ACL 2021

Length-Adaptive Transformer:

Train Once with Length Drop, Use Anytime with Search

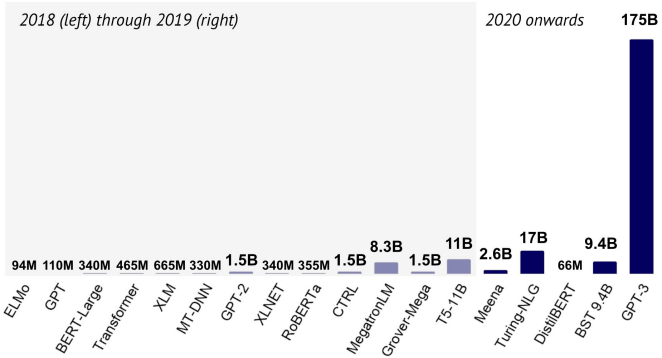
Gyuwan Kim and Kyunghyun Cho

NAVER CLOVA

NYU

Pre-trained Language Models

- Impressive accuracy in various NLP tasks
- Based on transformer architecture
- Improving **efficiency** is important for practical use (Green NLP)

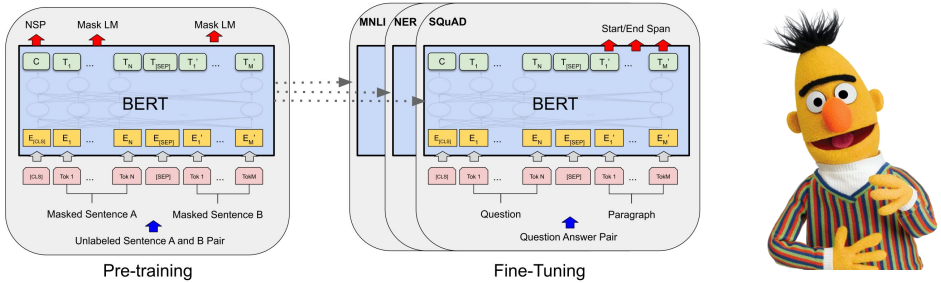


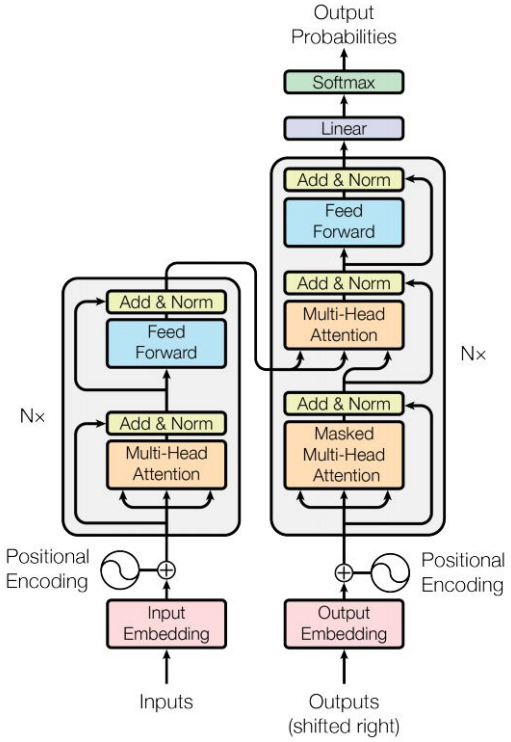
Model sizes of language models
(Source: State of AI Report 2020)

Green AI (Schwartz et al., CACM 2020)
Energy and Policy Considerations for Deep Learning in NLP (Strubell et al., ACL 2019)

Transformers and BERT

- Transformer: widely used NLP model architecture
  - Handle a variable-length sequence input
  - A stack of self-attention and fully connected layers
  - Quadratic complexity to the sequence length
- BERT: Transformer-based masked language model
  - Sequence-level classification: *[CLS]* vector
  - Token-level classification: token vectors



