PennState

BERT, Compression and Applications

Dongkuan (DK) Xu, Ph.D. Candidate
Pennsylvania State University, Advisor: Xiang Zhang
Web: www.personal.psu.edu/dux19/
04-08-2021

Agenda

- BERT Architecture
- Model Compression
- Applications

Background

- BERT (Bidirectional Encoder Representations from Transformers)
  - Published by Google AI Language [1]
  - Achieved state-of-the-art results in various NLP/CV tasks

- Key Innovation
  - Bidirectional training for language modelling
  - Previous efforts looked at a text sequence from left-to-right, right-to-left or combined way

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*

Background

- Performance on GLUE (General Language Understanding Evaluation) Benchmark
  - The most popular collection for training, evaluating and analyzing NLP systems [1]
  - Constructed by NYU, UW and DeepMind

gluebenchmark.com

GLUE SuperGLUE Paper </> Code Tasks Leaderboard FAQ Diagnostics Submit Login

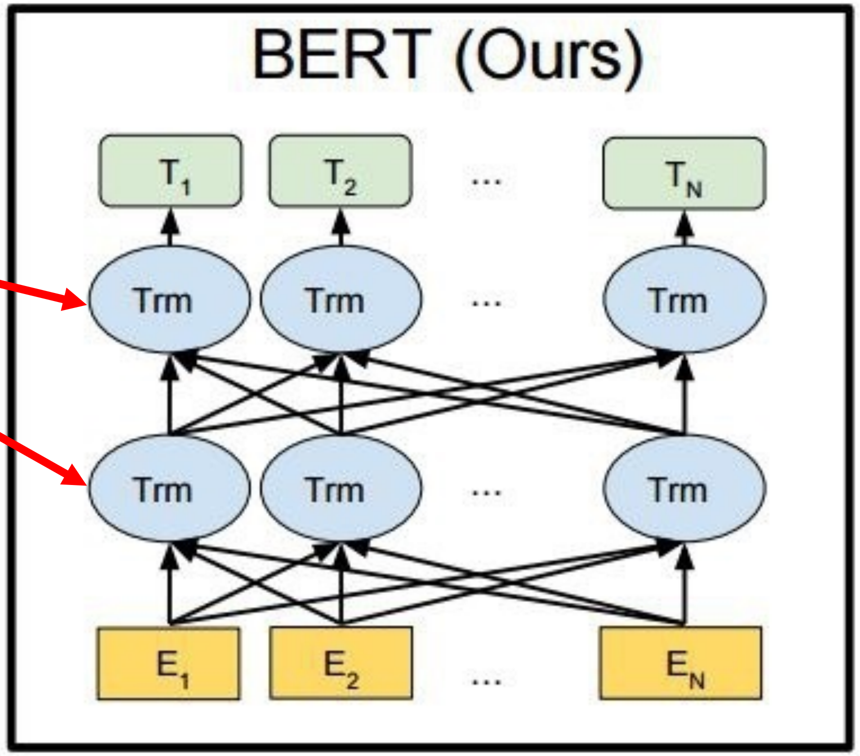
Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	91.3	97.5	91.7	94.5	51.2
2	ERNIE Team - Baidu	ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	91.0	96.6	90.9	94.5	51.7
3	Alibaba DAMO NLP	StructBERT		90.3	75.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.9	90.7	96.4	90.2	94.5	49.1
4	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9	92.8	94.5	53.1
5	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
6	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3	90.8	95.8	89.8	91.8	50.7
7	Huawei Noah's Ark Lab	NEZHA-Large		88.7	67.4	97.2	93.2/91.0	92.2/91.6	74.1/90.2	90.8	90.2	95.7	88.5	93.2	45.0
8	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	90.7	95.6	88.7	89.0	50.1
9	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0	49.3
10	Facebook AI	RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	95.4	88.2	89.0	48.7

Support BERT
-> Support All

[1] *https://gluebenchmark.com*

Overall

- General architecture
  - Multiple Transformer encoders
  - Input: Embeddings of words
  - Output: Hidden representations of words
- Downstream task
  - e.g., sentence classification
- How BERT works
  - Pre-training
    - The model is trained on unlabeled data over different pre-training tasks.
  - Fine-tuning
    - The model is first initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data from the downstream tasks.



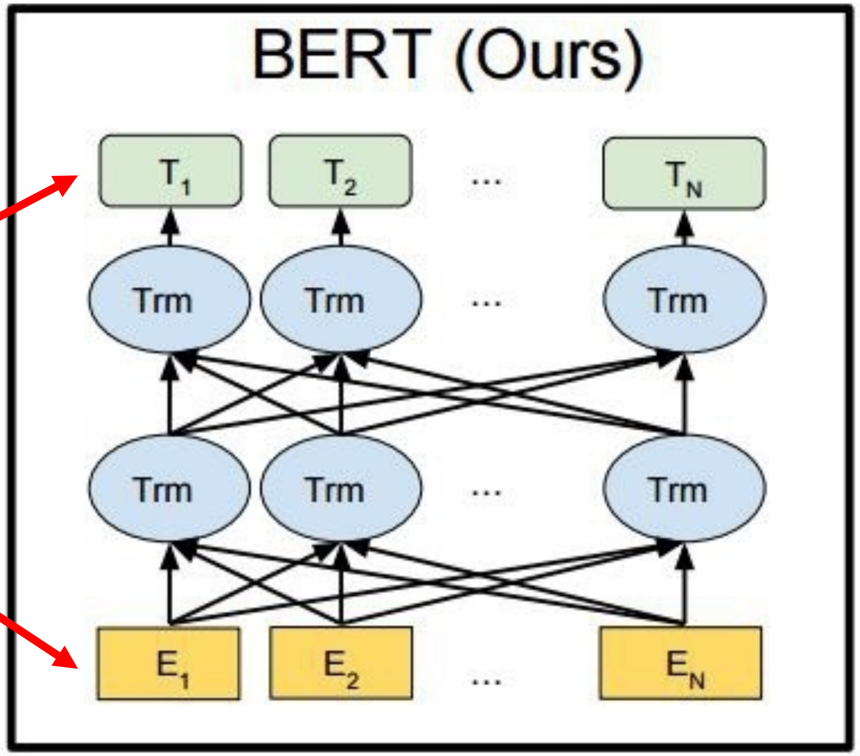
Architecture of BERT

Overall

- General architecture
  - Multiple Transformer encoders
  - Input: Embeddings of words
  - Output: Hidden representations of words

- Downstream task
  - e.g., sentence classification

- How BERT works
  - Pre-training
    - The model is trained on unlabeled data over different pre-training tasks.
  - Fine-tuning
    - The model is first initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data from the downstream tasks.

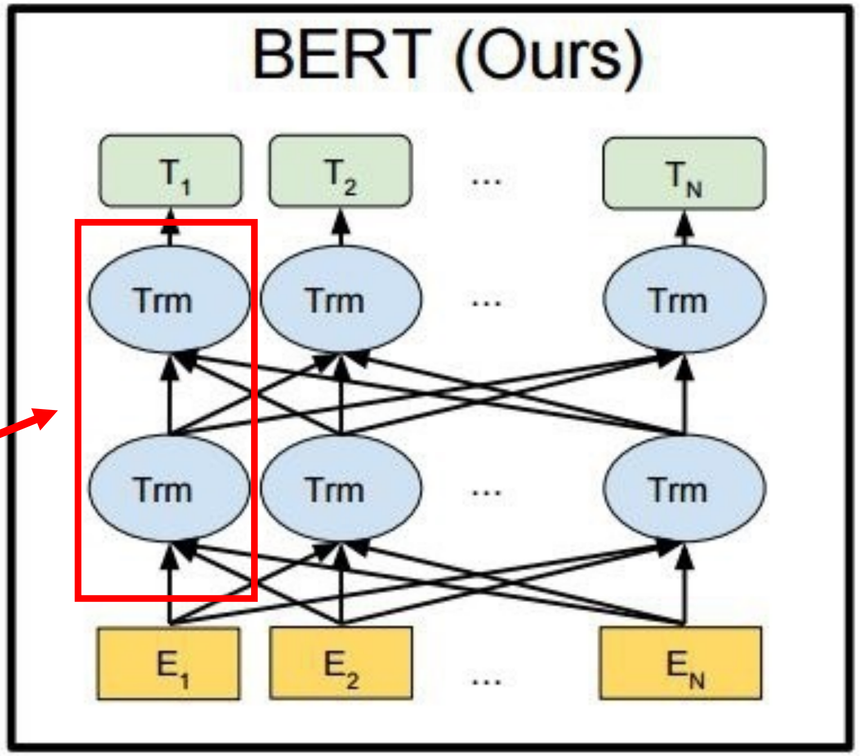


Architecture of BERT

Overall

- General architecture
  - Multiple Transformer encoders
  - Input: Embeddings of words
  - Output: Hidden representations of words
- Downstream task
  - e.g., sentence classification

1. Only one series of encoders
2. Shared by all word embeddings



Architecture of BERT

- How BERT works
  - Pre-training
    - The model is trained on unlabeled data over different pre-training tasks.
  - Fine-tuning
    - The model is first initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data from the downstream tasks.

