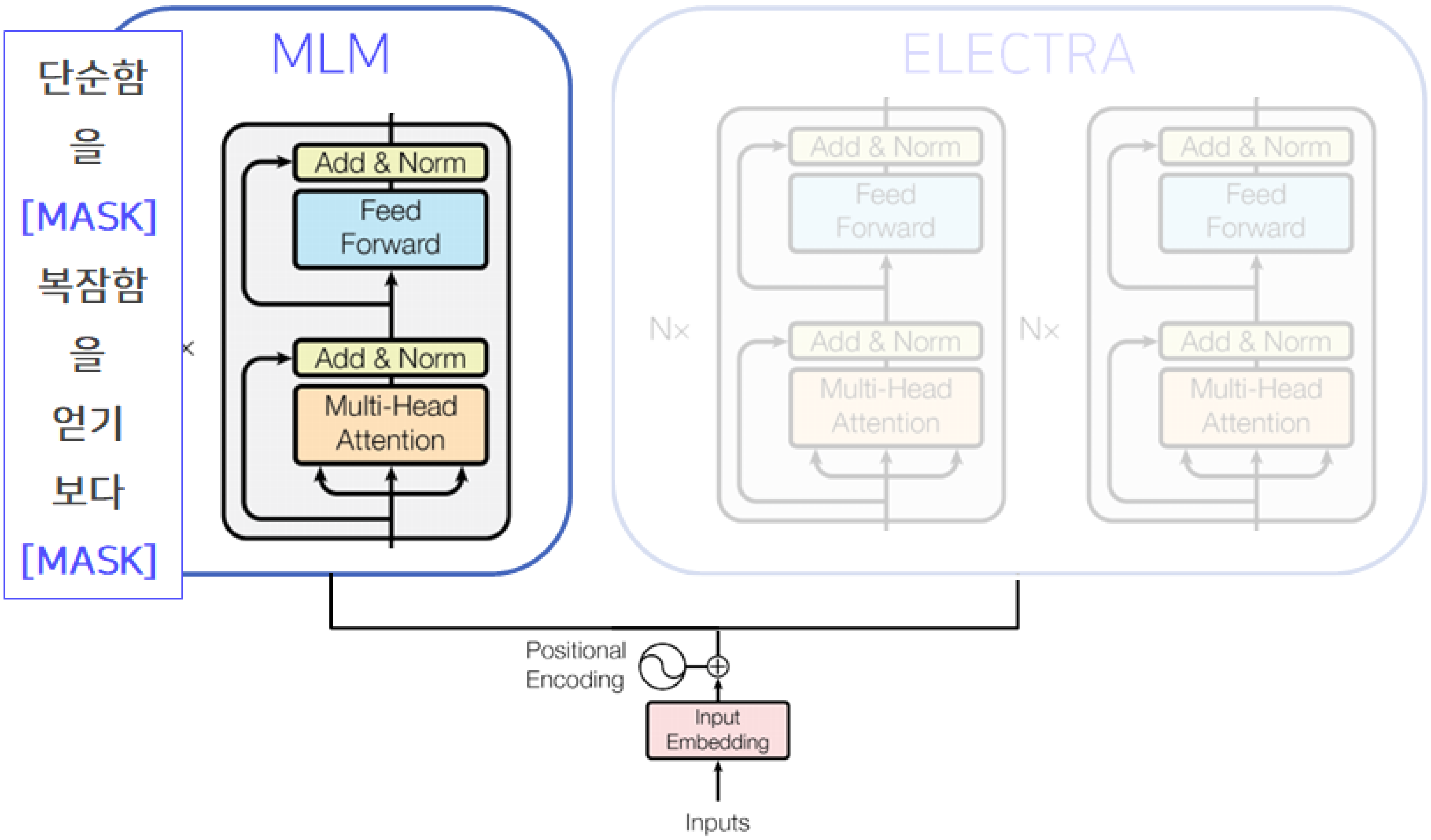
Training Process



Reference: https://ydy8989.github.io/2021-07-11-electra/

METHOD

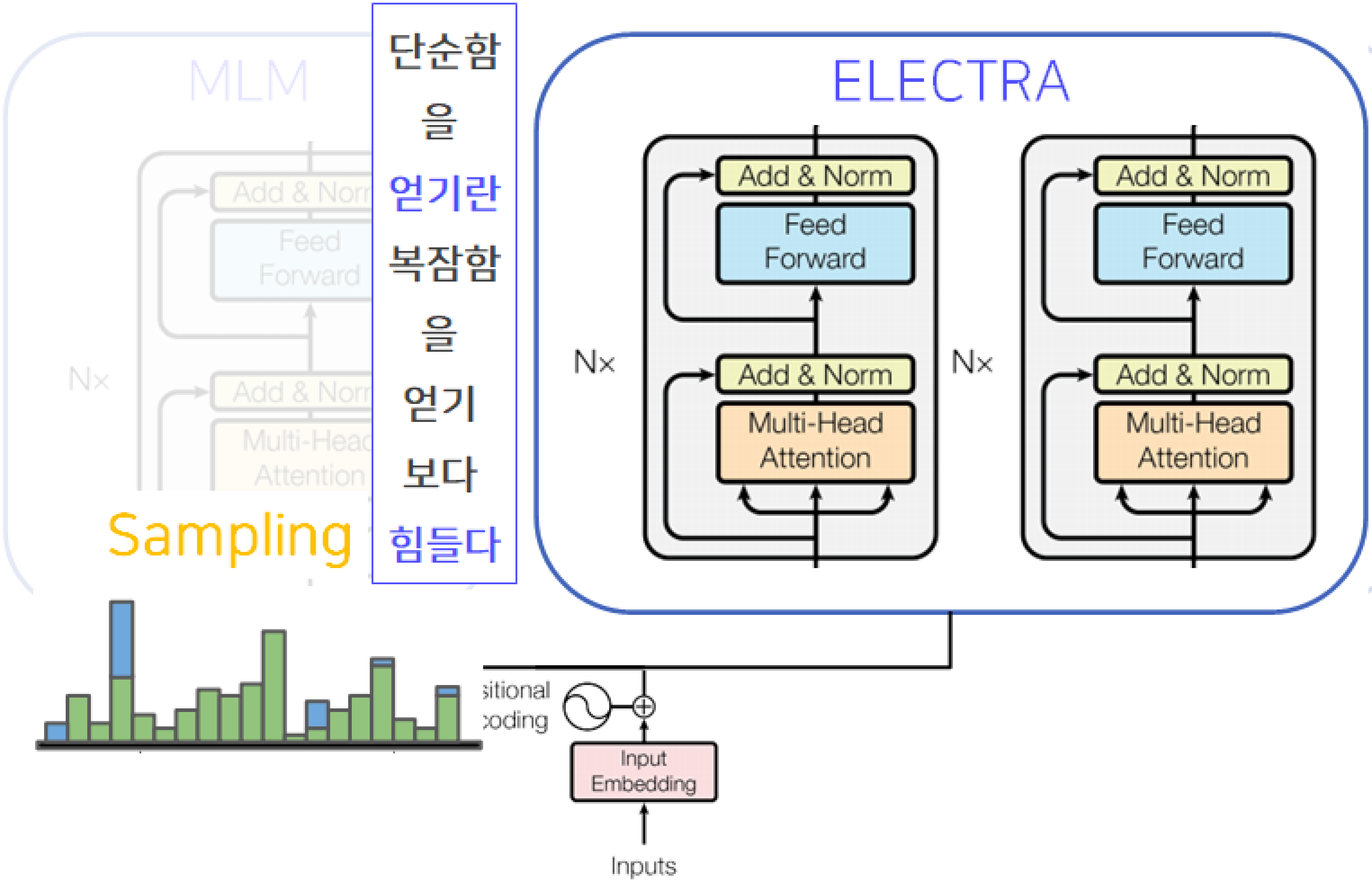
Training Process



