DEVIEW 2020

효율적인 BERT Inference

김규완 NAVER Clova AI

CONTENTS

1. Pretrained Language Models

1.1 자연어 처리에서의 전이 학습

언어 모델을 pre-training 후 여러 downstream task로 fine-tuning

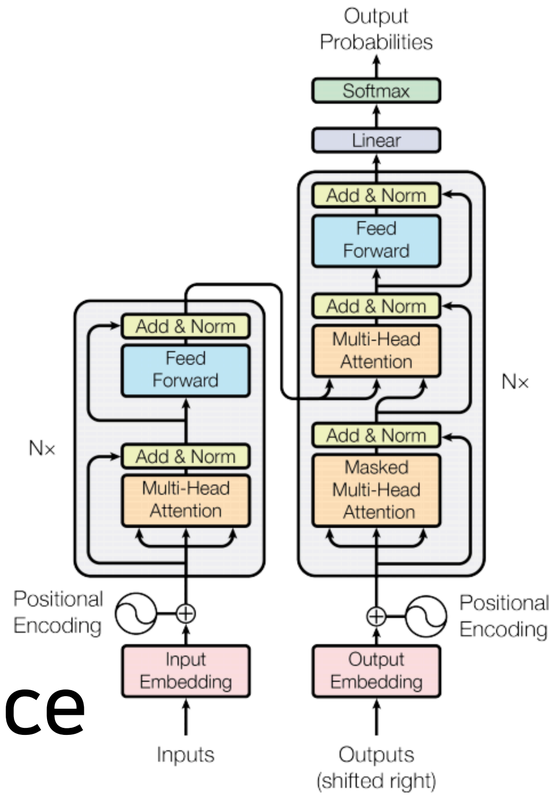
대량의 문서에서 자기 지도 학습을 통해 좋은 representation 학습

Transformer 구조의 모델 사용

1.2 Transformer

Self-attention과 feed-forward layer들로 구성

입력 문장 → token들의 sequence
→ token과 position embedding들의 합
→ 각 transformer layer마다 hidden vector들
→ 최종 contextualize된 representation들의 sequence



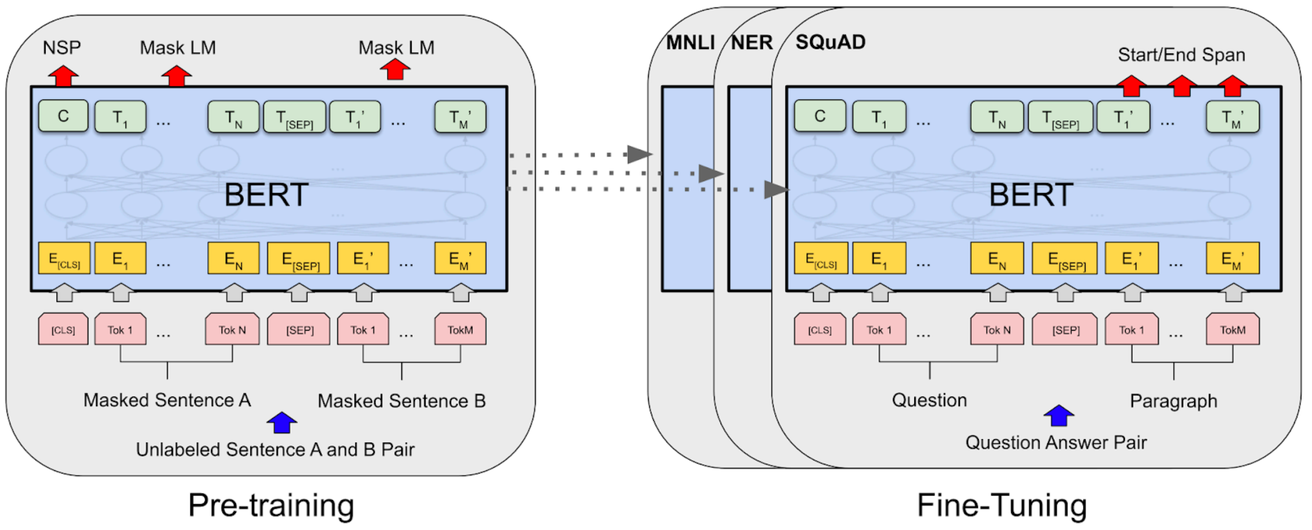
Attention is All You Need (Vaswani et al., NIPS 2017)

1.3 BERT

DEVIEW 2020

Pre-training: masked LM + next sentence prediction

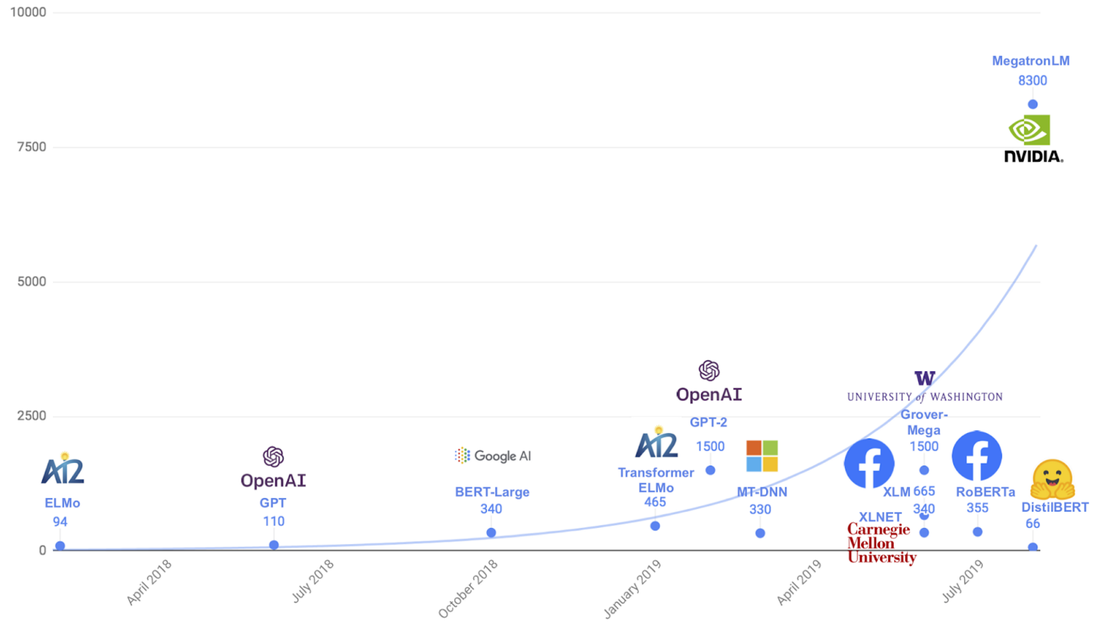
Fine-tuning: sentence-level & token-level tasks

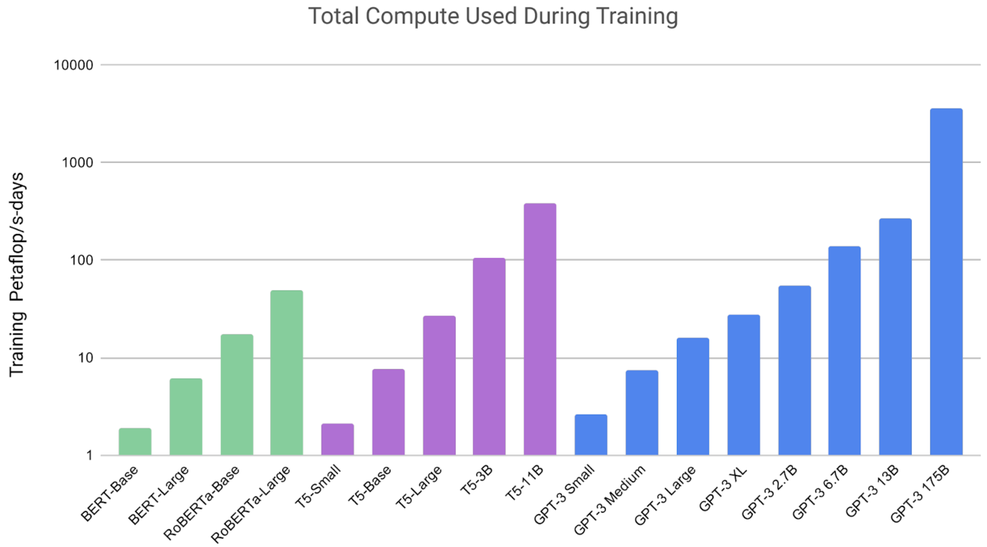


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019)

1.4 모델을 키울수록 더 좋은 정확도

DEVIEW 2020





1.5 하지만, 고려해야할 사항들

DEVIEW 2020

시간

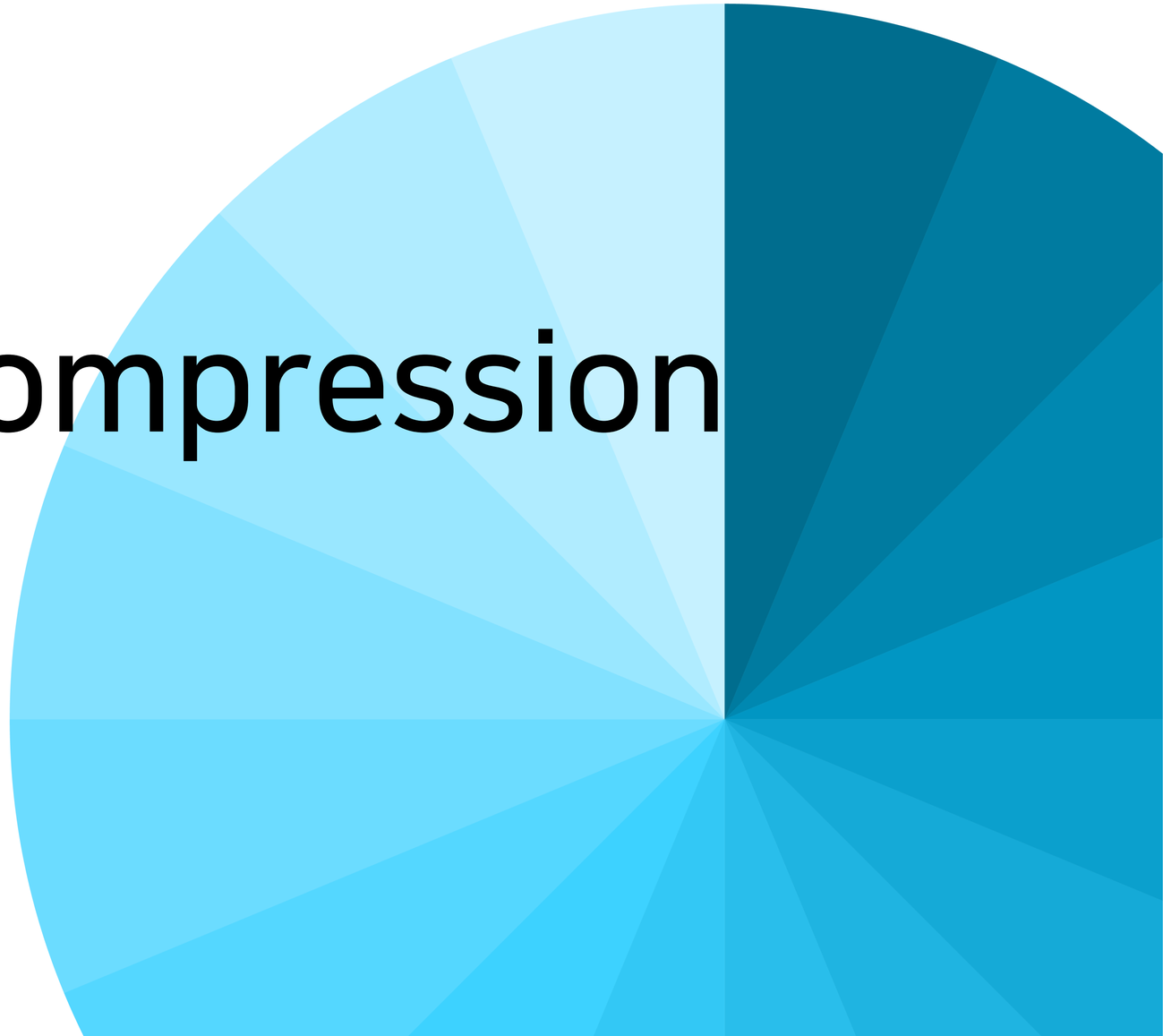
메모리

전력

Green AI (Schwartz et al., 2019)