Overall

- General architecture
  - Multiple Transformer encoders
  - Input: Embeddings of words
  - Output: Hidden representations of words

- Downstream task
  - e.g., sentence classification

- How BERT works
  - Pre-training
    - The model is trained on unlabeled data over different pre-training tasks.
  - Fine-tuning
    - The model is first initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data from the downstream tasks.

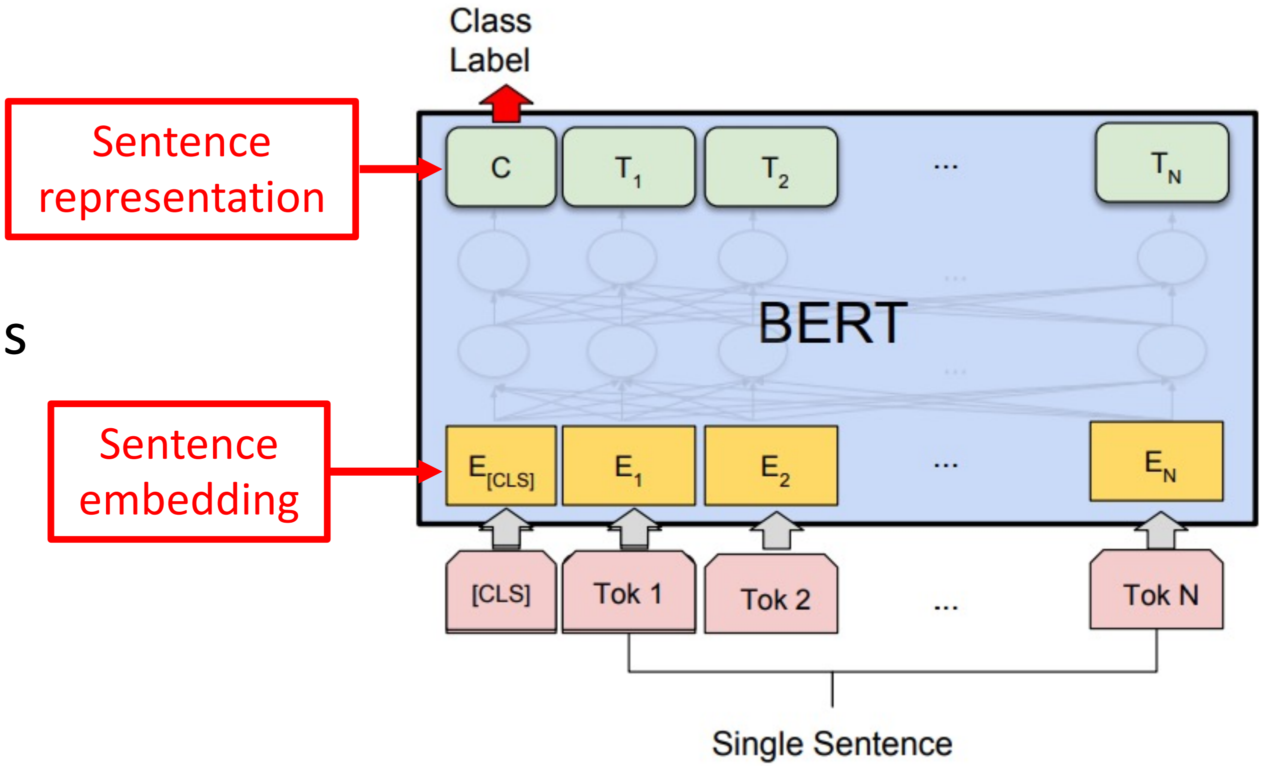


Illustration of BERT on classification

Overall

- General architecture
  - Multiple Transformer encoders
  - Input: Embeddings of words
  - Output: Hidden representations of words

- Downstream task
  - e.g., sentence classification

- How BERT works
  - Pre-training
    - The model is trained on unlabeled data over different pre-training tasks.
  - Fine-tuning
    - The model is first initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data from the downstream tasks.

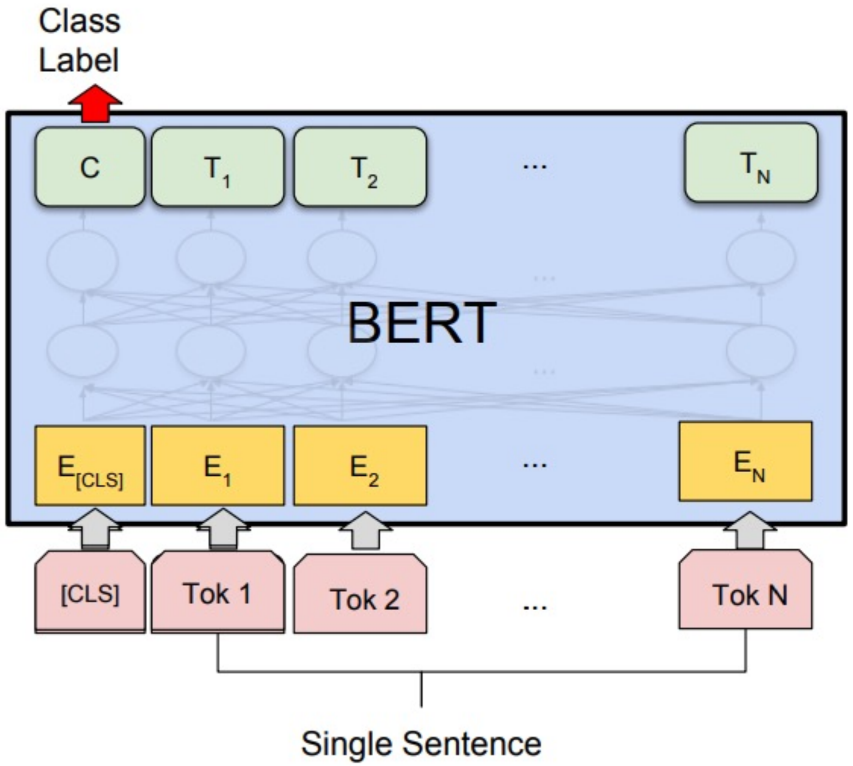
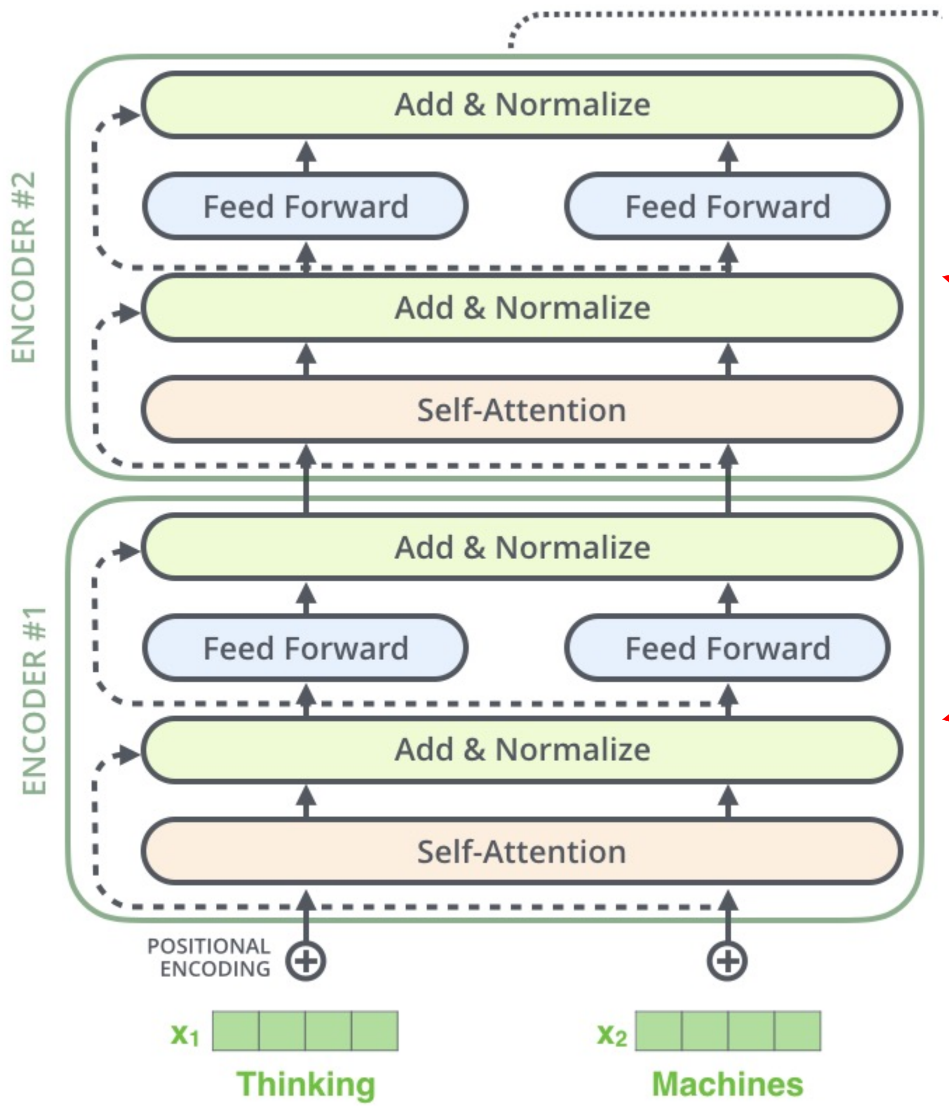