Reference: https://ydy8989.github.io/2021-07-11-electra/

METHOD

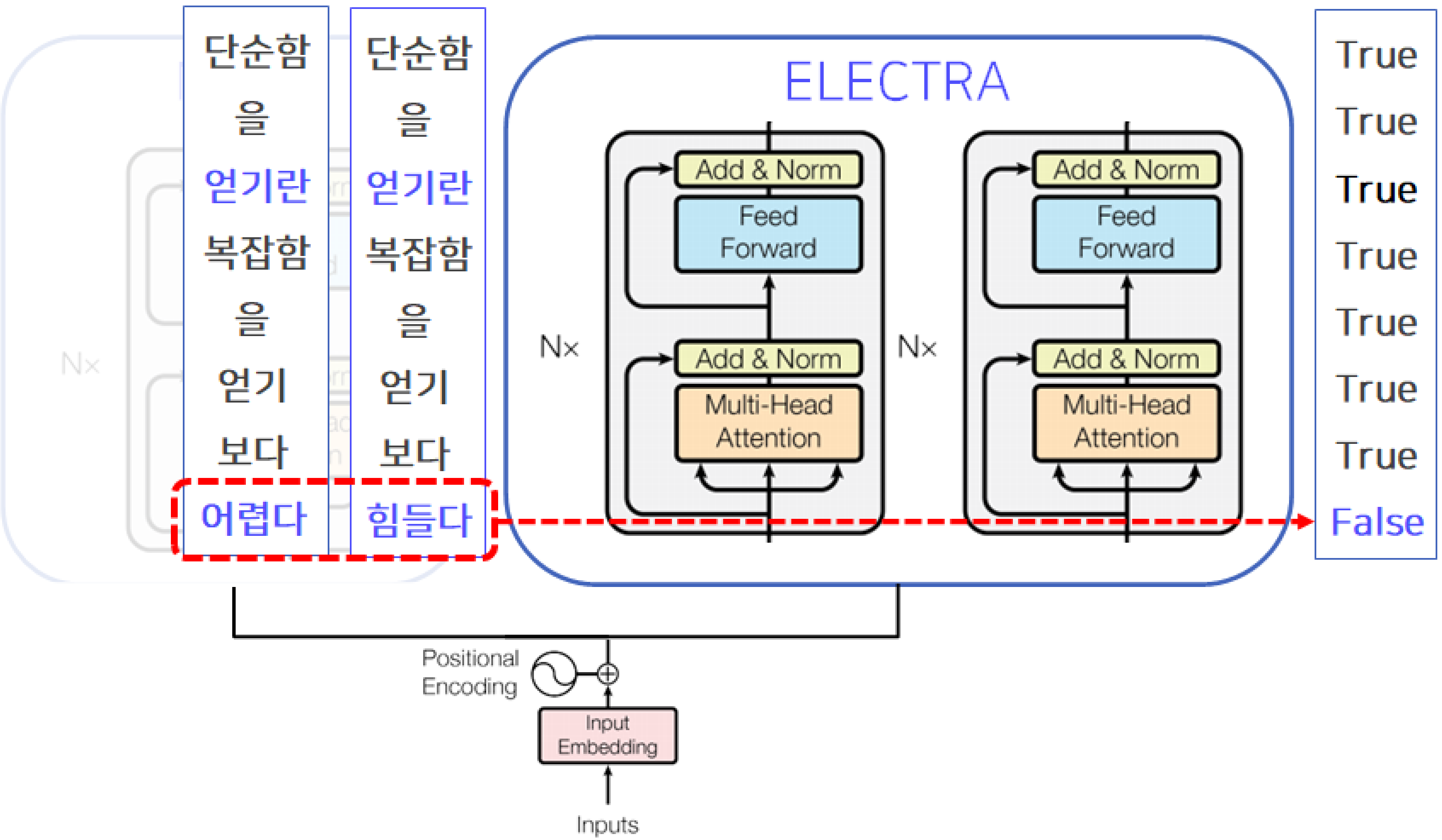
Training Process



Reference: https://ydy8989.github.io/2021-07-11-electra/

METHOD

Training Process



Reference: https://ydy8989.github.io/2021-07-11-electra/

Objective(Loss) Functions

- Generator G
  $\mathcal{L}_{\mathrm{MLM}}(x, \theta_G) = \mathbb{E}(\sum_{i \in m} -\log p_G(x_i | x^{masked}))$