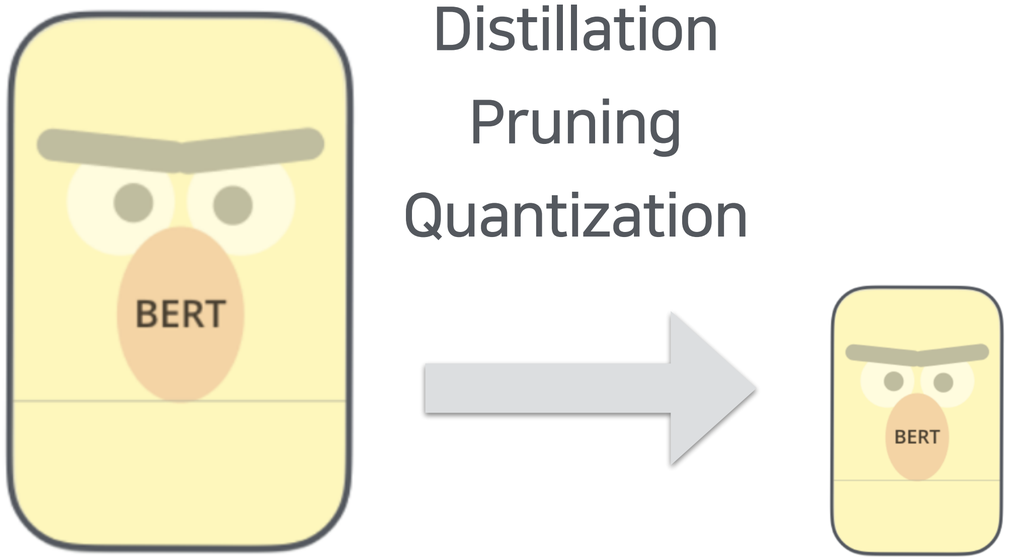
Energy and Policy Considerations for Deep Learning in NLP (Strubell et al., 2019)

2. Model Compression



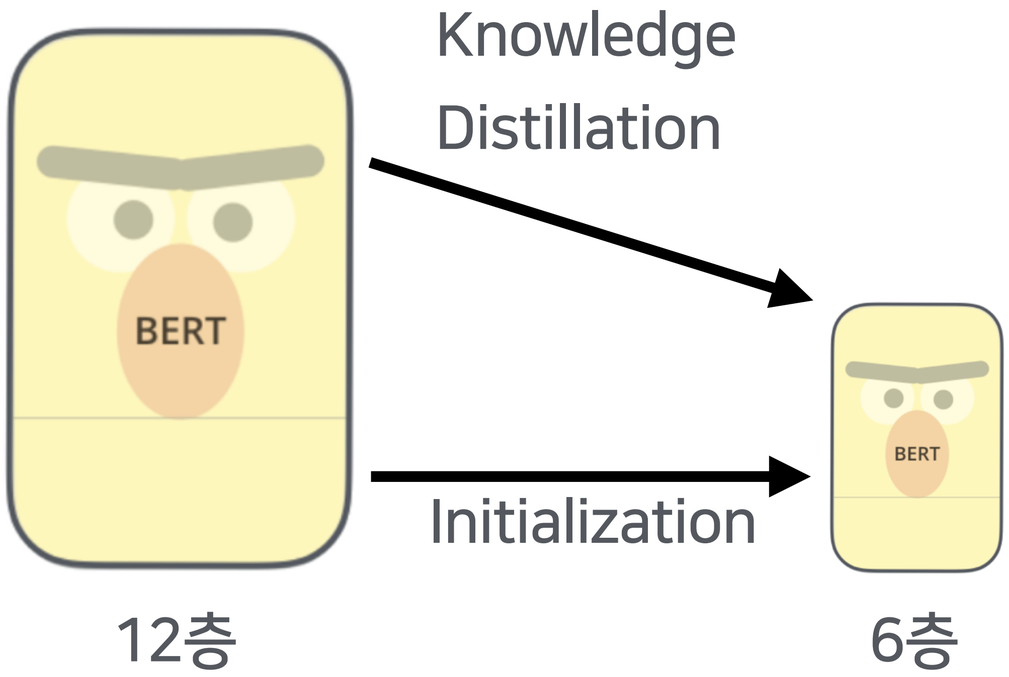
2.1 작지만 정확한 모델

일단 큰 모델을 만들고 작게 줄이기



2.2 DistilBERT

큰 모델 (선생님)과 비슷하게 예측하도록 작은 모델 (학생)을 훈련



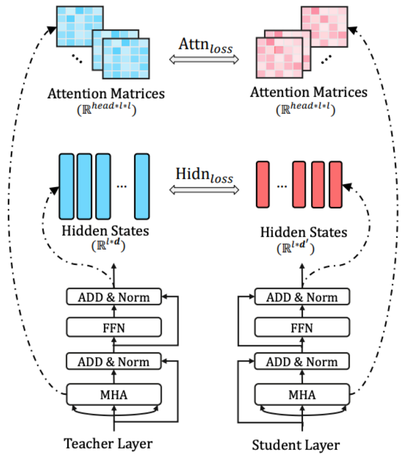
DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (Sanh et al., NeurIPS 2019 Workshop)

2.3 DistilLaRva

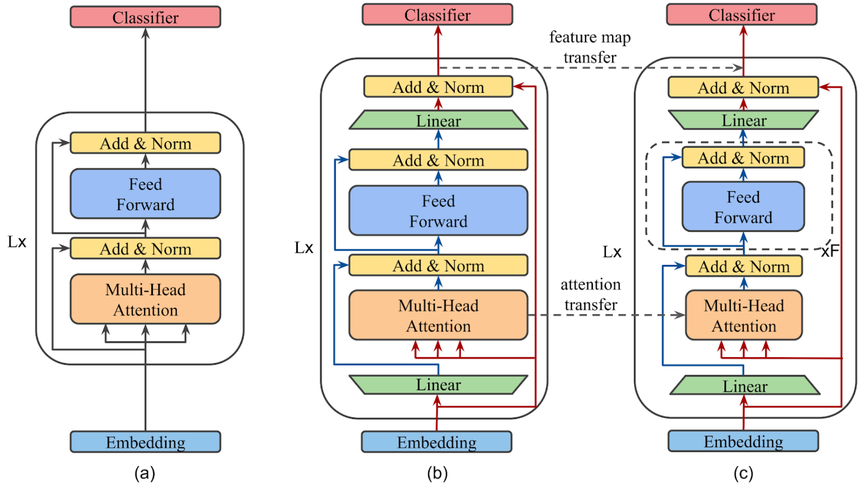
DEVIEW 2020

	HQA	HQA-long	KorQuAD	MR-2	MR-5	QS	SM	PAWS
mBERT (L12)	91.8	90.6	90.5	87.9	74.8	86.6	88.7	79.3
LaRva-kor (L12)	95.1	94.3	93.9	90.0	75.5	87.4	94.2	84.9
LaRva-kor (L6 from scratch)	93.1	92.9	91.8	89.2	75.3	85.9	90.9	80.5
DistilLaRva (L6)	**93.9**	**93.6**	**92.5**	**89.6**	**75.4**	**86.7**	**93.4**	**81.6**

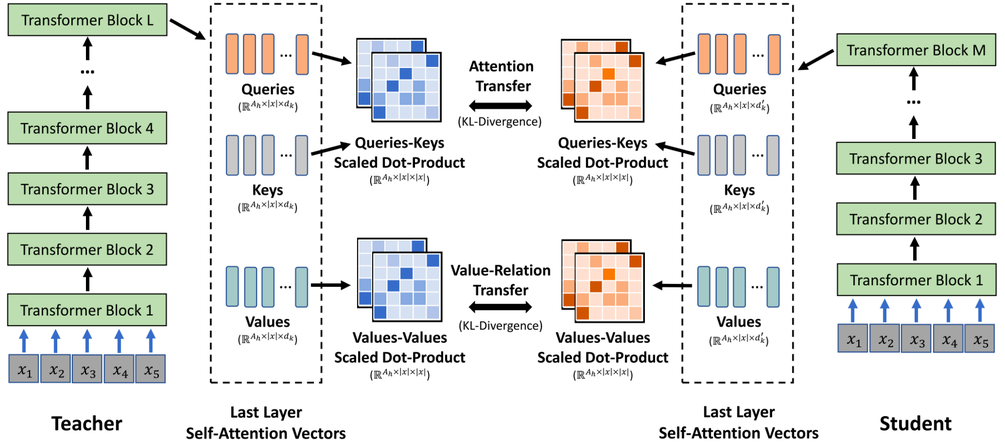
2.4 다른 Distillation된 BERT 모델들



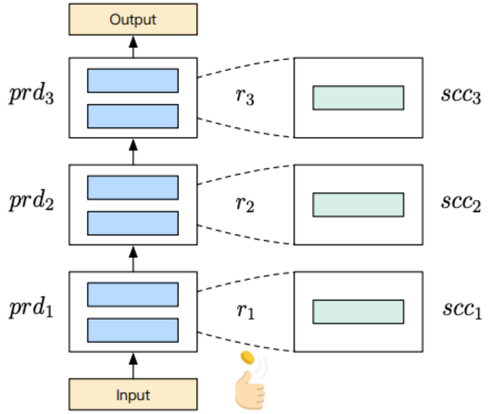
TinyBERT: Distilling BERT for Natural Language Understanding (Jiao et al., Findings of EMNLP 2020)



MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices (Sun et al., ACL 2020)



MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers (Wang et al., 2020)



BERT-of-Theseus: Compressing BERT by Progressive Module Replacing (Xu et al., EMNLP 2020)