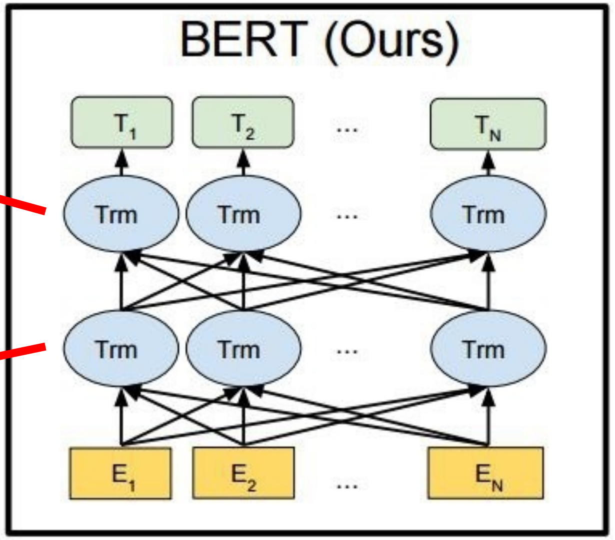
Illustration of BERT on classification

Transformer Encoder

Outputs



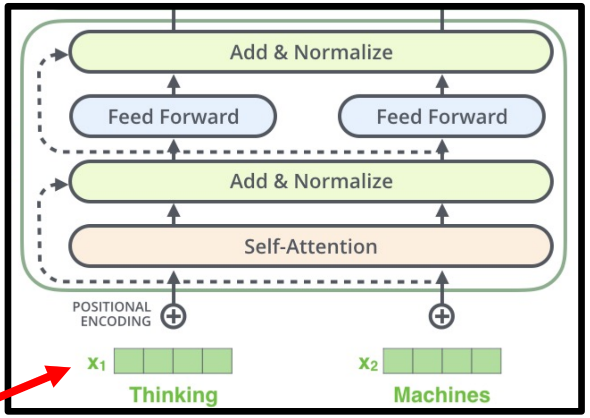
Inputs



Architecture of Encoder (Two Encoders Here)

Embedding Look_Up

- Embedding is the element-wise sum of three embeddings
- Goal
  - To give the model a sense of the order of the words



Architecture of Transformer Encoder

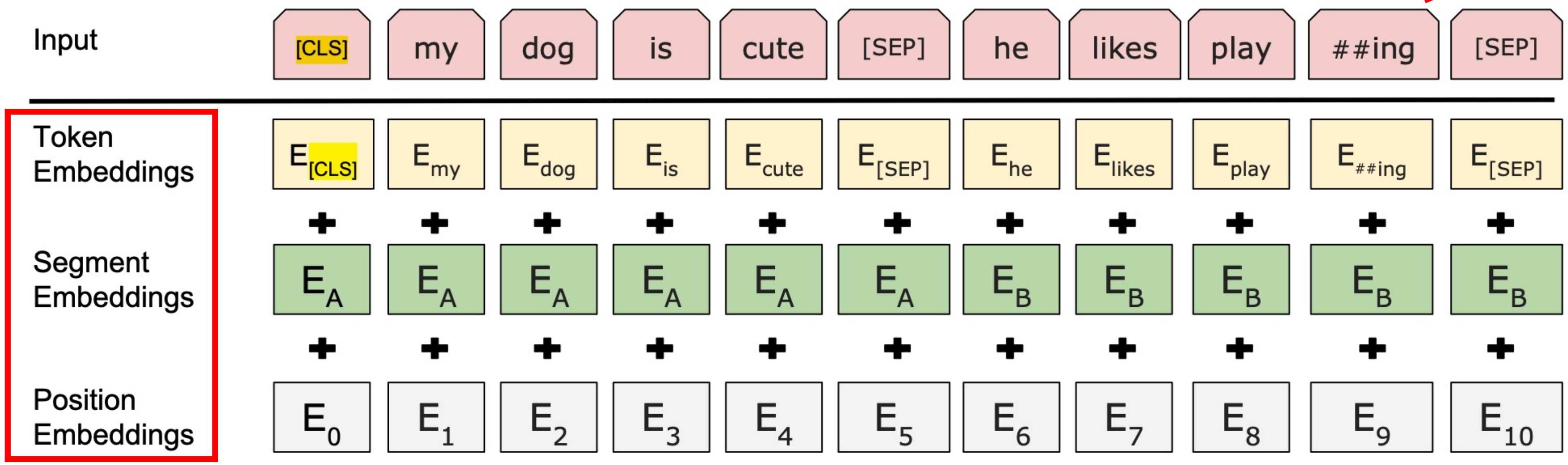


Illustration of BERT Input Representation

Embedding Look_Up

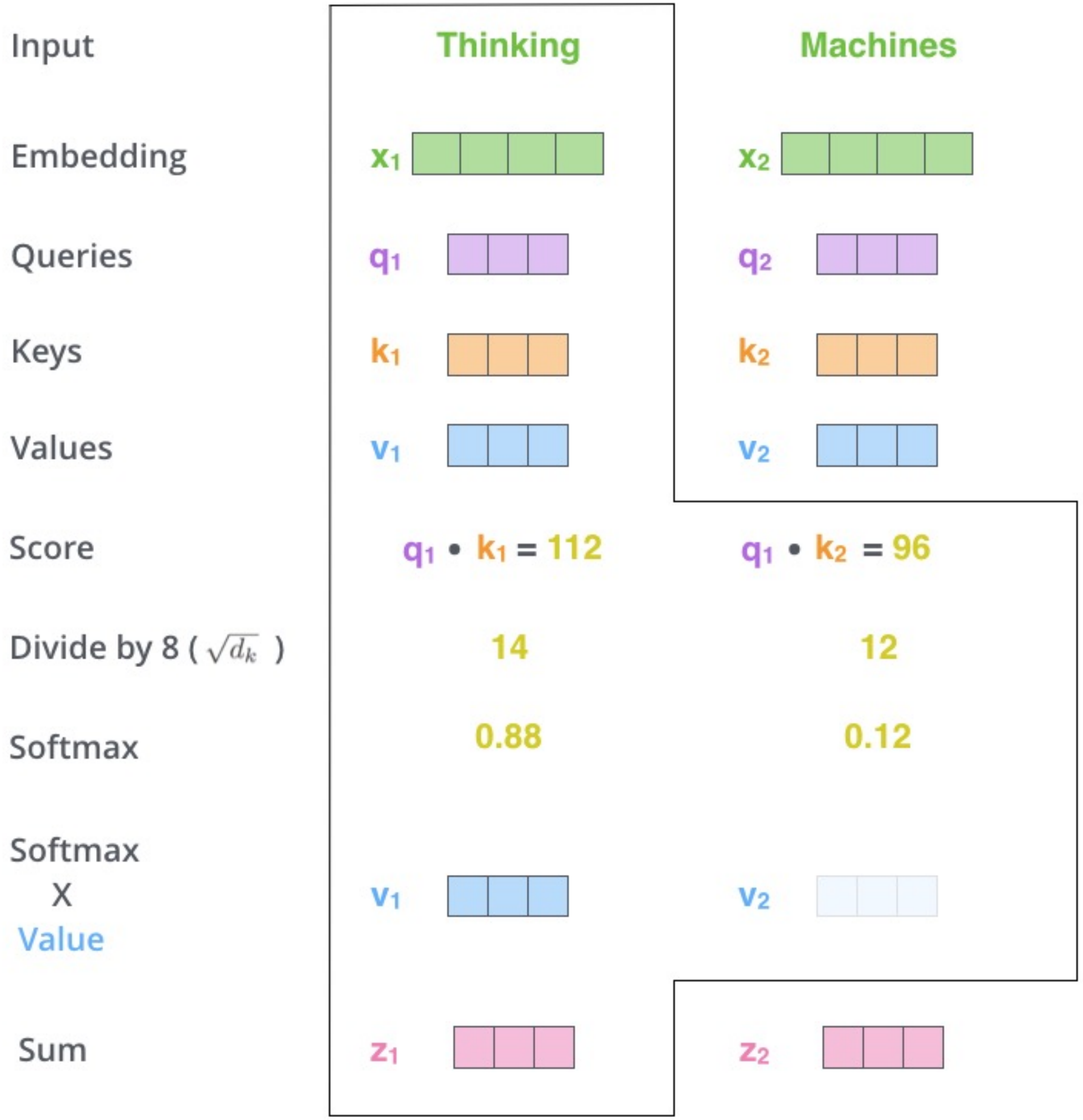
- Huge embedding table lookup

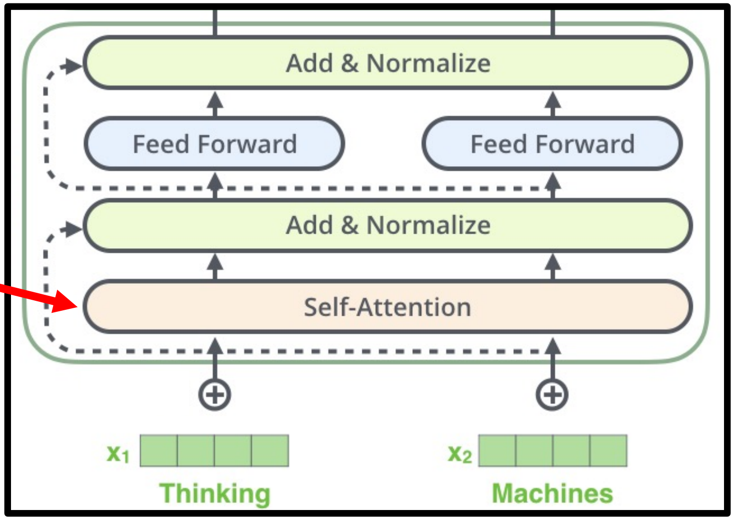
```python
class BertEmbeddings(nn.Module):
        """Construct the embeddings from word, position and token_type embeddings.
        """
        def __init__(self, config):
            super(BertEmbeddings, self).__init__()
            self.word_embeddings = nn.Embedding(config.vocab_size, config.hidden_size)
            self.position_embeddings = nn.Embedding(config.max_position_embeddings, config.hidden_size)
            self.token_type_embeddings = nn.Embedding(config.type_vocab_size, config.hidden_size)
```