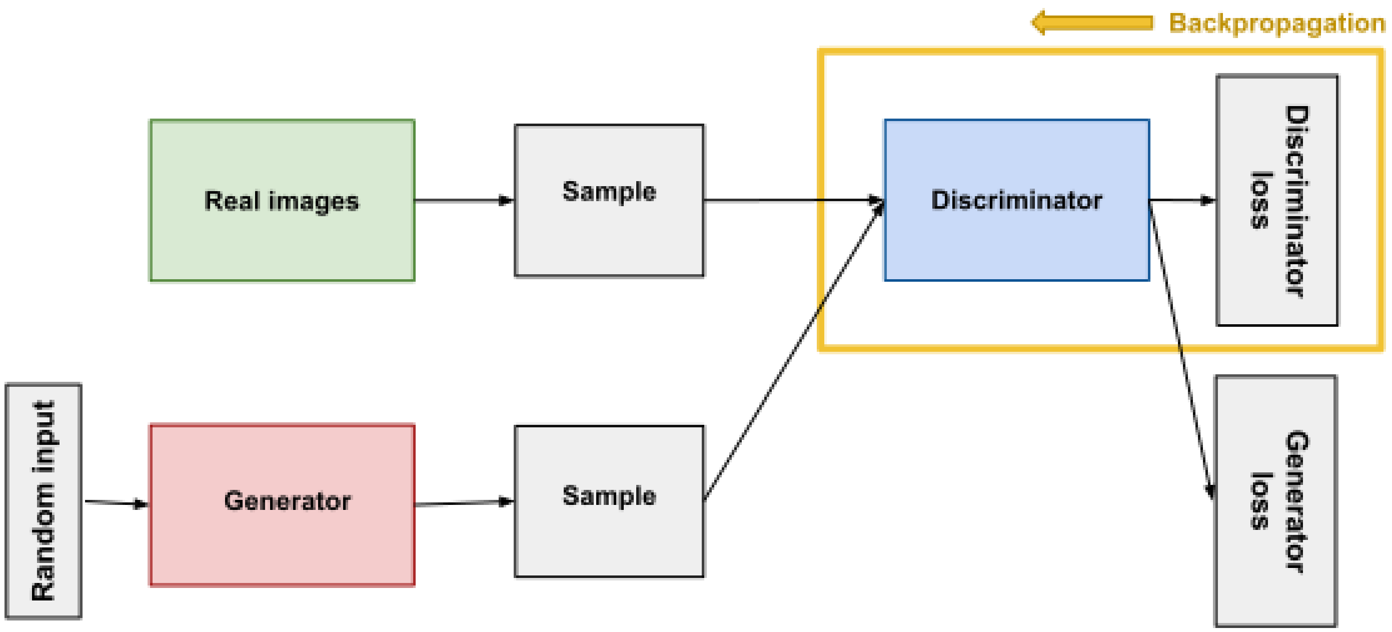
- Discriminator D
  $\mathcal{L}_{\mathrm{Disc}}(x, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\left(x_t^{corrupt} = x_t\right) \log D(x^{corrput}, t) - \left(x_t^{corrupt} \neq x_t\right) \log(1 - D(x^{corrput}, t))\right)$

- $\min_{\theta_G, \theta_D} \sum_{x \in \chi} \mathcal{L}_{\mathrm{MLM}}(x, \theta_G) + \lambda \mathcal{L}_{\mathrm{Disc}}(x, \theta_D)$
  χ: large corpus of raw text

METHOD

Difference with GAN

- If the generator happens to generate the correct token, that token is considered **“real”** instead **“fake”**
  → to moderately improve results on downstream tasks