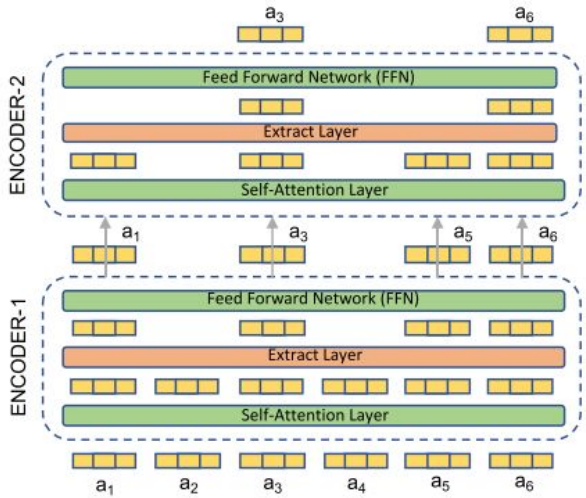


Attention Is All You Need (Vaswani et al., NIPS 2017)

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

PoWER-BERT

- Reducing a **sequence length** while passing transformer layers
  - What to eliminate? Based on significance score from self-attention
  - How many vectors to keep? Length configuration $(l_1, \cdots, l_L)$
  - Training procedure
    i. fine-tuning
    ii. length configuration search
    iii. re-training
- Limitations
  - Separate model for each efficiency budget
  - Applicable only to sentence-level classification tasks



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

Length-Adaptive Transformer

Our framework: train once with length drop, use anytime with search!

Training with LengthDrop

- Training a one-shot model with sampled length configurations to robust to any length reduction
- How to choose length configurations? **LengthDrop**!
  - Sample the next layer sequence length $l_{i+1} \sim Uni(ceil((1-p)l_i), l_i)$ at each layer
- Additional training techniques
  - **LayerDrop**: randomly skip each transformer layer
  - **Sandwich rule**: simultaneously update (1) the full model, (2) several randomly sampled sub-models (sandwiches), and (3) the smallest-possible sub-model for every minibatch
  - **Inplace distillation**: making sub-models' prediction close to the full models' prediction

Reducing Transformer Depth on Demand with Structured Dropout (Fan et al., ICLR 2020)
Universally Slimmable Networks and Improved Training Techniques (Yu and Huang, ICCV 2019)

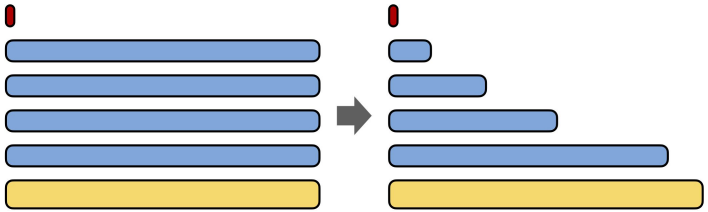
Evolutionary Search of Length Configurations

- Initial population
  - Length configurations of reduction with constant ratios: $l_{i+1} = ceil((1-r)l_i)$
  - Choose several values of r so that the amount of computation is uniformly distributed between those of the smallest and full models
- Mutation
  - Change part of lengths while satisfying the monotonic constraint
  - Sample $l'_i \sim Uni(l'_{i-1}, l_{i+1})$ with the probability p_m or keep $l'_i = l_i$
- Crossover
  - Layer-wise average of two configurations
- Keep configurations on the Pareto curve and iterate multiple times

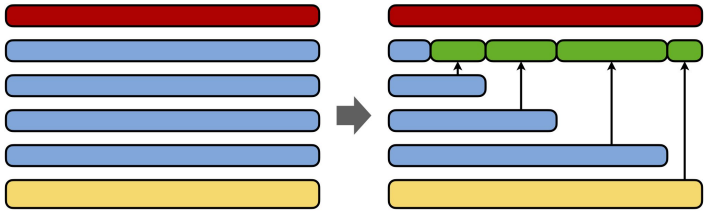
Once-for-All: Train One Network and Specialize it for Efficient Deployment (Cai et al., ICLR 2020)
HAT: Hardware-Aware Transformers for Efficient Natural Language Processing (Wang et al., ACL 2020)