- nn.Embedding( )
    - A simple lookup table that stores embeddings
    - All the elements are parameters
    - Input is a list of indices, and output is the word embeddings
- Implementation links
    - PyTorch: https://pytorch.org/docs/stable/_modules/torch/nn/modules/sparse.html#Embedding
    - TensorFlow: https://github.com/tensorflow/tensorflow/blob/v2.2.0/tensorflow/python/ops/embedding_ops.py#L329-L373

Self-Attention

- Take two words as an example

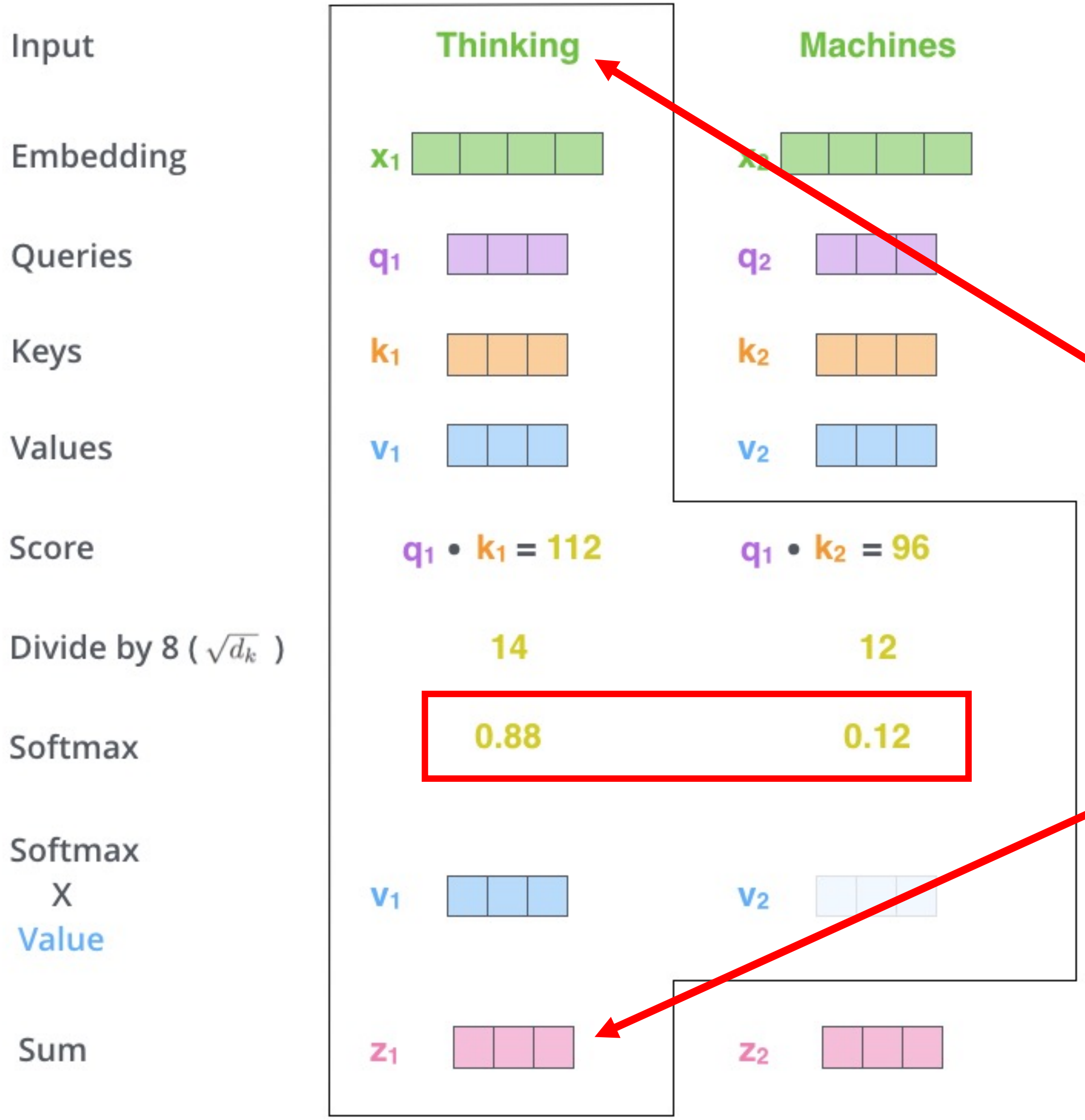


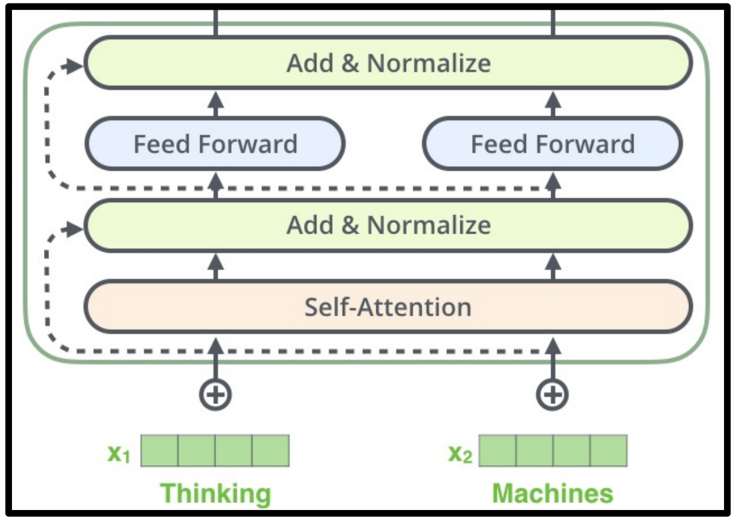


Architecture of Transformer Encoder

Self-Attention

- Take two words as an example





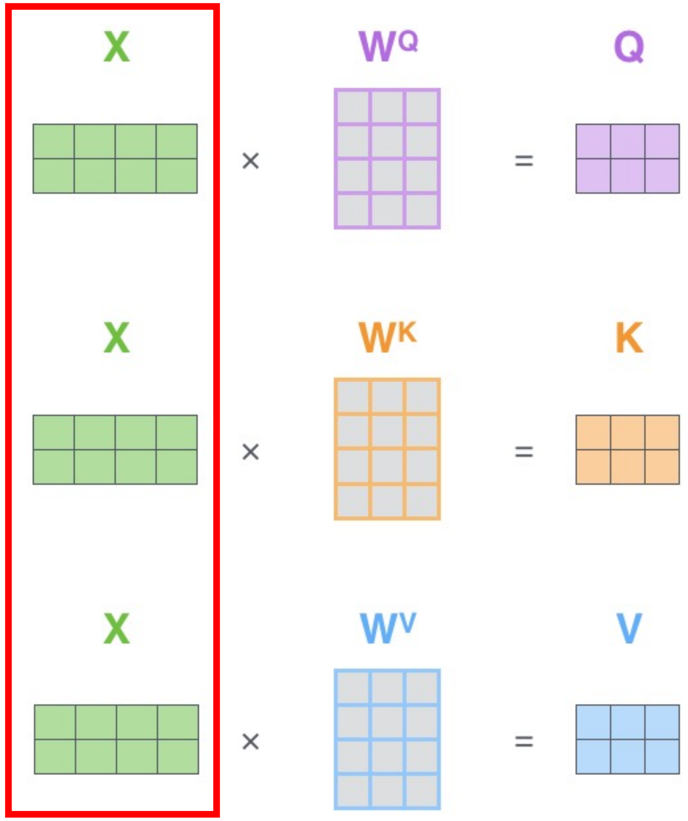
Architecture of Transformer Encoder

Input: Word embedding vectors