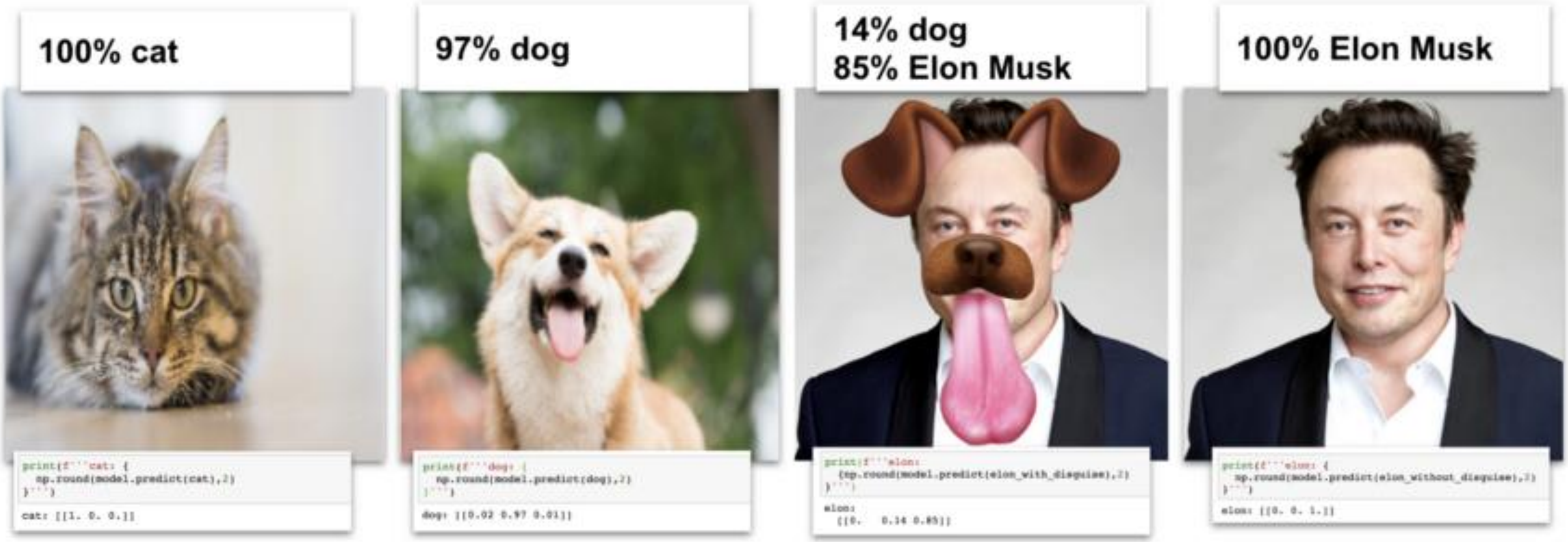


Reference: https://developers.google.com/machine-learning/gan/generator

METHOD

Difference with GAN

- The generator is trained with maximum likelihood, not adversarially
  → impossible to backpropagate through sampling from generator



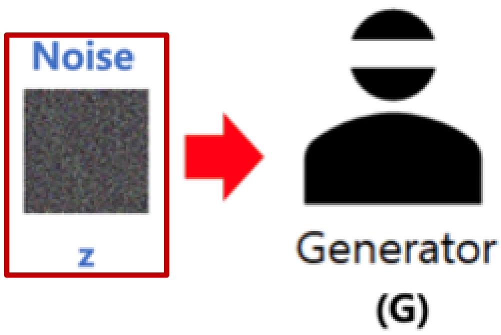
90% Catt 100% Cat ??