3. Better Attention Mechanism

3.1 다양한 X-former들

DEVIEW 2020

Self-attention의 연산 복잡도 개선

Long-term context 반영 가능

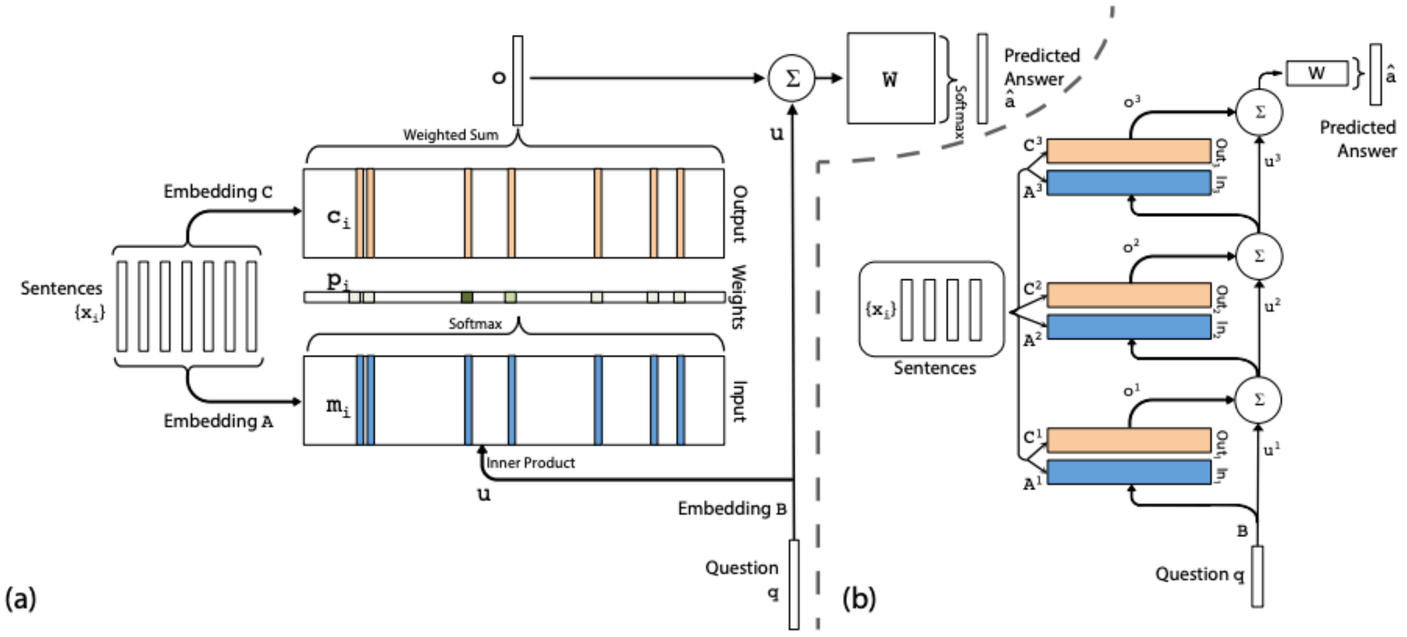
Model / Paper	Complexity	Decode	Class
Memory Compressed† (Liu et al., 2018)	$\mathcal{O}(n_c^2)$	✓	FP+M
Image Transformer† (Parmar et al., 2018)	$\mathcal{O}(n.m)$	✓	FP
Set Transformer† (Lee et al., 2019)	$\mathcal{O}(nk)$	✗	M
Transformer-XL† (Dai et al., 2019)	$\mathcal{O}(n^2)$	✓	RC
Sparse Transformer (Child et al., 2019)	$\mathcal{O}(n\sqrt{n})$	✓	FP
Reformer† (Kitaev et al., 2020)	$\mathcal{O}(n \log n)$	✓	LP
Routing Transformer (Roy et al., 2020)	$\mathcal{O}(n \log n)$	✓	LP
Axial Transformer (Ho et al., 2019)	$\mathcal{O}(n\sqrt{n})$	✓	FP
Compressive Transformer† (Rae et al., 2020)	$\mathcal{O}(n^2)$	✓	RC
Sinkhorn Transformer† (Tay et al., 2020b)	$\mathcal{O}(b^2)$	✓	LP
Longformer (Beltagy et al., 2020)	$\mathcal{O}(n(k+m))$	✓	FP+M
ETC (Ainslie et al., 2020)	$\mathcal{O}(n_g^2 + nn_g)$	✗	FP+M
Synthesizer (Tay et al., 2020a)	$\mathcal{O}(n^2)$	✓	LR+LP
Performer (Choromanski et al., 2020)	$\mathcal{O}(n)$	✓	KR
Linformer (Wang et al., 2020b)	$\mathcal{O}(n)$	✗	LR
Linear Transformers† (Katharopoulos et al., 2020)	$\mathcal{O}(n)$	✓	KR
Big Bird (Zaheer et al., 2020)	$\mathcal{O}(n)$	✗	FP+M

Efficient Transformers: A Survey (Tay et al., 2020)

4. Memory Augmentation

4.1 메모리 구조

모델 파라메터와 별도로 모델의 표현력을 증가

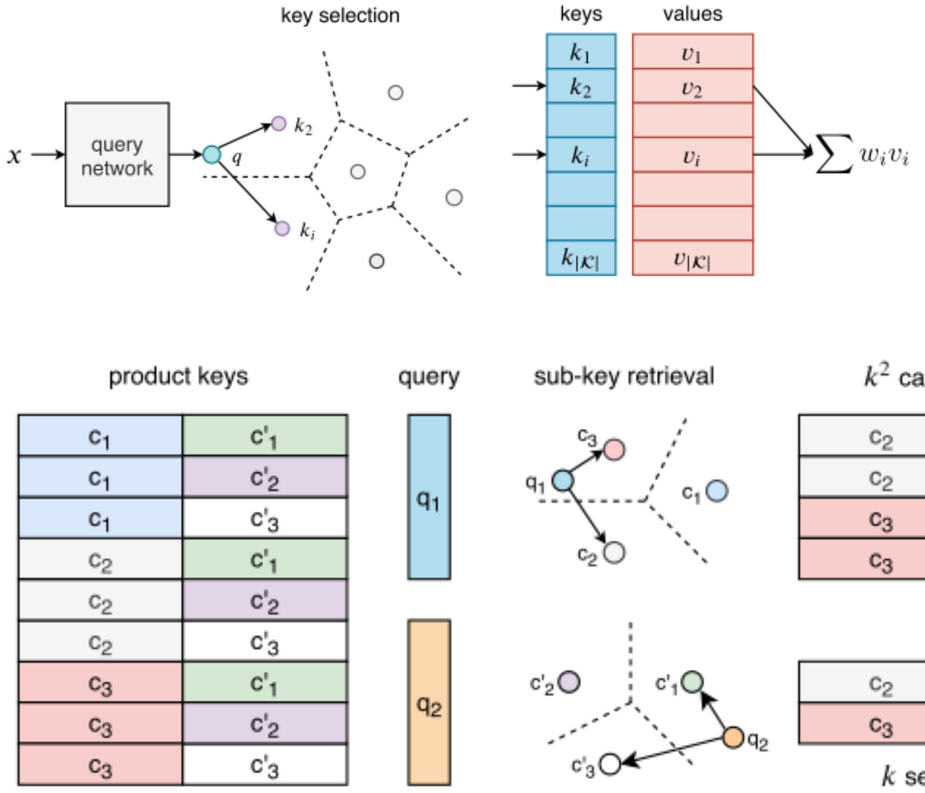


End-To-End Memory Networks (Sukhbaatar et al., 2015)

4.2 Product Key Memory (PKM)

DEVIEW 2020

메모리 크기를 효율적으로 키우면서 빠르고 정확한 탐색 가능

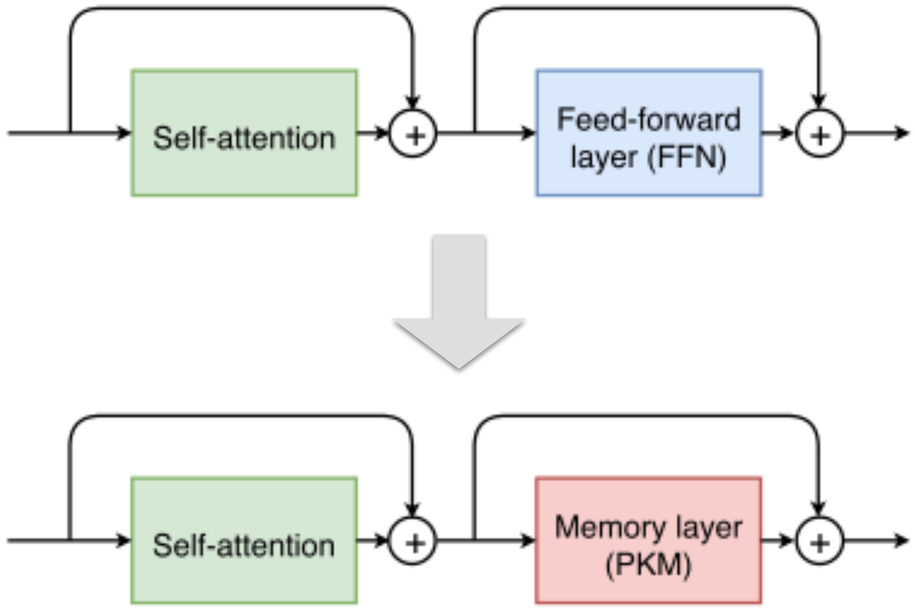


Large Memory Layers with Product Keys (Lample et al., NeurIPS 2019)

4.2 Product Key Memory (PKM)

DEVIEW 2020

Feed-forward layer를 PKM으로 대체

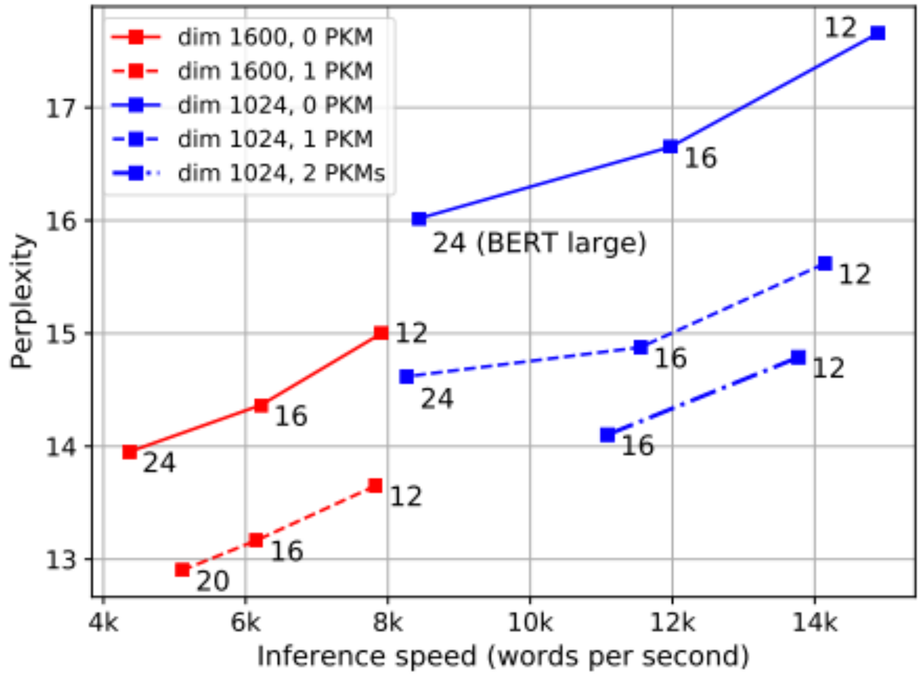


Large Memory Layers with Product Keys (Lample et al., NeurIPS 2019)

4.2 Product Key Memory (PKM)

DEVIEW 2020

속도는 많이 느려지지 않으면서 더 좋은 정확도



Large Memory Layers with Product Keys (Lample et al., NeurIPS 2019)

4.2 Product Key Memory (PKM)

Q: 이런 PKM을 BERT와 같은 PLM에 사용할 수 있을까?

4.3 PKM을 PLM에 써보자

DEVIEW 2020

단순히 적용했더니 도움이 안되네?!

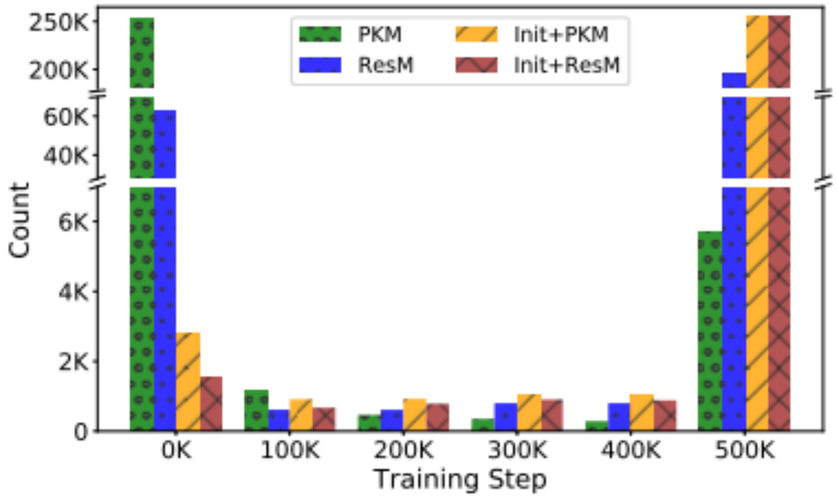
Model	QA	GLUE					
	SQuAD 1.1 (EM/F1)	MNLI-(m/mm) (Acc)	QQP (Acc)	QNLI (Acc)	SST-2 (Acc)	CoLA (Matt)	**Avg** -
(a) $\text{BERT}_{\text{BASE}}^{\dagger}$	82.7/89.8	84.3/84.5	91.0	89.3	92.8	60.8	83.8
(c) +PKM	81.9/89.1	84.4/85.0	91.1	89.0	93.6	59.7	83.8
(g) $\text{BERT}_{\text{BASE}}^{\star}$	81.1/88.5	83.9/84.4	91.0	88.4	92.9	59.8	83.4
(h) $\text{BERT}_{\text{LARGE}}^{\star}$	83.3/90.6	86.2/86.1	91.4	90.4	93.8	64.1	85.3