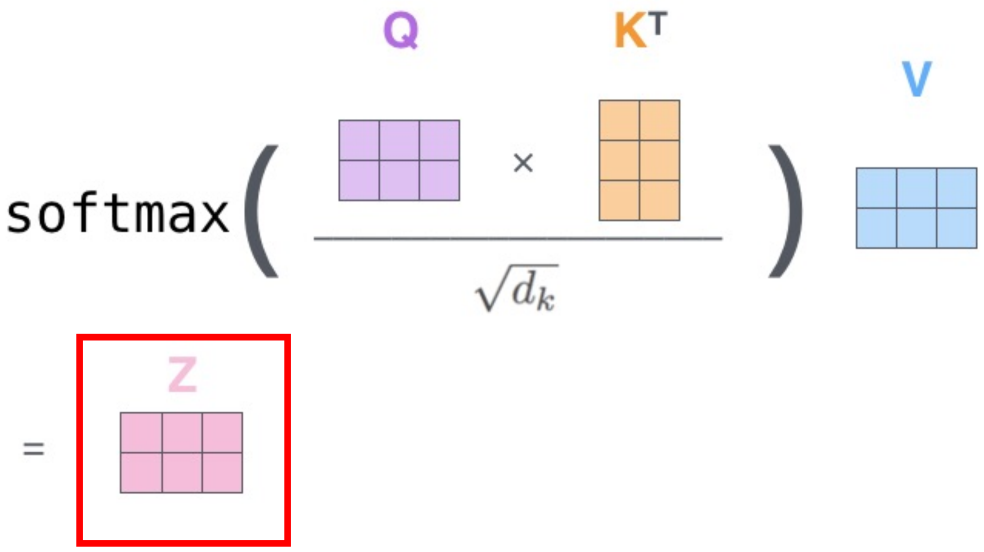
Output: New vector representations

Self-Attention in Matrix Calculation

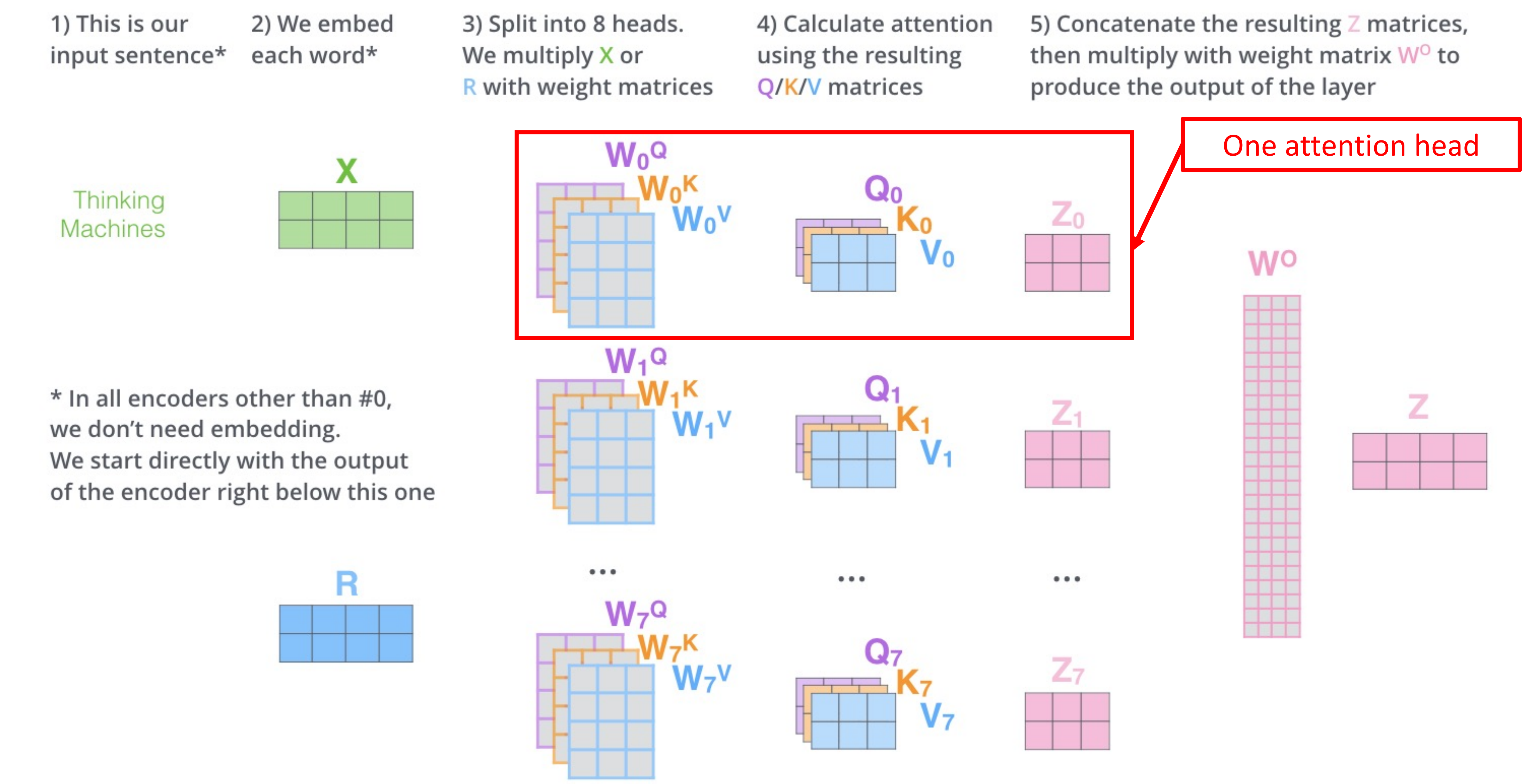
- First step to calculate ***Q, K, V***
  - Rows: ***words of a sentence***
  - Columns: ***hidden_dims***



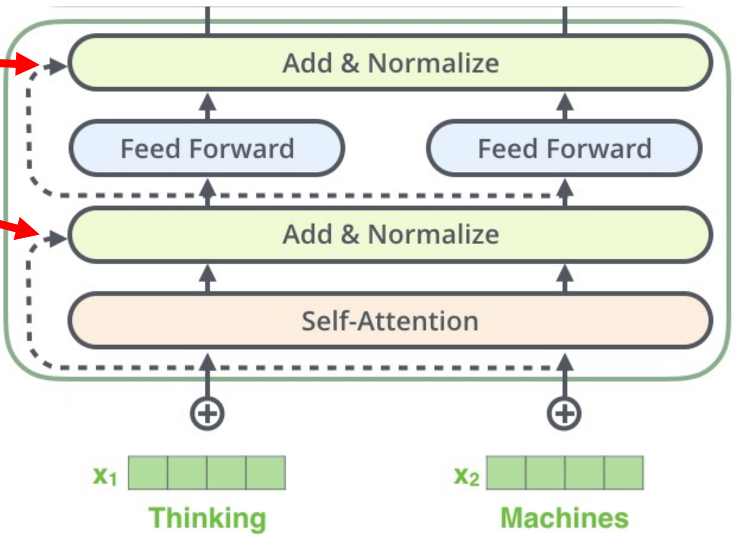
- Second step to calculate the output ***Z***
  - Rows: ***words of a sentence***
  - Columns: ***hidden_dims***



Multi-Head Self-Attention



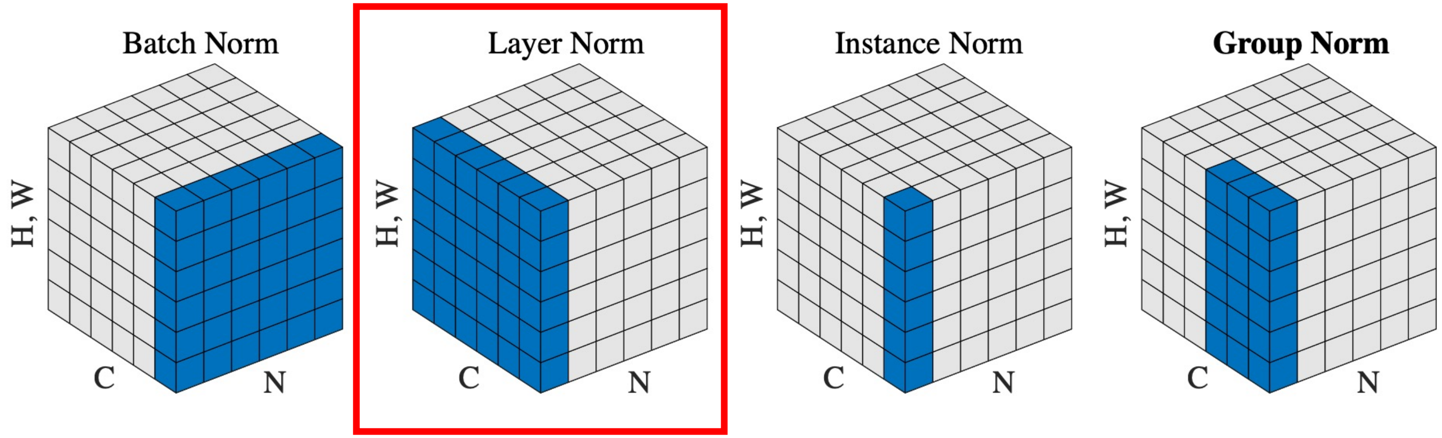
Layer Normalization [1]



- Motivations
  - 1. Dynamic length
  - 2. Different meaning of the same position
- Formula

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

- Comparison between four normalizations [2]