Reference: https://towardsdatascience.com/cat-dog-or-elon-musk-145658489730

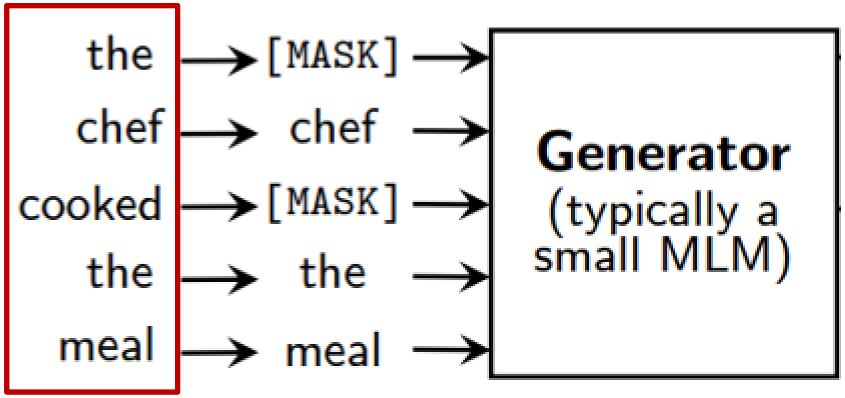
METHOD

Difference with GAN

- Electra do not supply the generator with a noise vector as input



GAN



ELECTRA

MODEL EXTENSIONS

Smaller Generators

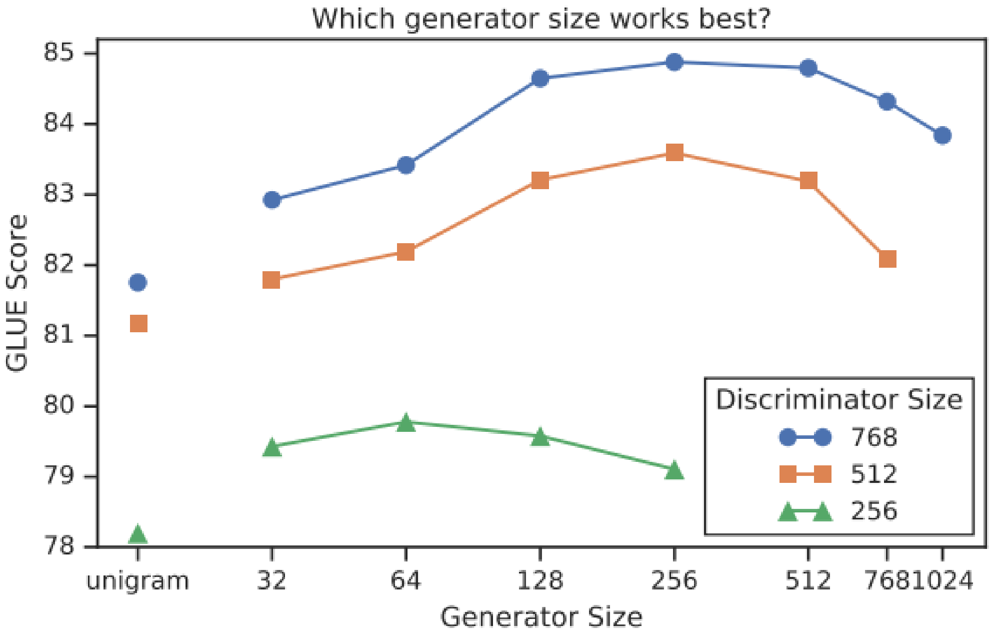
- If the generator and discriminator are the same size, training ELECTRA would take around twice as much compute per step as training only MLM
  → ELECTRA is generator-discriminator structure

- Suggests using a smaller generator
  → by decreasing layer size (hidden layer size, FFN size, attention heads)

MODEL EXTENSIONS

Smaller Generators

- Worked best with generators ¼ - ½ the size of discriminator
- Having too strong of generator may pose a too-challenging task for discriminator
- Discriminator may have to use many of its parameters modeling the generator rather than the actual data distribution.