Large Product Key Memory for Pretrained Language Models (**Gyuwan Kim** and Tae-Hwan Jung, Findings of EMNLP 2020)

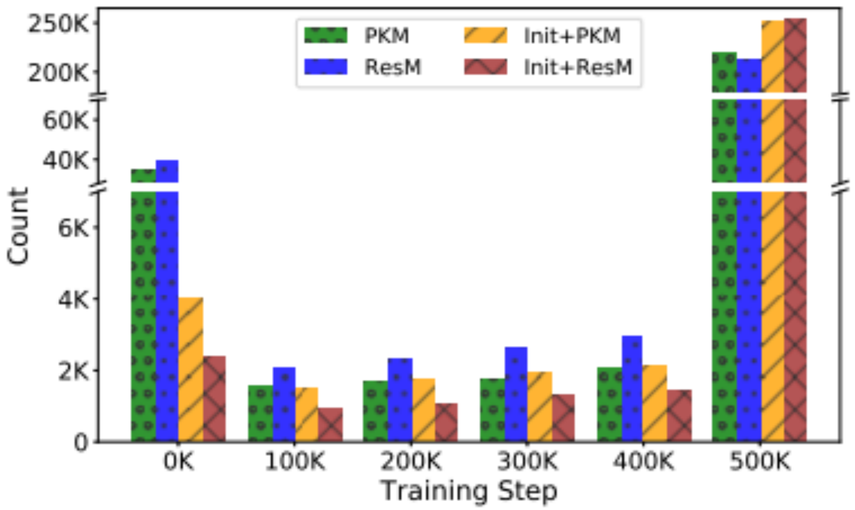
4.3 PKM을 PLM에 써보자

DEVIEW 2020

메모리를 잘 활용하고 있을까? Top-1 → No!
Catastrophic drift



(a) 4th Layer



(b) 8th Layer

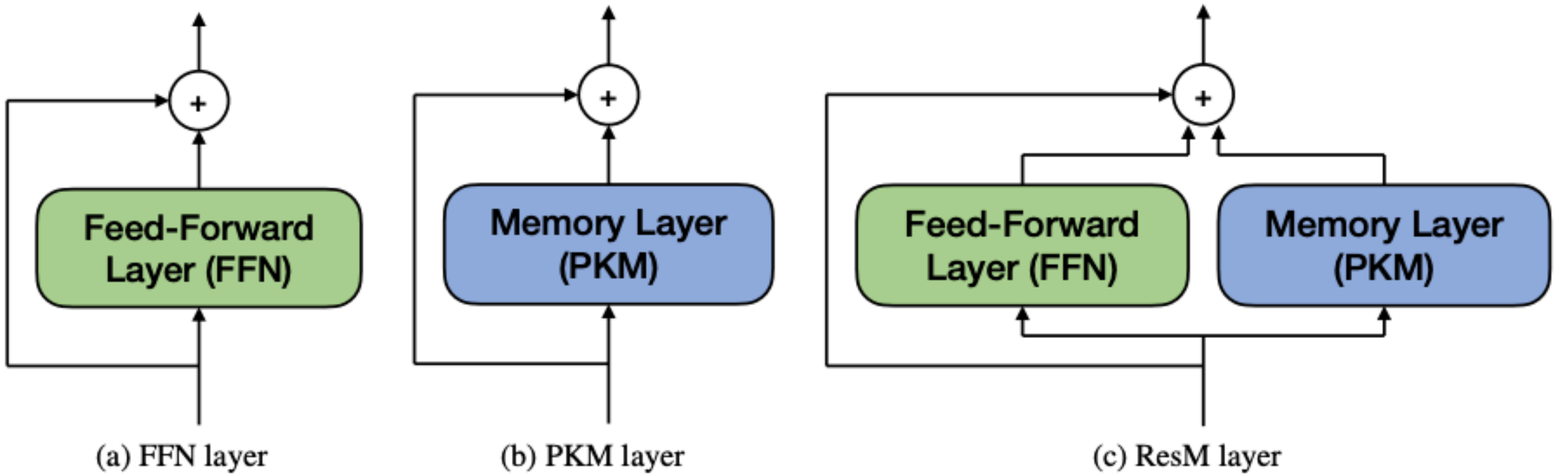
Large Product Key Memory for Pretrained Language Models (**Gyuwan Kim** and Tae-Hwan Jung, Findings of EMNLP 2020)

4.3 PKM을 PLM에 써보자

해결책

(1) 메모리 없이 학습 후 초기화

(2) Residual 메모리



Large Product Key Memory for Pretrained Language Models (Gyuwan Kim and Tae-Hwan Jung, Findings of EMNLP 2020)

4.3 PKM을 PLM에 써보자

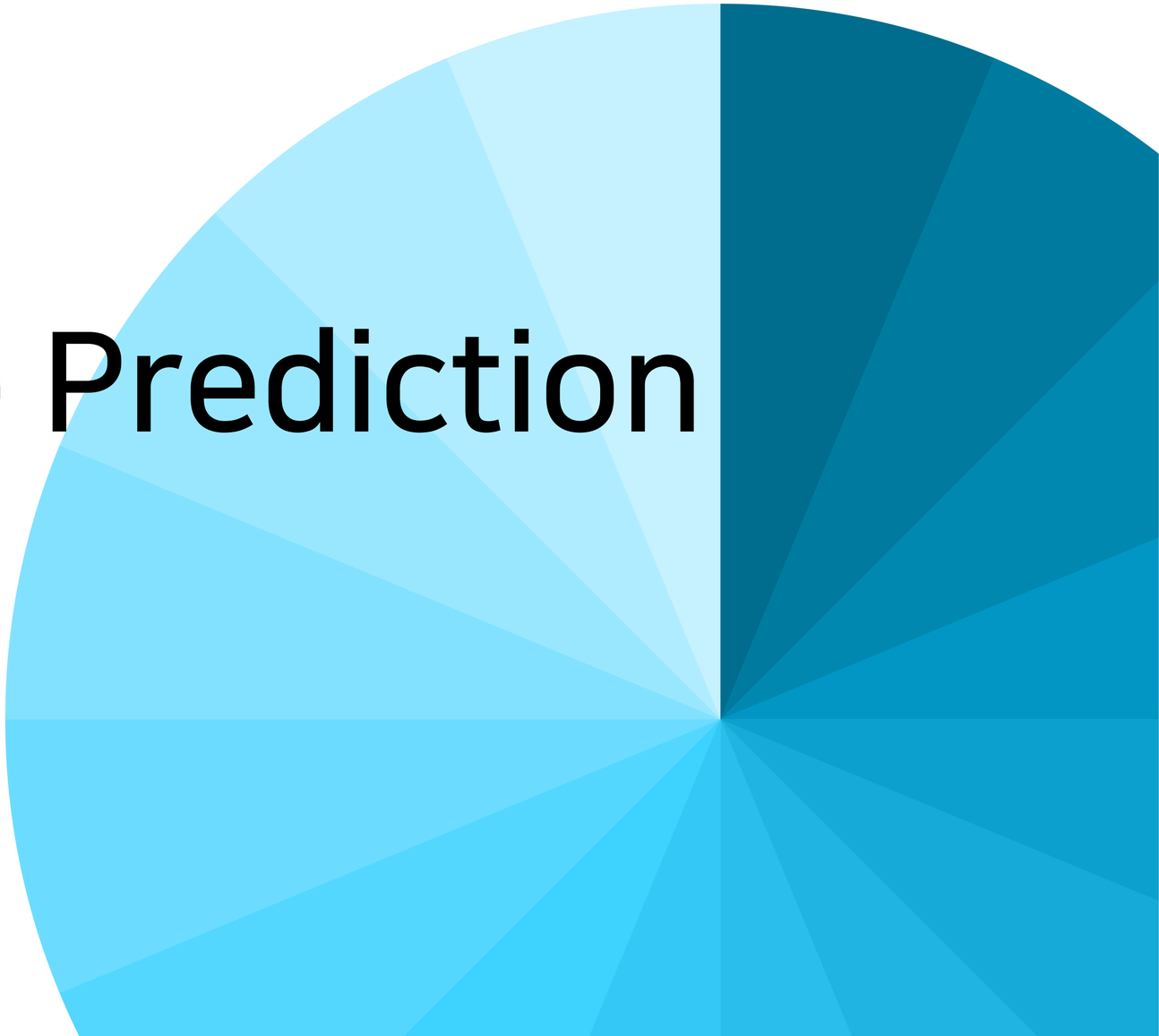
BERT-base에 PKM을 추가하여
BERT-large보다 빠르면서 비슷한 정확도!

Model	# Layers	# Params	Inference Speed (batch/sec)
$\text{BERT}_{\text{BASE}}$	12	110M	79.8
$\text{BERT}_{\text{BASE}}$ +PKM	12	506M	61.4
$\text{BERT}_{\text{BASE}}$ +ResM	12	**515M**	**59.3**
$\text{BERT}_{\text{LARGE}}$	24	**340M**	**43.1**
$\text{BERT}_{\text{LARGE}}$ +PKM	24	860M	37.2
$\text{BERT}_{\text{LARGE}}$ +ResM	24	876M	36.1

Model	QA	GLUE				
	SQuAD 1.1 (EM/F1)	MNLI-(m/mm) (Acc)	QQP (Acc)	QNLI (Acc)	SST-2 (Acc)	CoLA (Matt)	**Avg** -
(a) $\text{BERT}_{\text{BASE}}^{\dagger}$	82.7/89.8	84.3/84.5	91.0	89.3	92.8	60.8	83.8
(b) +500k steps	83.3/90.1	84.8/84.9	91.2	89.2	92.4	61.4	84.0
(c) +PKM	81.9/89.1	84.4/85.0	91.1	89.0	93.6	59.7	83.8
(d) +ResM	81.5/89.4	84.6/84.8	91.0	88.2	93.2	62.8	84.1
(e) +Init +PKM	83.8/90.6	85.8/85.6	91.2	90.0	93.6	63.6	85.0
(f) +Init +ResM	**83.9/90.8**	**86.0/85.8**	**91.4**	**90.4**	**94.0**	**64.1**	**85.3**
(g) $\text{BERT}_{\text{BASE}}^{\star}$	81.1/88.5	83.9/84.4	91.0	88.4	92.9	59.8	83.4
(h) $\text{BERT}_{\text{LARGE}}^{\star}$	83.3/90.6	86.2/86.1	91.4	90.4	93.8	64.1	85.3

Large Product Key Memory for Pretrained Language Models (**Gyuwan Kim** and Tae-Hwan Jung, Findings of EMNLP 2020)

5. Anytime Prediction



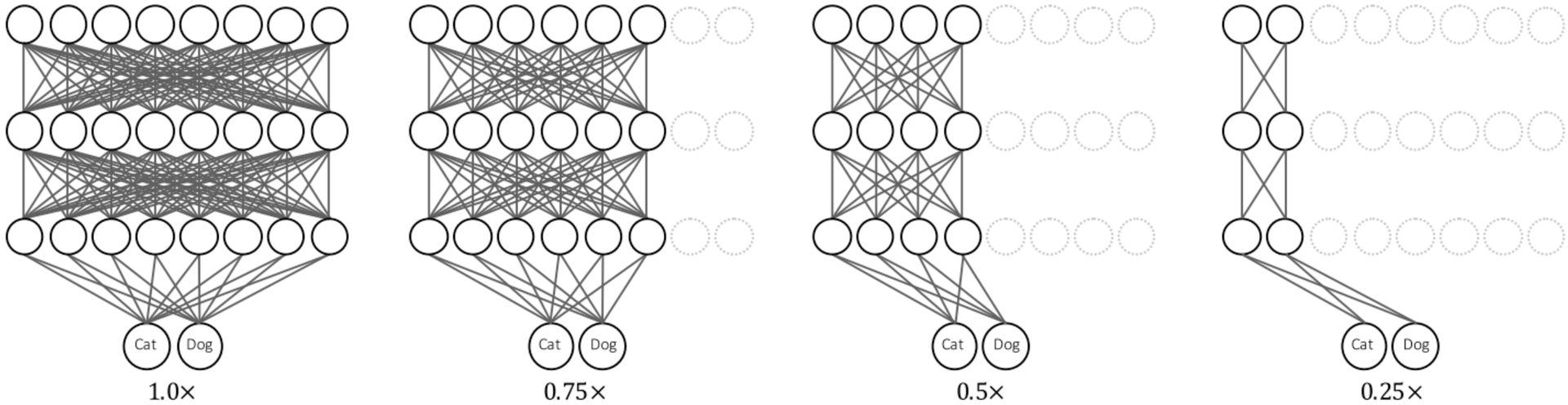
5.1 상황에 따라 선택할 수 있는 모델

하나의 모델로
 어떨 때는 좀 느리더라도 정확히,
 어떨 때는 좀 틀리더라도 빠르게
쓸 수 있을까?