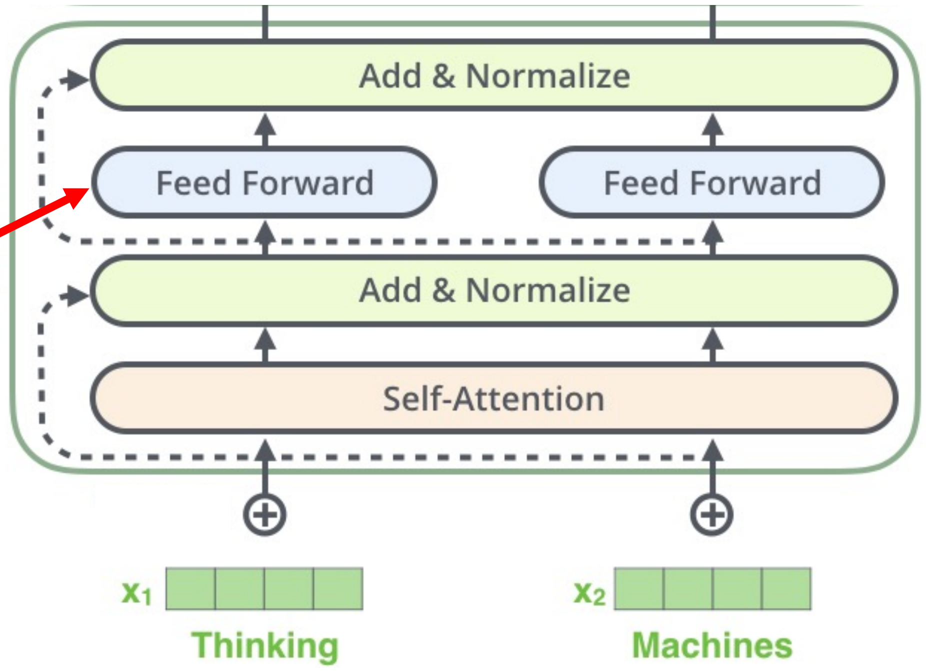
[1] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.
[2] Wu, Y., & He, K. (2018). Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 3-19).

Feed-Forward Networks

- Architecture
  - Two linear transformations with a ReLU in between
  - Dimension of input and output =512
  - Dimension of inner-layer = 512*4

- Formula

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



Other Transformer Based NLP Models

- **XLNet (CMU + Google AI)**
  - Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems* (pp. 5754-5764).
- **ALBERT (Google Language)**
  - Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.
- **RoBERTa (Facebook AI)**
  - Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
- **Transformer-XL (CMU + Google Brain)**
  - Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.
- **ERNIE (Baidu)**
  - Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., ... & Wu, H. (2019). Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.
- **GPT-2 (OpenAI)**
  - Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 9.

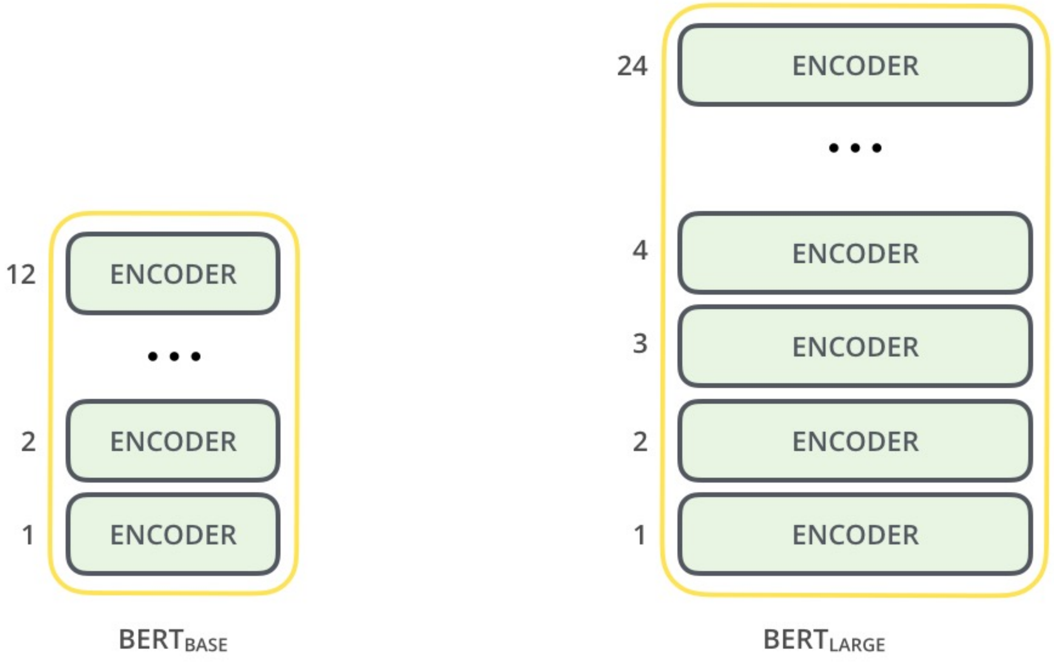
Support BERT -> Support All

Agenda

- BERT Architecture
- Model Compression
- Applications

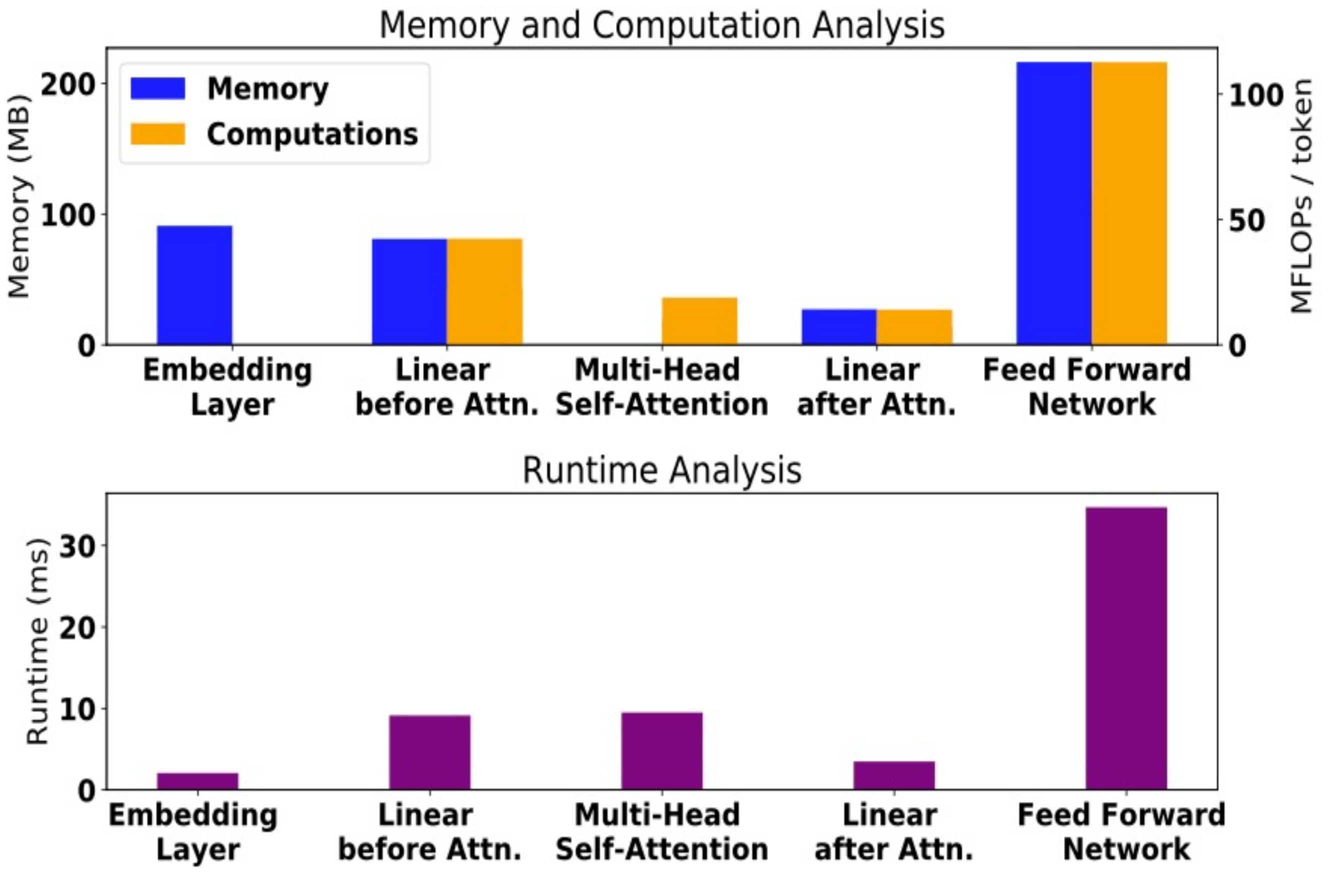
$\text{BERT}_{\text{Base}}$ *v.s.* $\text{BERT}_{\text{Large}}$

- $\text{BERT}_{\text{Base}}$
  - #para = 110M, #encoders = 12, #dim = 768, #head = 12,
  - #FLOPs = 123M * sentence_length * #Batch
- $\text{BERT}_{\text{Large}}$
  - #para = 340M, #encoders = 24, #dim = 1024, #head = 16
  - #FLOPs = 1857M * sentence_length * #Batch



Comparison between BERT_based and BERT_Large

Parameter / FLOPs Distributions



Computation Analysis [1]

[1] BGanesh, P., etc. (2020). Compressing large-scale transformer-based models: A case study on bert. arXiv preprint arXiv:2002.11985.