MODEL EXTENSIONS

Weight Sharing

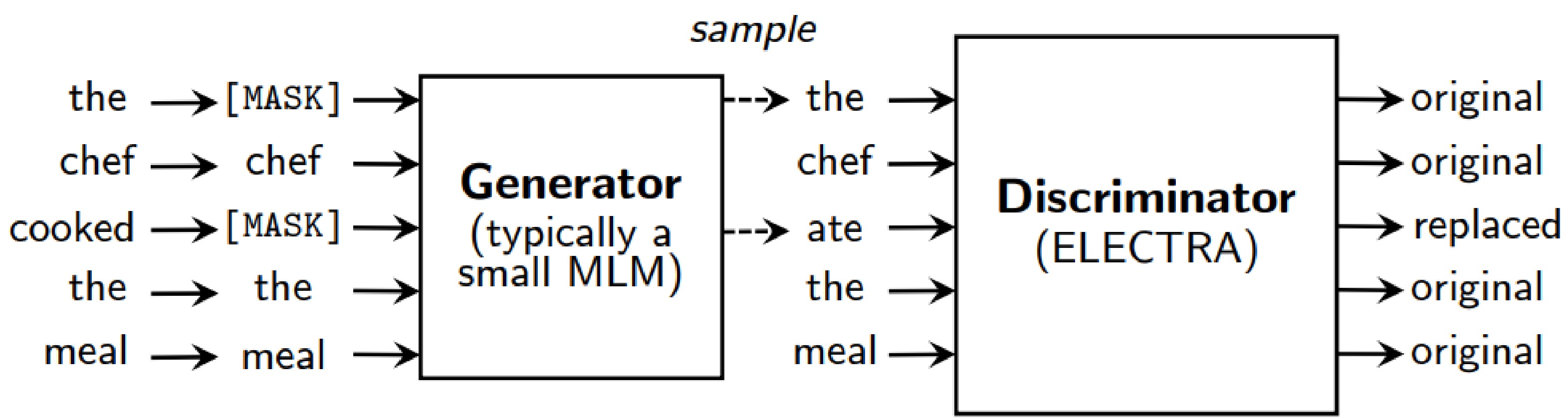
- Tying all weights require same size of generator and discriminator
- Small generator size to be more efficient
- ELECTRA shares weight only embedding layers
  (the token + positional embeddings)
- GLUE scores are 83.6 for no weight tying, 84.3 for tying token embeddings, 84.4 for tying all weights

MODEL EXTENSIONS

Weight Sharing

- Discriminator only updates tokens that are present in the input or are sampled by generator

- Generator's softmax over the vocabulary densly updates all token embeddings



MODEL EXTENSIONS

Training Algorithms

- Two-Stage Method
  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

- Without initialize weights of discriminator, would sometimes fail to learn at all beyond the majority class
  → generator started so far ahead of the discriminator

- Initialize weights provide a curriculum for the discriminator where the generator starts off weak but gets better throughout training.

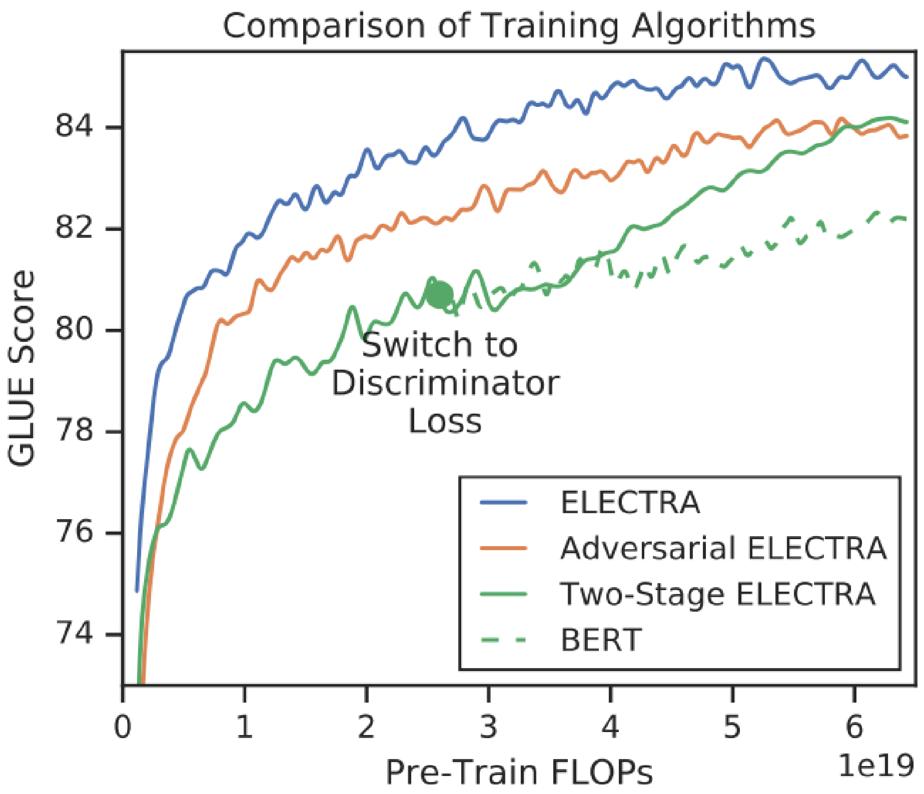
MODEL EXTENSIONS

Training Algorithms

- Adversarial Method like GAN
  using reinforcement learning to accommodate the discrete operations of sampling from the generator