5.2 모델 구조를 선택

DEVIEW 2020

다른 사이즈의 모델을 하나의 share된 weight로 동시에 학습

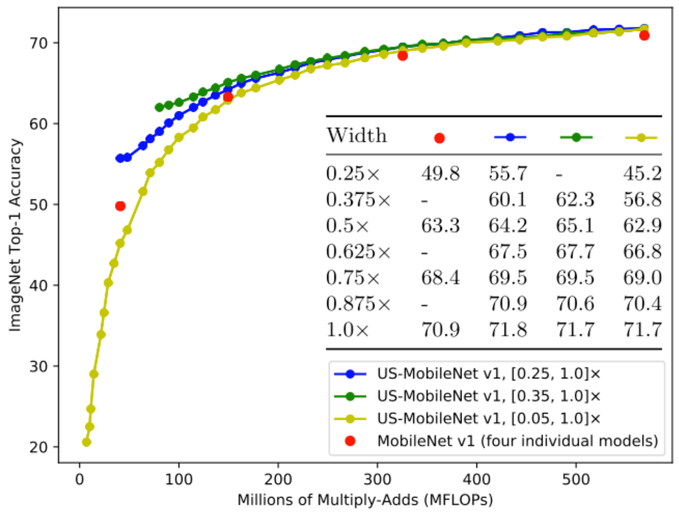


Slimmable Neural Networks (Yu et al., ICLR 2019)

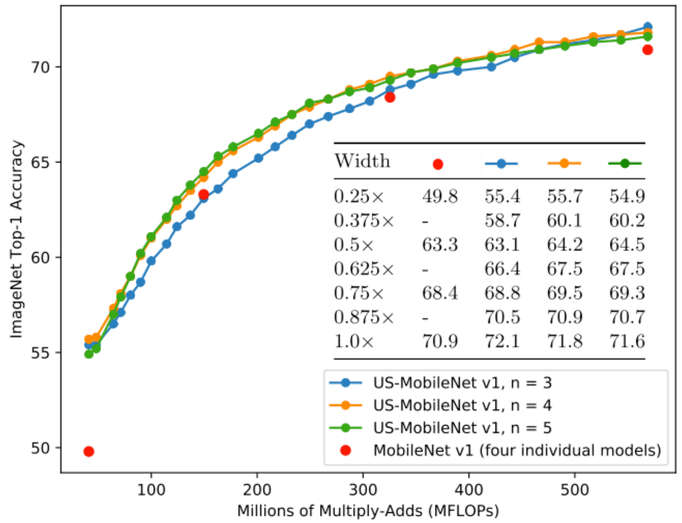
5.2 모델 구조를 선택

DEVIEW 2020

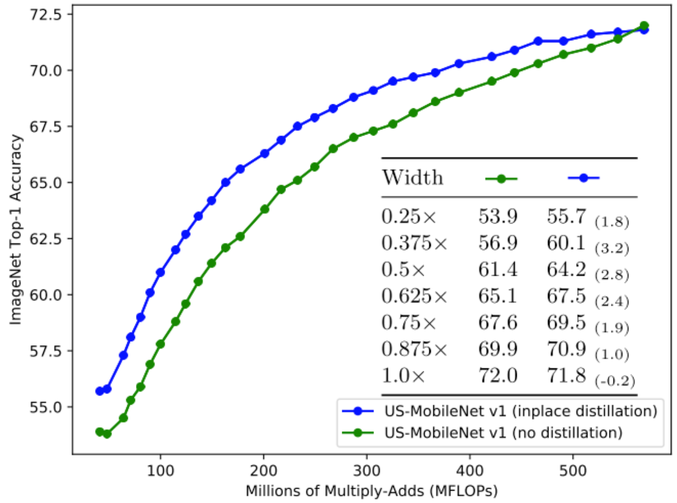
학습 방법이 중요 (sandwich rule과 inplace distillation)



Width	●	—	—	—
0.25×	49.8	55.7	-	45.2
0.375×	-	60.1	62.3	56.8
0.5×	63.3	64.2	65.1	62.9
0.625×	-	67.5	67.7	66.8
0.75×	68.4	69.5	69.5	69.0
0.875×	-	70.9	70.6	70.4
1.0×	70.9	71.8	71.7	71.7



Width	●	—	—	—
0.25×	49.8	55.4	55.7	54.9
0.375×	-	58.7	60.1	60.2
0.5×	63.3	63.1	64.2	64.5
0.625×	-	66.4	67.5	67.5
0.75×	68.4	68.8	69.5	69.3
0.875×	-	70.5	70.9	70.7
1.0×	70.9	72.1	71.8	71.6



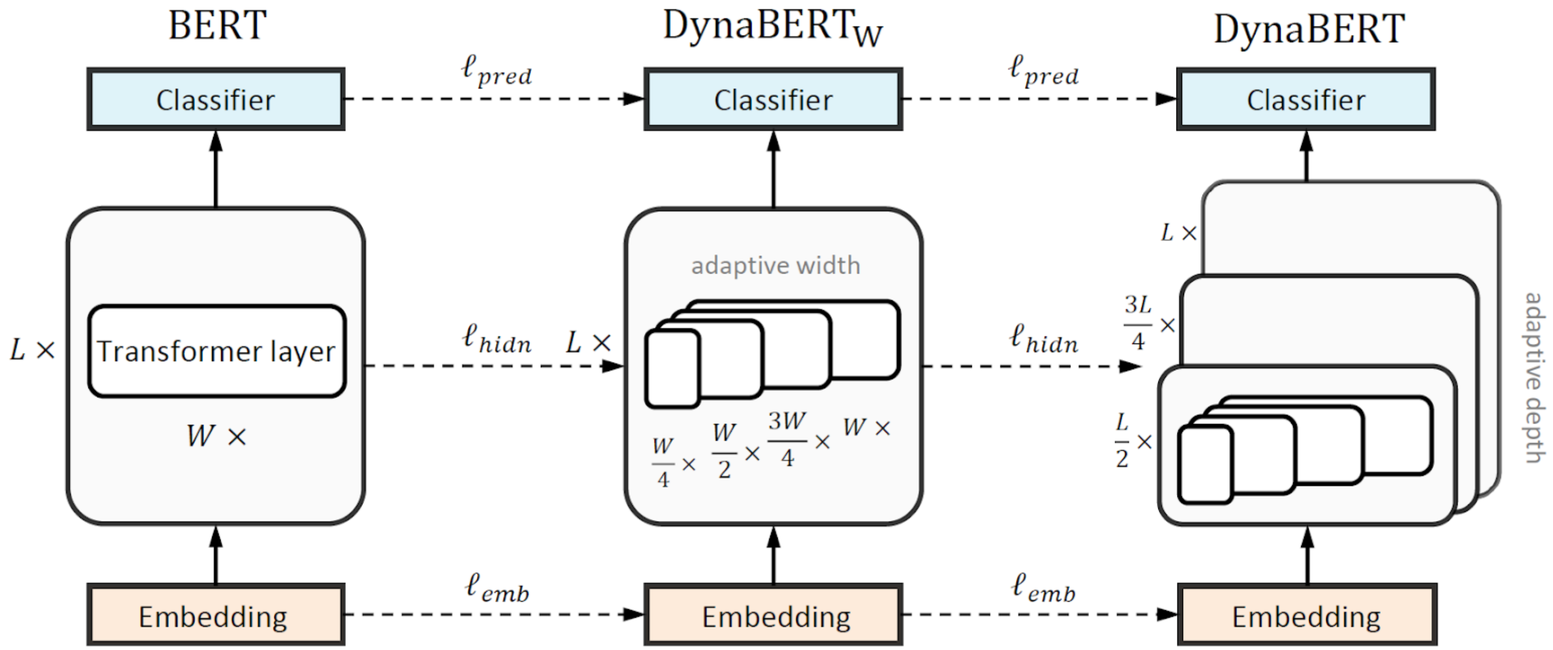
Width	—	—
0.25×	53.9	55.7 (1.8)
0.375×	56.9	60.1 (3.2)
0.5×	61.4	64.2 (2.8)
0.625×	65.1	67.5 (2.4)
0.75×	67.6	69.5 (1.9)
0.875×	69.9	70.9 (1.0)
1.0×	72.0	71.8 (-0.2)

Universally Slimmable Networks and Improved Training Techniques (Yu and Huang, ICCV 2019)

5.2 모델 구조를 선택

DEVIEW 2020

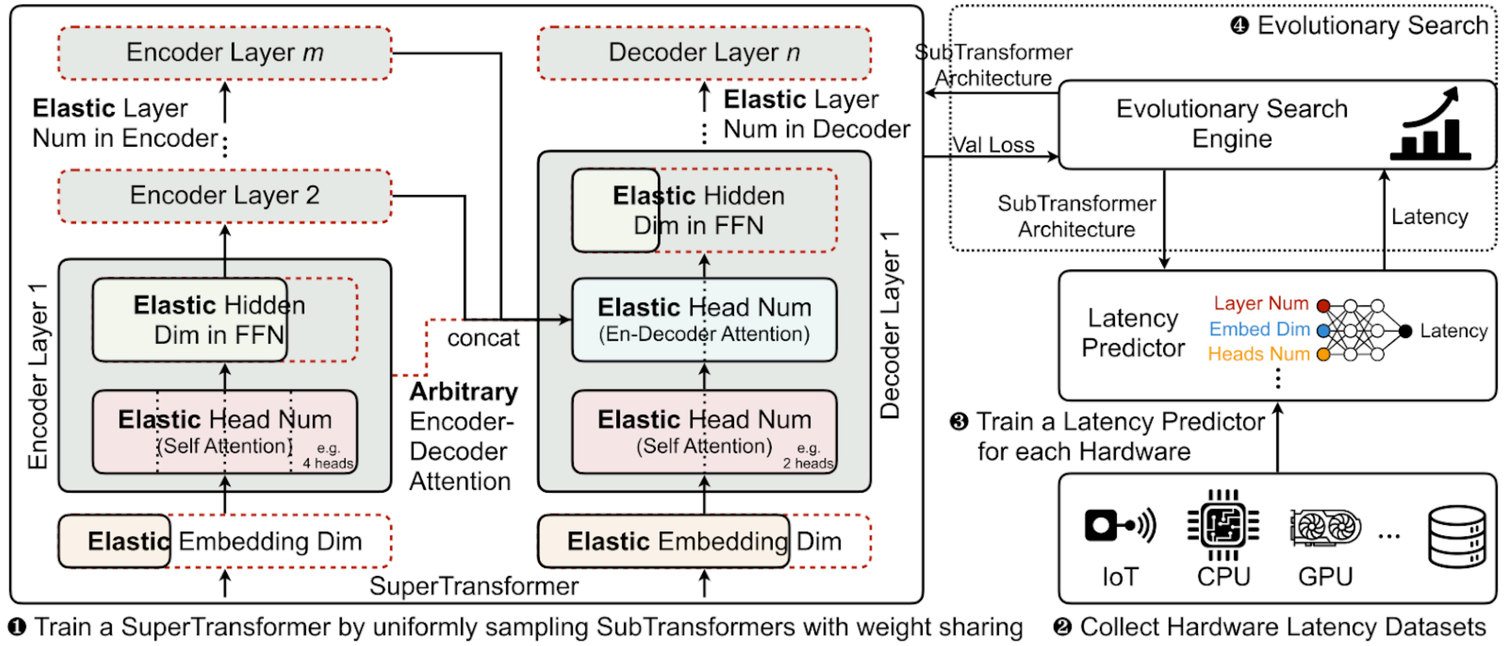
너비와 깊이를 adaptive 하게 선택
Network rewiring



DynaBERT: Dynamic BERT with Adaptive Width and Depth (Hou et al., NeurIPS 2020)