Compression Methods on BERT

- Data Quantization
  - Embedding layer is more sensitive to quantization than other layers
  - More bits to maintain its accuracy

- Pruning
  - **Sparse pruning** (trending and with promising future):
  - Our NAACL'21 work [1], **SparseBERT**, achieved SOTA (**compression ratio = x20, only 1.4% performance drop**)
  - Structured pruning: #encoders, #att_heads, #hidden_dims

- Knowledge Distillation
  - Distillation on output logits, encoder outputs, attention maps

- Architecture-Invariant Compression
  - Parameter sharing, weight matrix decomposition

[1] Dongkuan Xu, Ian En-Hsu Yen, Jinxi Zhao, Zhibin Xiao. Rethinking Network Pruning -- under the Pre-train and Fine-tune Paradigm. *NAACL 2021.*

All The Ways You Can Compress BERT [1]

- Literatures

Paper	Prune	Factor	Distill	W. Sharing	Quant.	Pre-train	Downstream
Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning	☑					☑	☑
Are Sixteen Heads Really Better than One?	☑						☑
Pruning a BERT-based Question Answering Model	☑						☑

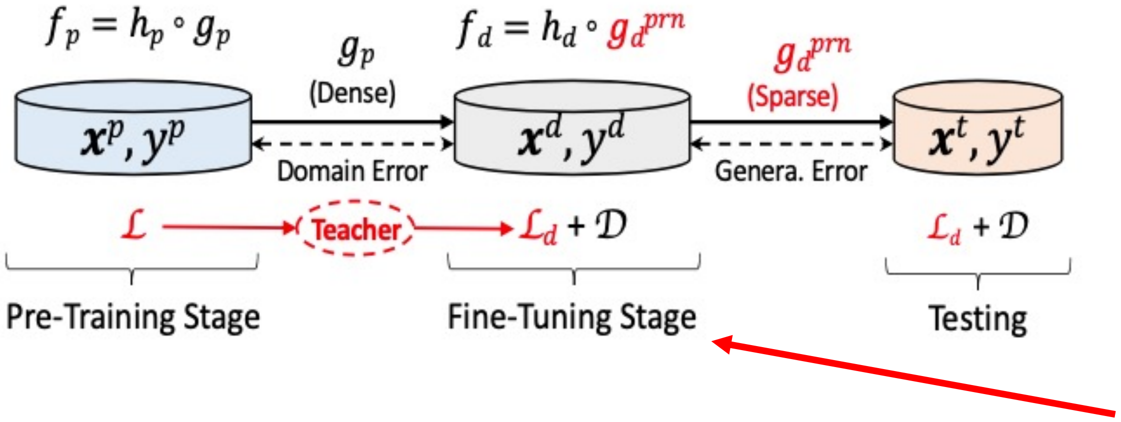
- Experimental Results

Paper	Reduction	Of	Speed-up	Accuracy?	Comments
Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning	30%	params	?	Same	Some interesting ablation experiments and fine-tuning analysis
Are Sixteen Heads Really Better than One?	50-60%	attn heads	1.2x	Same	
Pruning a BERT-based Question Answering Model	50%	attn Heads + FF	2x	-1.5 F1	

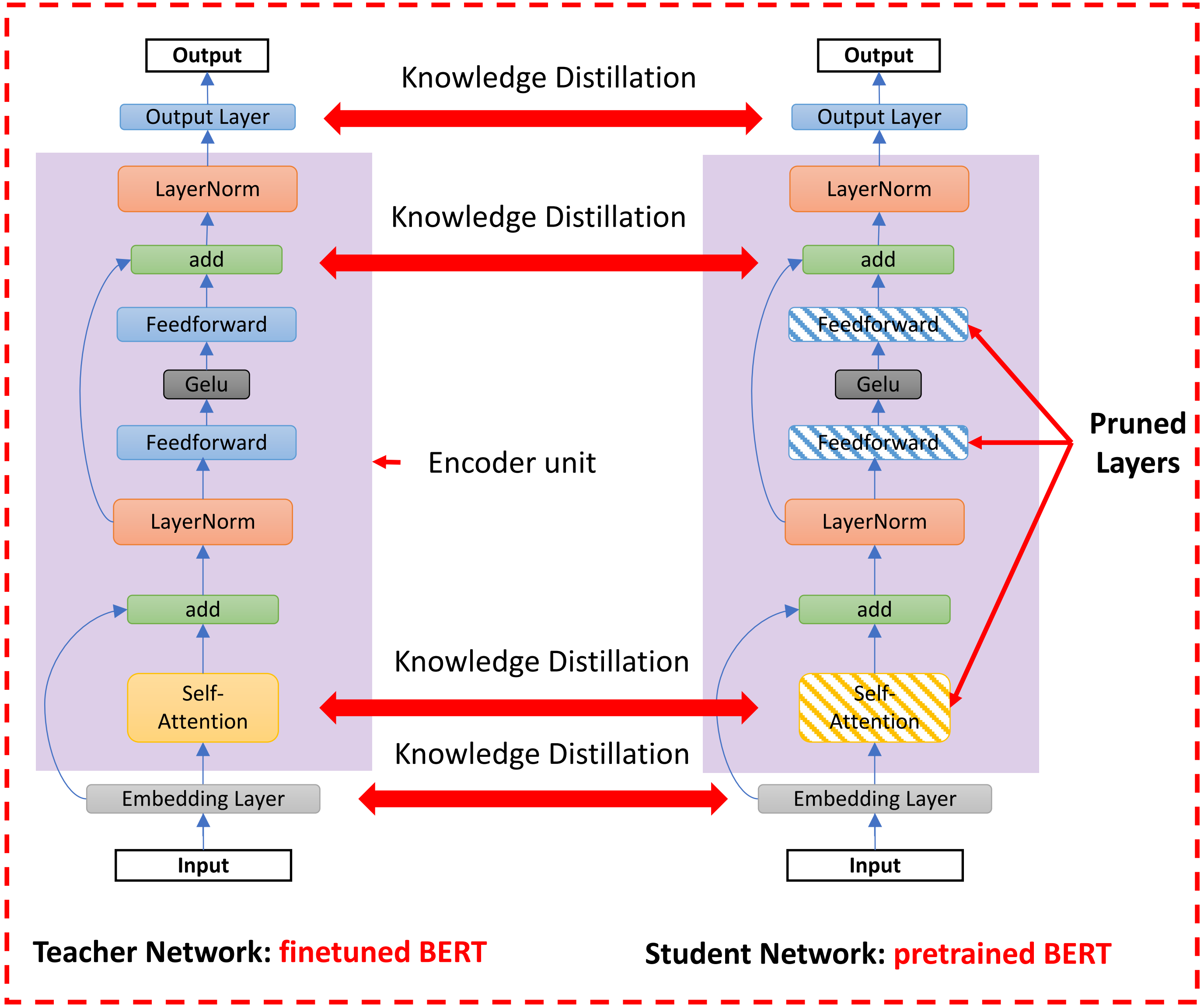
[1] http://mitchgordon.me/machine/learning/2019/11/18/all-the-ways-to-compress-BERT.html

SparseBERT: Knowledge-Aware Sparse Pruning [1]

- Motivation: gap of sparse pruning in NLP
- Analysis about knowledge lost



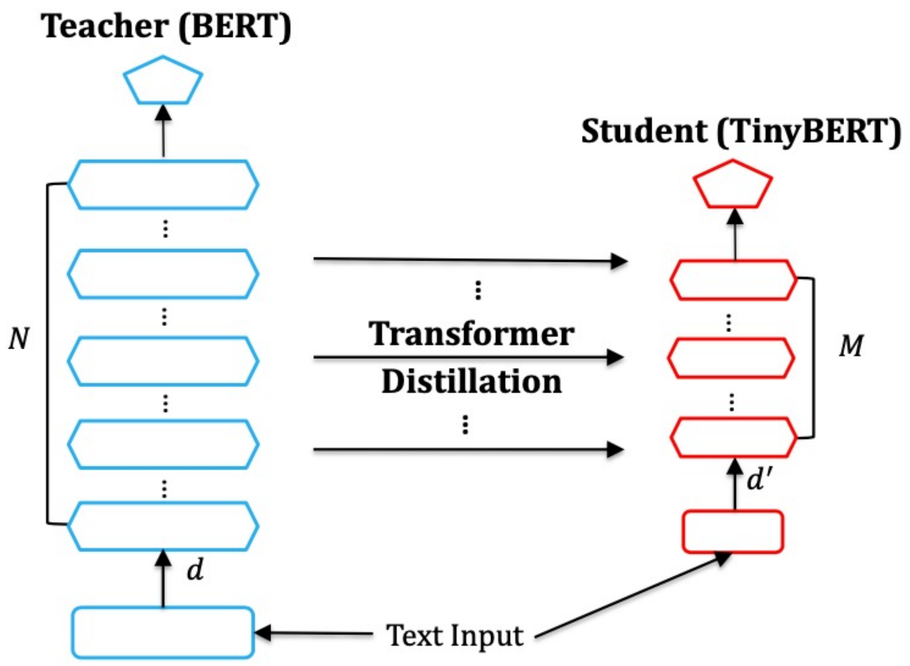
How is knowledge transferred and lost?