Drop-and-Restore Process

- **Drop** rather than eliminate word vectors at each layer and **Restore** them at the final layer if necessary
- Extend applicability of PoWER-BERT to token-level classification such as span-based question answering



PoWER

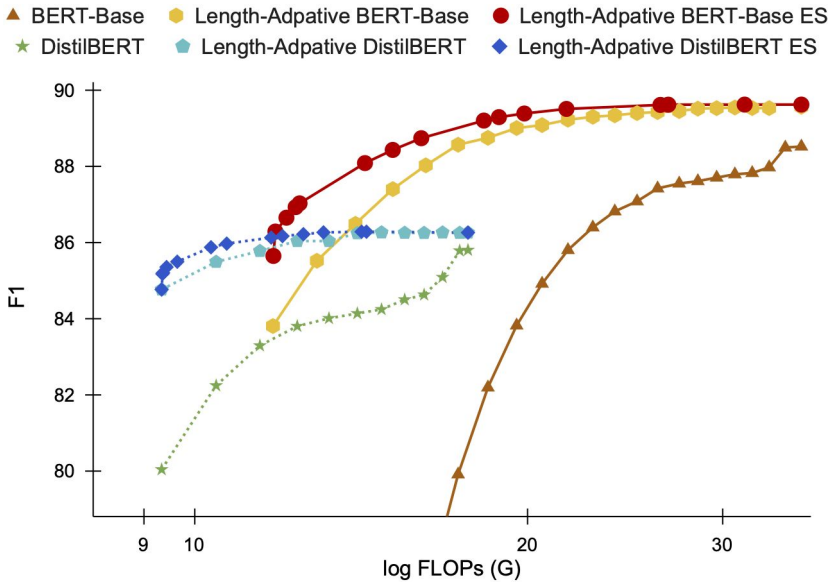


Drop-and-Restore

Experiment Setup

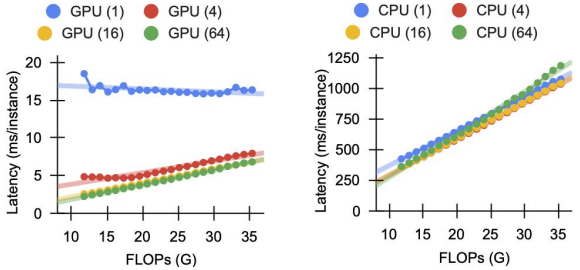
- Datasets
  - Sequence-level classifications - MNLI-m and SST-2
  - Token-level classification - SQuAD 1.1
- Evaluation metrics
  - FLOPs vs accuracy
- Pre-trained transformers
  - BERT-Base (12-layers)
  - DistilBERT (6-layers)
- Learning
  - LengthDrop probability 0.2
  - LayerDrop probability 0.2
  - Two sandwich sub-models
- Search
  - 30 iterations
  - 30 mutations and 30 crossovers per iteration

Better Efficiency-Accuracy Trade-off (+ **Anytime Prediction**!)



Pareto curves on SQuAD 1.1

Pretrained Transformer	Model Method	SQuAD 1.1 F1	SQuAD 1.1 FLOPs	MNLI-m Acc	MNLI-m FLOPs	SST-2 Acc	SST-2 FLOPs
$\text{BERT}_{\text{Base}}$	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



Summary

- **Length-Adaptive Transformer**: train a transformer once and use it for efficient inference under any computational budget
  - Training with LengthDrop
  - Evolutionary search for length configurations
  - Drop-and-Restore process
- Future Work
  - Applying to other transformers and downstream tasks
  - Combination with other dimension reductions or orthogonal methods

For more details, please check our
paper: https://arxiv.org/abs/2010.07003
code: https://github.com/clovaai/length-adaptive-transformer