5.2 모델 구조를 선택

Encoder-decoder, 다양한 adaptive dimension
Hardware dependent한 latency budget에 맞게 진화 탐색

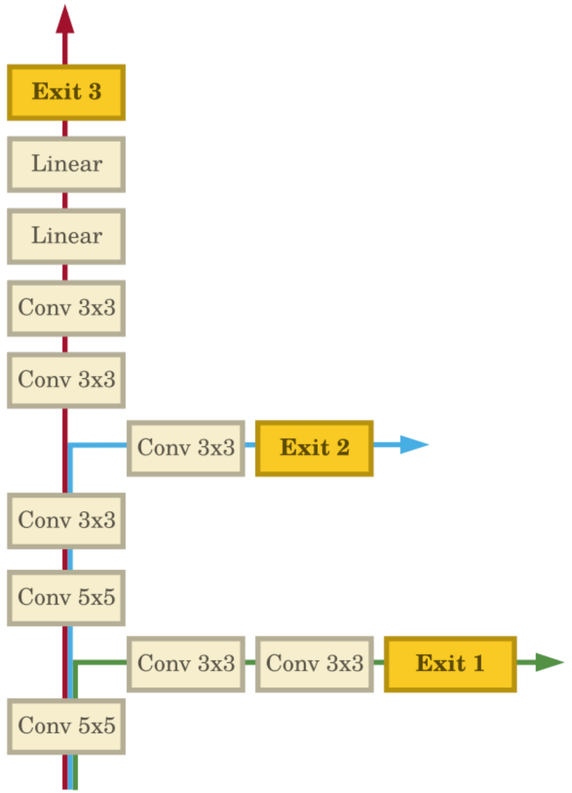


HAT: Hardware-Aware Transformers for Efficient Natural Language Processing (Wang et al., ACL 2020)

5.3 중간에 빠져나가기

입력 example에 따라 다른 난이도

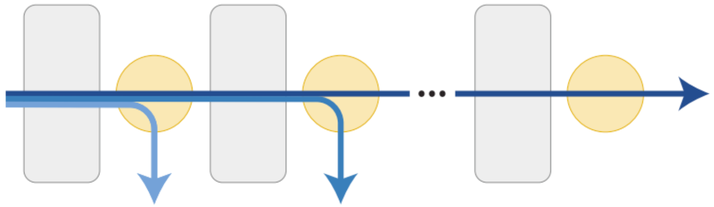
각 레이어 별 분류기들을 동시에 학습 후
사용 때는 예측 값을 확신하면 중간에 빠져나가기



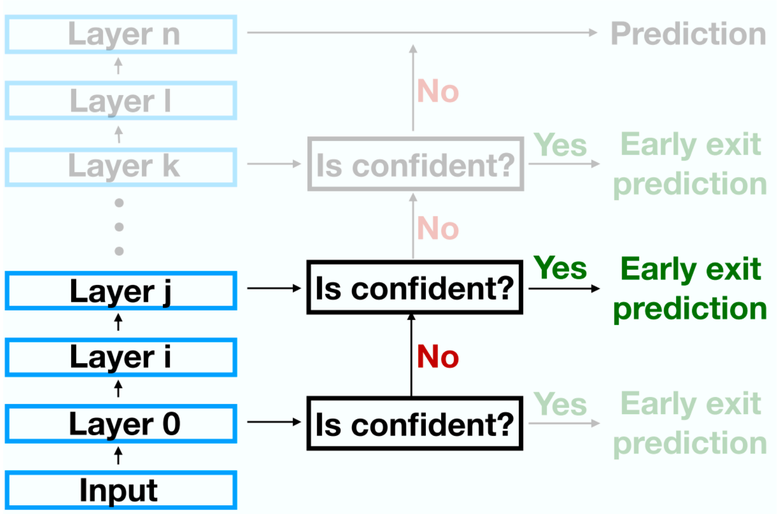
BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks (Teerapittayanon et al., 2017)

Adaptive Computation Time for Recurrent Neural Networks (Graves, 2016)

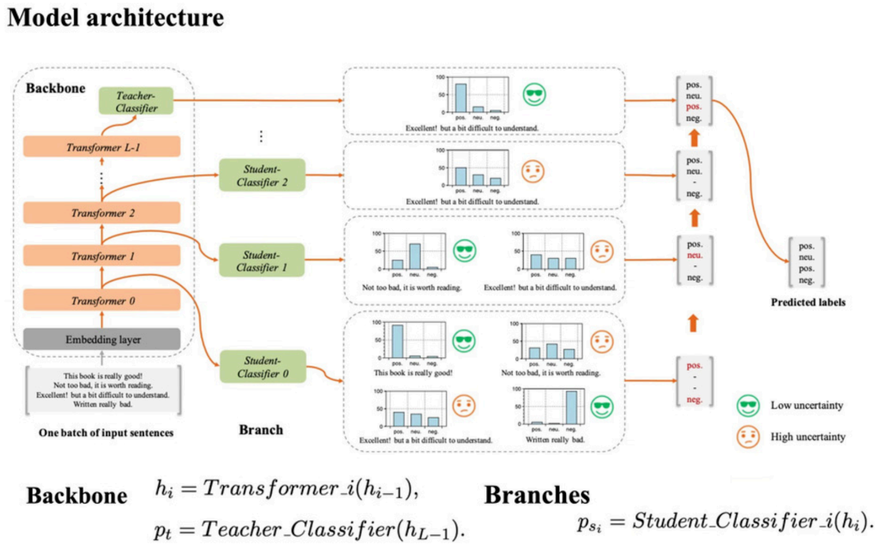
5.3 중간에 빠져나가기



DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference (Xin et al., ACL 2020)



The Right Tool for the Job: Matching Model and Instance Complexities (Schwartz et al., ACL 2020)

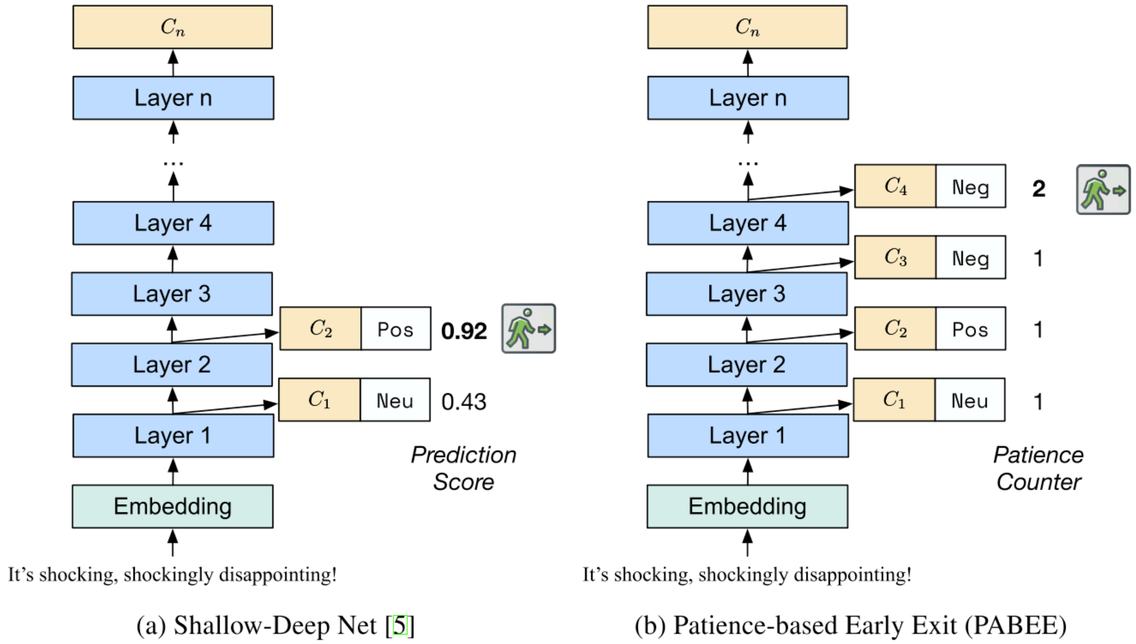


FastBERT: a Self-distilling BERT with Adaptive Inference Time (Liu et al., ACL 2020)

5.3 (참다가) 중간에 빠져나가기

DEVIEW 2020

연속해서 같은 예측을 하면 빠져나가기



(a) Shallow-Deep Net [5]

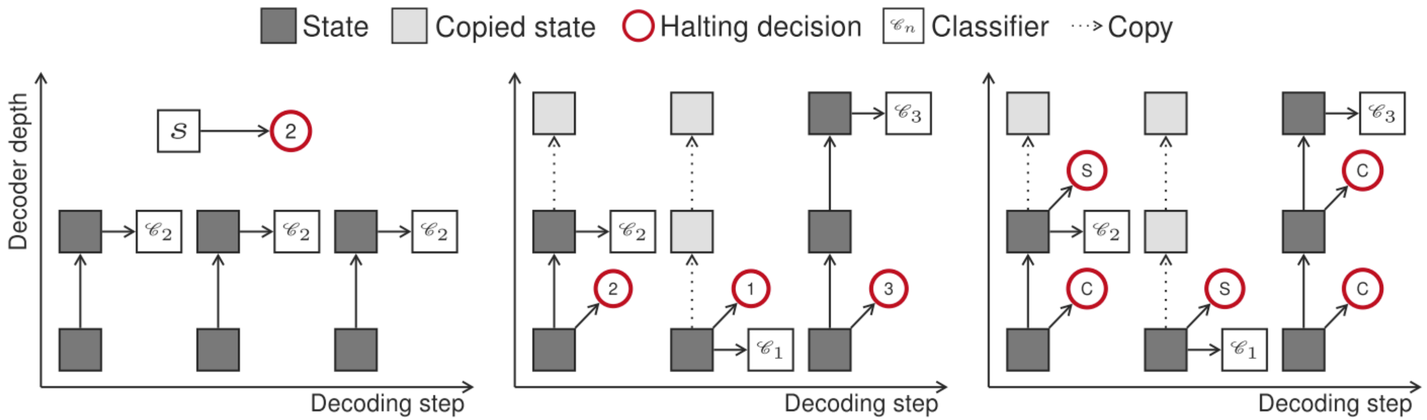
(b) Patience-based Early Exit (PABEE)

BERT Loses Patience: Fast and Robust Inference with Early Exit (Zhou et al., NeurIPS 2020)

5.3 중간에 빠져나가기

DEVIEW 2020

문장 단위가 아닌 단어 단위로 빠져 나가기

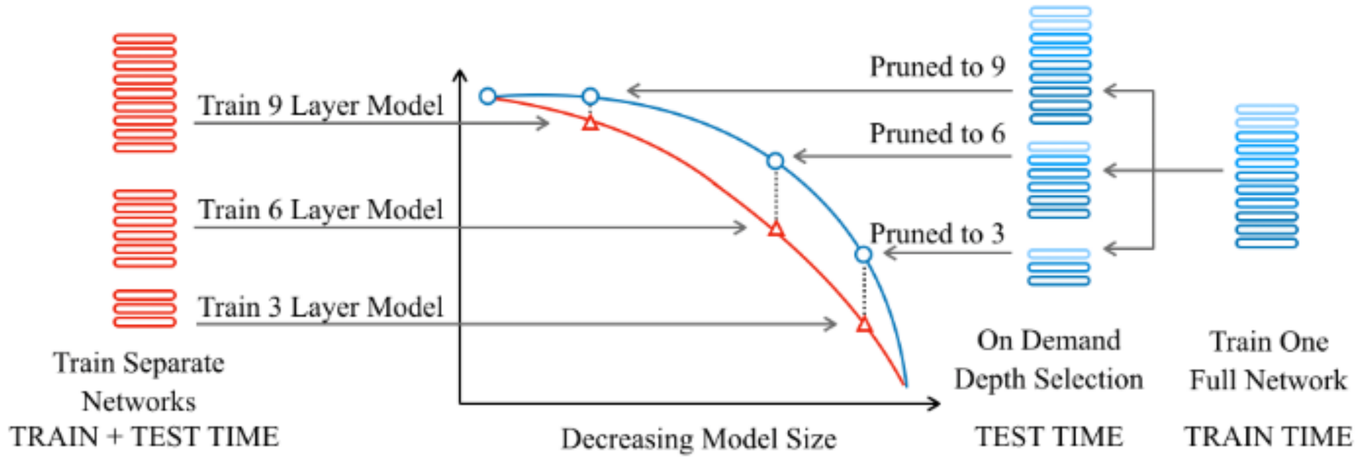


(a) Sequence-specific depth (b) Token-specific - Multinomial (c) Token-specific - Geometric-like