- Adversarial training to underperform maximum-likelihood training
  1. Worse at Masked Language Modeling: 58% accuracy
     (Maximum Likelihood Estimation is 65% accuracy)
     → poor sample efficiency when working in the large action space of generating text
  2. Low entropy output distribution: adversarially trained generator produces a low-entropy output distribution where most of the probability mass is on a single token, which means there is not much diversity in the generator samples

Training Algorithms



MODEL EXTENSIONS

Small Models

- Changed some BERT-Base hyperparameters
  - Sequence length (512 → 128)
  - Batch size (256 → 128)
  - Hidden dims (768 → 256)
  - Token embedding (768 → 128)

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

MODEL EXTENSIONS

Large Models

- ELECTRA-400K took less than ¼ of the compute RoBERTa and XLNet

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**

Model	Train FLOPs	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI	Avg.*	Score
BERT	1.9e20 (0.06x)	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	65.1	79.8	80.5
RoBERTa	3.2e21 (1.02x)	67.8	96.7	89.8	91.9	90.2	90.8	95.4	88.2	89.0	88.1	88.1
ALBERT	3.1e22 (10x)	69.1	**97.1**	**91.2**	92.0	90.5	**91.3**	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	**97.1**	90.5	**92.6**	90.4	90.9	–	88.5	**92.5**	89.1	–
ELECTRA	3.1e21 (1x)	**71.7**	**97.1**	90.7	92.5	**90.8**	**91.3**	**95.8**	**89.8**	**92.5**	**89.5**	**89.4**

MODEL EXTENSIONS

Large Models

- Also improves better results on the SQuAD

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**

Efficiency Analysis

- ELECTRA 15%: Except the discriminator loss only comes from the 15% of the tokens that were masked out of the input
- Replace MLM: Similar to MLM, gives input to discriminator with [MASK] tokens that are replaced tokens from a generator model
- All-Tokens MLM: Like in Replace MLM, masked tokens are replaced with generator samples

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

CONCLUSION

- Proposed replaced token detection (RTD), a new self-supervised task for language representation learning
- The key idea is training a text encoder to distinguish input tokens from high-quality negative samples produced by an small generator network
- Compared to Masked Language Modeling, more compute-efficient and results in better performance on downstream tasks
- Hope will make developing and applying pre-trained text encoders more accessible to researchers and practitioners with less access to computing resources
- Also hope more future work on NLP pre-training will consider efficiency as well as absolute performance, and follow efforting in reporting compute usage and parameter counts along with evaluation metrics

APPENDIX

Pre-training Details

- Mostly use the same hyperparameters as BERT
- Set λ, the weight for the discriminator objective in the loss to 50
- Used dynamic token masking with the masked positions decided on-the-fly instead of during preprocessing
- Did not use the next sentence prediction (NSP) objective proposed in the original BERT paper
  $\rightarrow$ in recent work (XLNet, Roberta) has suggested it does not improve scores
- Used a higher mask percent (25%) on ELECTRA-Large model

APPENDIX

Pre-training Details

Hyperparameter	Small	Base	Large
Number of layers	12	12	24
Hidden Size	256	768	1024
FFN inner hidden size	1024	3072	4096
Attention heads	4	12	16
Attention head size	64	64	64
Embedding Size	128	768	1024
Generator Size (multiplier for hidden-size, FFN-size, and num-attention-heads)	1/4	1/3	1/4
Mask percent	15	15	25
Learning Rate Decay	Linear	Linear	Linear
Warmup steps	10000	10000	10000
Learning Rate	5e-4	2e-4	2e-4
Adam ϵ	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999
Attention Dropout	0.1	0.1	0.1
Dropout	0.1	0.1	0.1
Weight Decay	0.01	0.01	0.01
Batch Size	128	256	2048
Train Steps (BERT/ELECTRA)	1.45M/1M	1M/766K	464K/400K

Q&A 질문과 답변

Thank you!