[1] Dongkuan Xu, Ian En-Hsu Yen, Jinxi Zhao, Zhibin Xiao. Rethinking Network Pruning -- under the Pre-train and Fine-tune Paradigm. *NAACL 2021.*

TinyBERT: Distilling BERT for Natural Language Understanding [1]

- Knowledge distillation of the Transformer-based models
- A two-stage learning framework
- Results
  - > 96% the performance of teachers
  - 7.5x smaller and 9.4x faster on inference



Overview of TinyBERT distillation

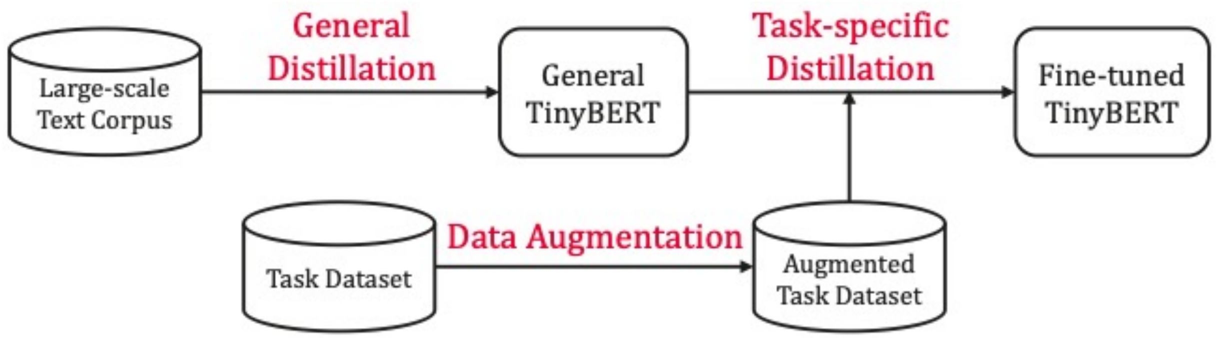


Illustration of TinyBERT learning

[1] Huawei Noah's Ark Lab. Tinybert: Distilling bert for natural language understanding. *EMNLP 2020.*

Agenda

- BERT Architecture
- Model Compression
- Applications

Novel CV Applications using Transformers [1]

🏞️ Transformer for Image Synthesis - 🔗 Esser et al. (2020)

🔵 Transformer for Multi-Object Tracking - 🔗 Sun et al. (2020)

🎶 Transformer for Music Generation - 🔗 Hsiao et al. (2021)

💃 Transformer for Dance Generation with Music - 🔗 Huang et al. (2021)

🔮 Transformer for 3D Object Detection - 🔗 Bhattacharyya et al. (2021)

🗿 Transformer for Point-Cloud Processing - 🔗 Guo et al. (2020)

⏰ Transformer for Time-Series Forecasting - 🔗 Lim et al. (2020)

👁️ Transformer for Vision-Language Modeling - 🔗 Zhang et al. (2021)

🛣️ Transformer for Lane Shape Prediction - 🔗 Liu et al. (2020)

🌠 Transformer for End-to-End Object Detection - 🔗 Zhu et al. (2021)

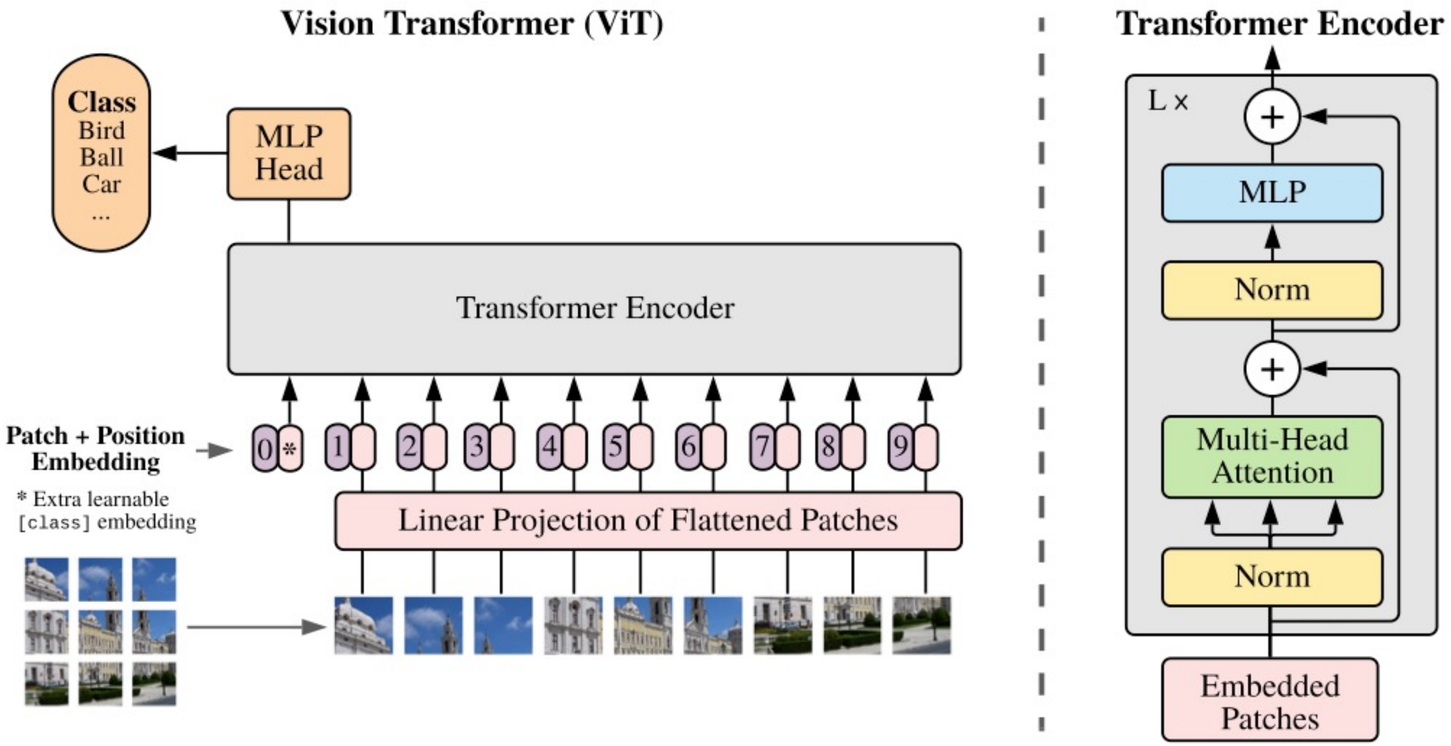
[1] https://paperswithcode.com/newsletter/3

NLP Applications using Transformers

- ❑ Question Answer and Reading Comprehension
- ❑ Web Search and Information Retrieval
- ❑ Dialog System or Chatbot
- ❑ Text Summarization
- ❑ Data Augmentation in NLP areas
- ❑ Text/Word Classification
- ❑ Sequence Labelling
- ❑ Others

Vision Transformer (ViT) [1]

- Open the door of applying Transformer architecture to CV domains
- Split an image into fixed-size patches
- Transformer encoder

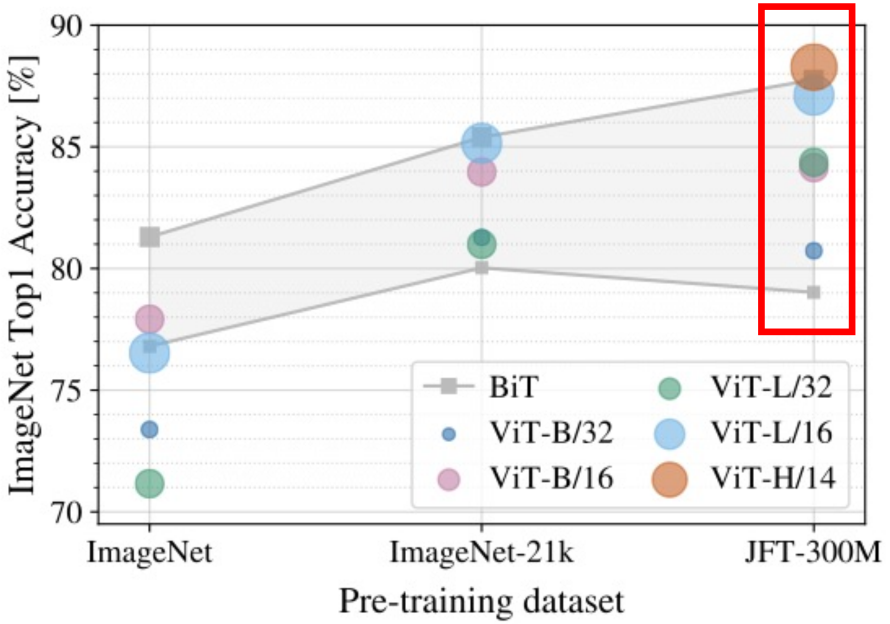


Model overview

[1] Google Brain. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR 2021* (oral).

Vision Transformer (ViT) [1]

- Vision Transformer >> SOTA convolutional networks
  - Need to be pre-trained on large data and transferred to smaller ones
- Substantially fewer computational resources to train



Results on ImageNet

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Results on popular image classification benchmarks

[1] Google Brain. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR 2021* (oral).

Q & A

Web: www.personal.psu.edu/dux19/

Email: dux19@psu.edu