Depth-Adaptive Transformer (Elbayad et al., ICLR 2020)

5.4 레이어 줄이기

학습 때는 임의의 레이어를 drop 시키고 (LayerDrop)
나중에는 일부 레이어만 골라서 사용

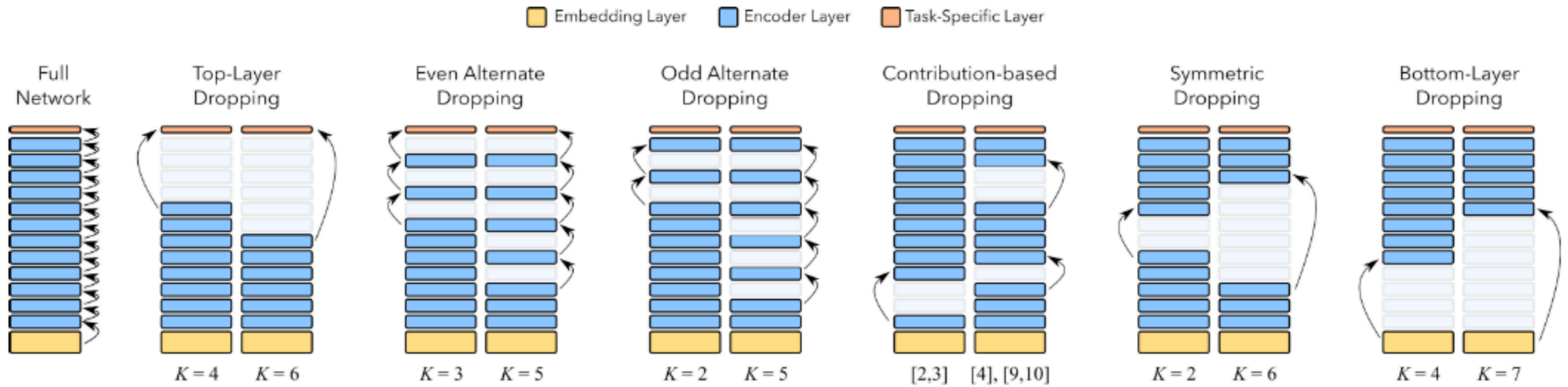


Reducing Transformer Depth on Demand with Structured Dropout (Fan et al., ICLR 2020)

5.4 레이어 줄이기

DEVIEW 2020

레이어 줄이는 다양한 방식 시도



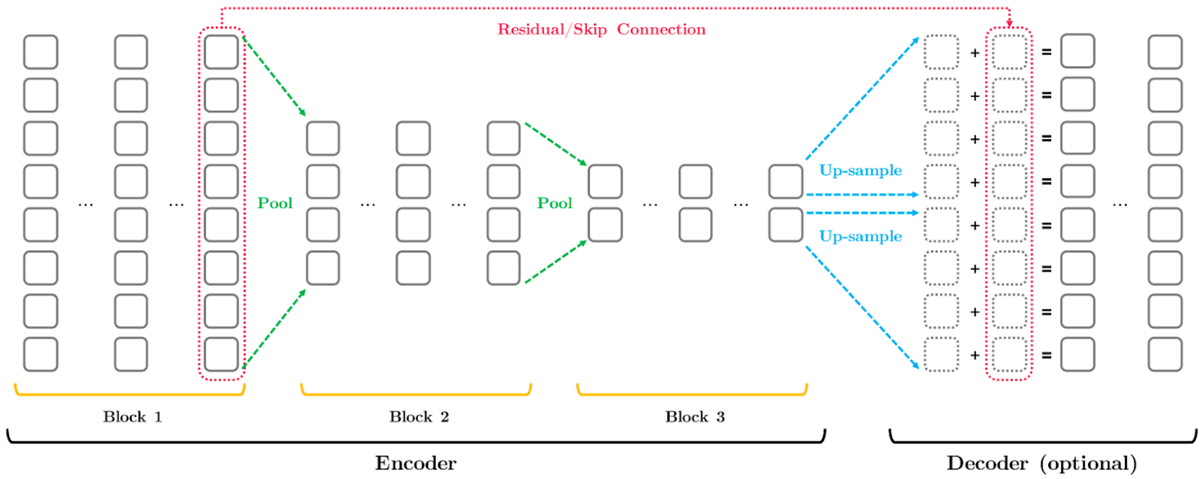
Poor Man's BERT: Smaller and Faster Transformer Models (Sajjad et al., 2019)

5.5 길이 줄이기

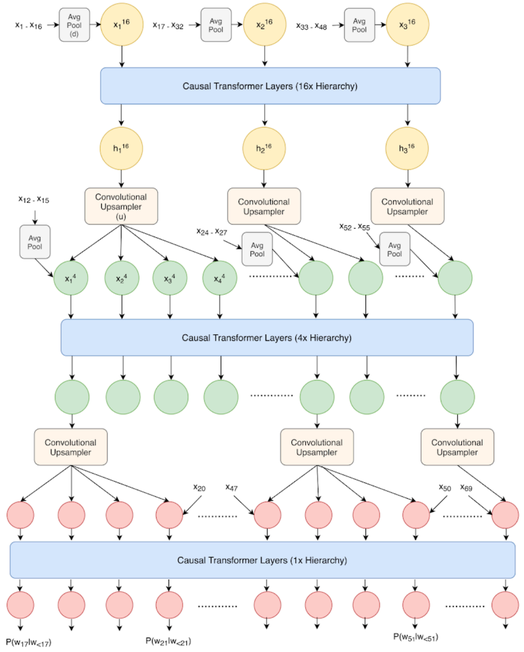
DEVIEW 2020

Self-attention을 생각하면 효과적으로 효율성을 높일 수 있는 방법

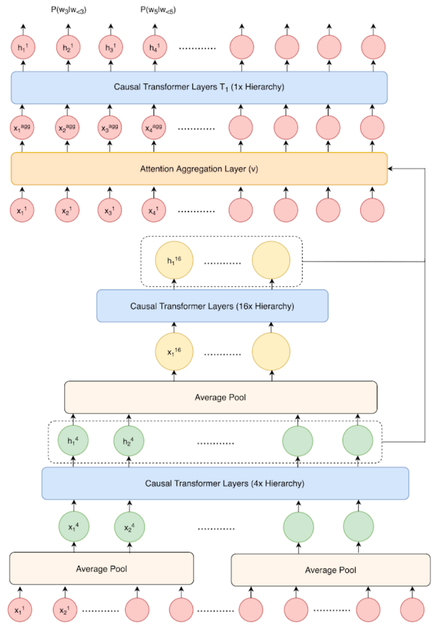
정보 손실의 가능성



Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing (Dai et al., NeurIPS 2020)



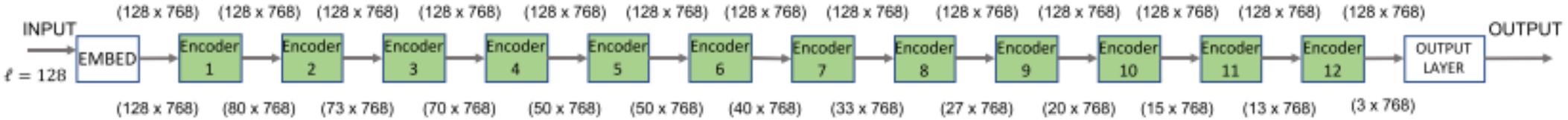
(a) Top-down Model



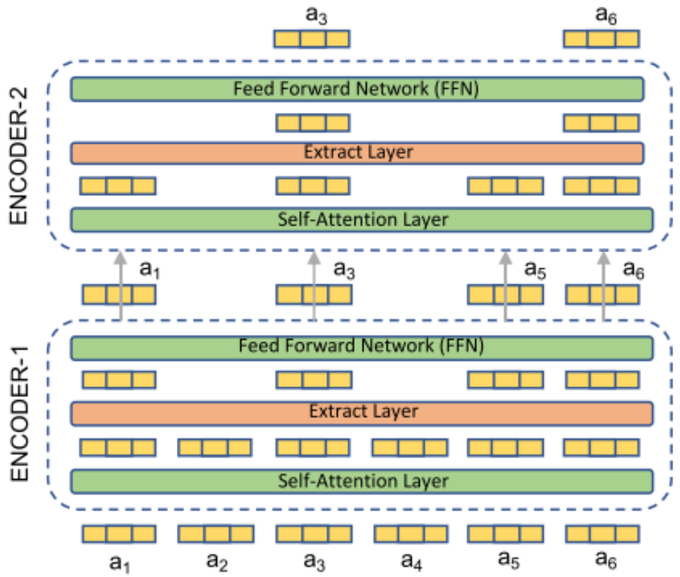
(b) Bottom-up Model

Multi-scale Transformer Language Models (Subramanian et al., 2020)

5.5 길이 줄이기



Attention 점수 기반으로 덜 중요한 단어를 제거
각 layer 마다 얼마나 남길지: 길이 설정

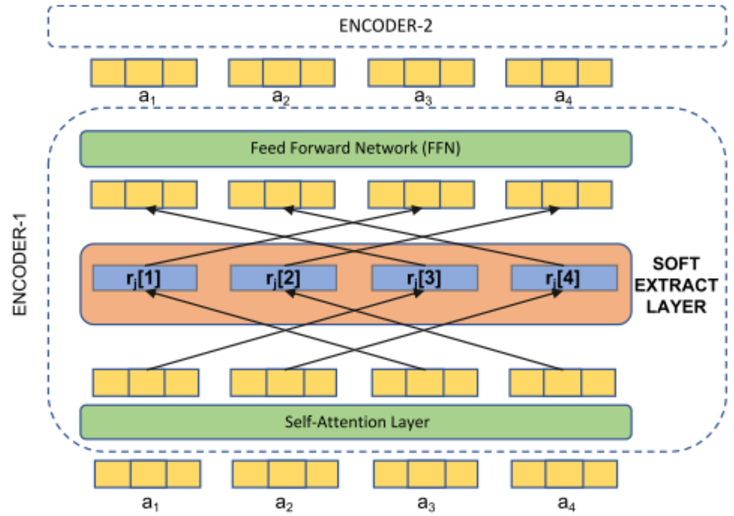


PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination (Goyal et al., ICML 2020)

5.5 길이 줄이기

DEVIEW 2020

학습을 통한 길이 설정 탐색
찾아진 길이 설정으로 PoWER-BERT 학습



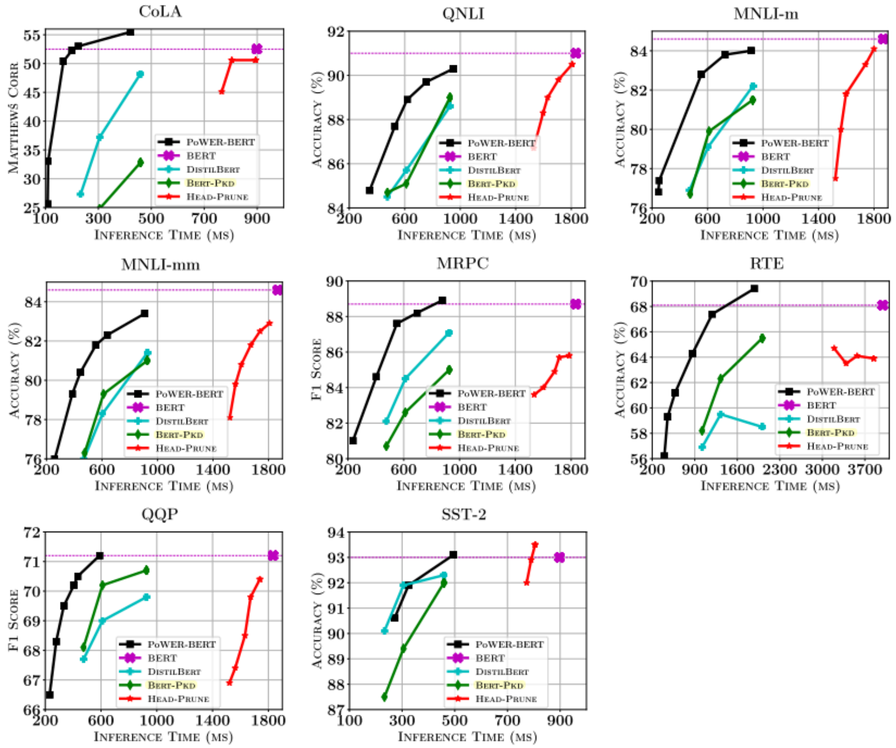
$$\min_{\mathbf{\Theta},\mathbf{r}} \left[\mathcal{L}(\mathbf{\Theta}, \mathbf{r}) \; + \; \lambda \cdot \sum_{j=1}^{L} j \cdot \text{mass}(j \; ; \; \mathbf{r}) \right] \quad \text{s.t.} \;\; \mathbf{r}_j[k] \in [0, 1]$$

PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination (Goyal et al., ICML 2020)

5.5 길이 줄이기

DEVIEW 2020

다른 방법 대비 더 나은 trade-off



PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination (Goyal et al., ICML 2020)

5.5 길이 줄이기

Q1: 하나의 모델로 여러 길이 설정으로 사용할 수 있을까?
Q2: 분류 문제말고 다른 문제에도 적용 할 수 있을까?

5.6 Length-Adaptive Transformer

DEVIEW 2020

Q1: 하나의 모델로 여러 길이 설정으로 사용할 수 있을까?
A1: LengthDrop으로 학습

Q1′: 학습 후 어떤 길이 설정으로 모델을 사용하면 될까?
A1′: 길이 설정에 대한 진화 탐색

Q2: 분류 문제말고 다른 문제에도 적용 할 수 있을까?
A2: Drop-and-Restore 방법

Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search (**Gyuwan Kim** and Kyunghyun Cho, 2020)

5.6.1 LengthDrop으로 학습

하나의 모델 파라미터로 여러 길이 설정에 해당 하는 모델을 동시 학습
매 위치에서 얼마나 길이를 줄일지 랜덤하게 결정

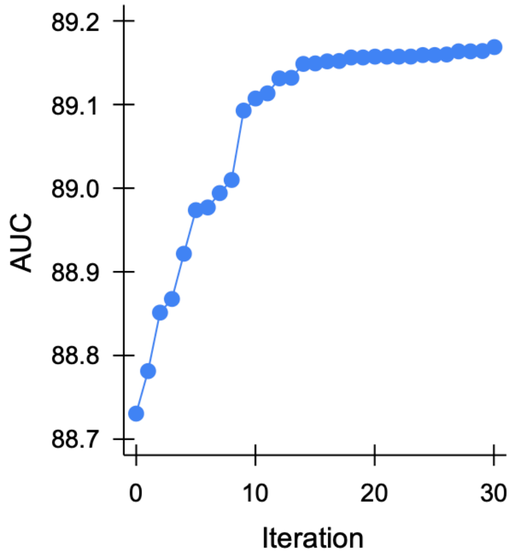
LayerDrop, sandwich rule과 inplace distillation 사용

Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search (**Gyuwan Kim** and Kyunghyun Cho, 2020)
Universally Slimmable Networks and Improved Training Techniques (Yu and Huang, ICCV 2019)

5.6.2 길이 설정에 대한 진화 탐색

Initialization: 동일한 비율로 줄이는 여러 길이 설정들
Mutation: 일정 확률로 monotonic 제약을 만족하는 범위에서 샘플링
Crossover: 두 길이 설정의 평균

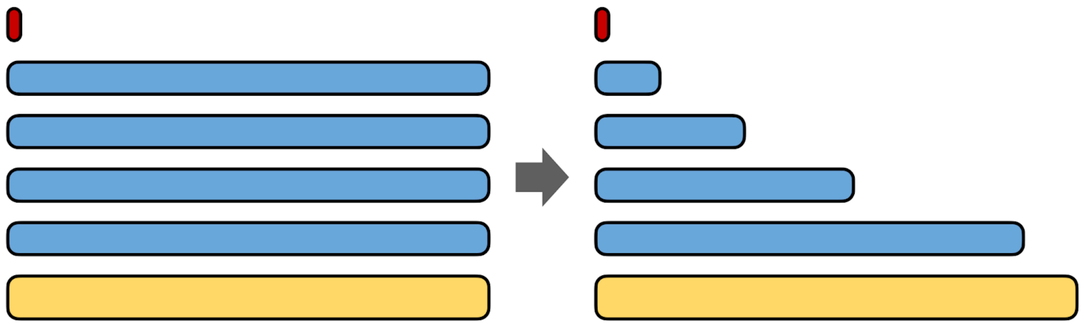
Pareto 곡선 위의 점을 유지하며 세대를 반복



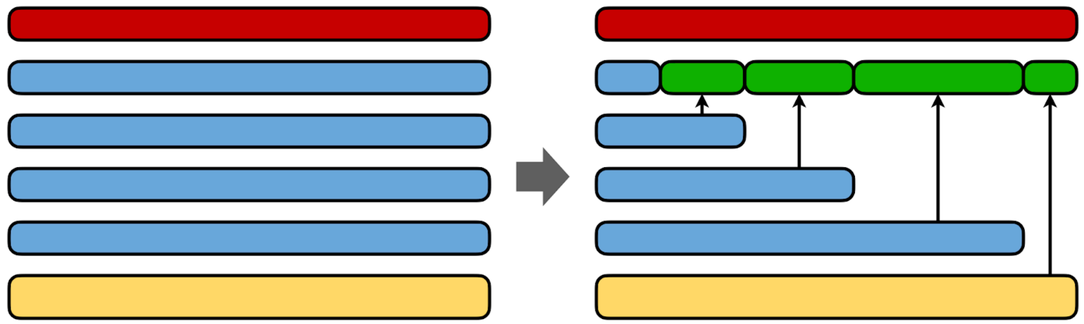
Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search (**Gyuwan Kim** and Kyunghyun Cho, 2020)

5.6.3 Drop-and-Restore 방법

제거된 단어들에 대한 vector들을 마지막 layer에서 다시 복원
Span-based QA 등의 token-level 분류 task에도 적용 가능



PoWER



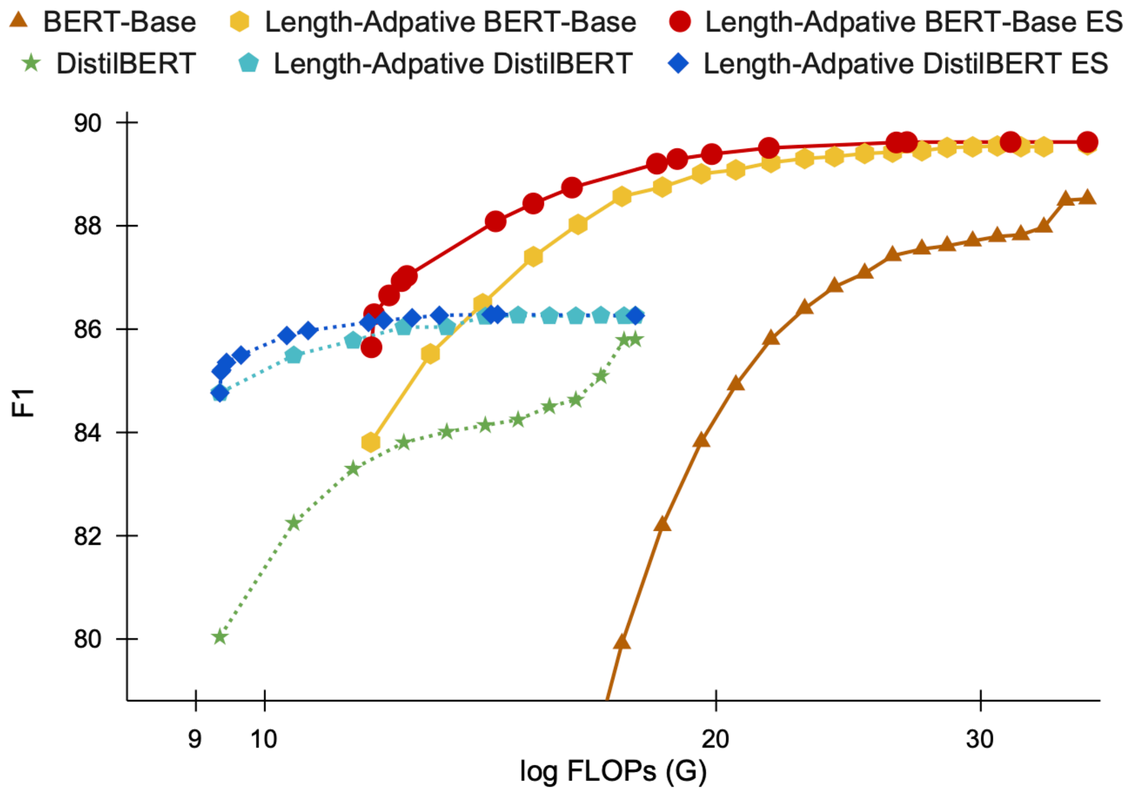
Drop-and-Restore

Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search (**Gyuwan Kim** and Kyunghyun Cho, 2020)

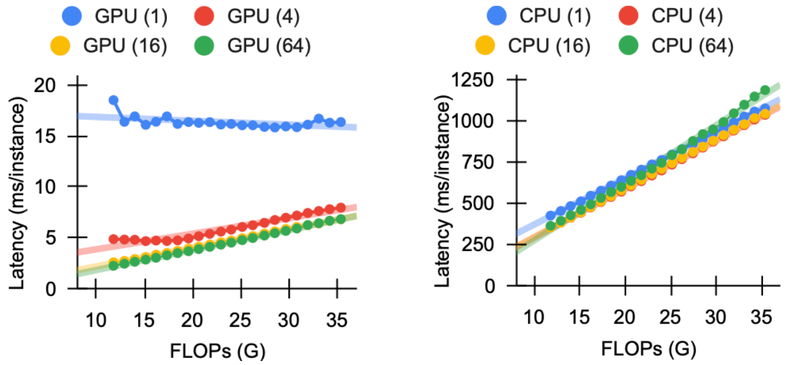
5.6 Length-Adaptive Transformer

DEVIEW 2020

더 빠르고 정확한 모델! (+ anytime prediction 가능)



Pretrained Transformer	Model Method	SQuAD 1.1 F1	SQuAD 1.1 FLOPs	MNLI-m Acc	MNLI-m FLOPs	SST-2 Acc	SST-2 FLOPs
$BERT_{Base}$	Standard	88.5	1.00x	84.4	1.00x	92.8	1.00x
	Lengnth-Adaptive*	89.6	0.89x	85.0	0.58x	93.1	0.36x
	Lengnth-Adaptive†	88.7	0.45x	84.4	0.35x	92.8	0.35x
DistilBERT	Standard	85.8	1.00x	80.9	1.00x	90.6	1.00x
	Lengnth-Adaptive*	86.3	0.81x	81.5	0.56x	92.0	0.55x
	Lengnth-Adaptive†	85.9	0.59x	81.3	0.54x	91.7	0.54x



Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search (**Gyuwan Kim** and Kyunghyun Cho, 2020)

Q & A

Thank You