Latest on Transformers: New Techniques after BERT

Yang You

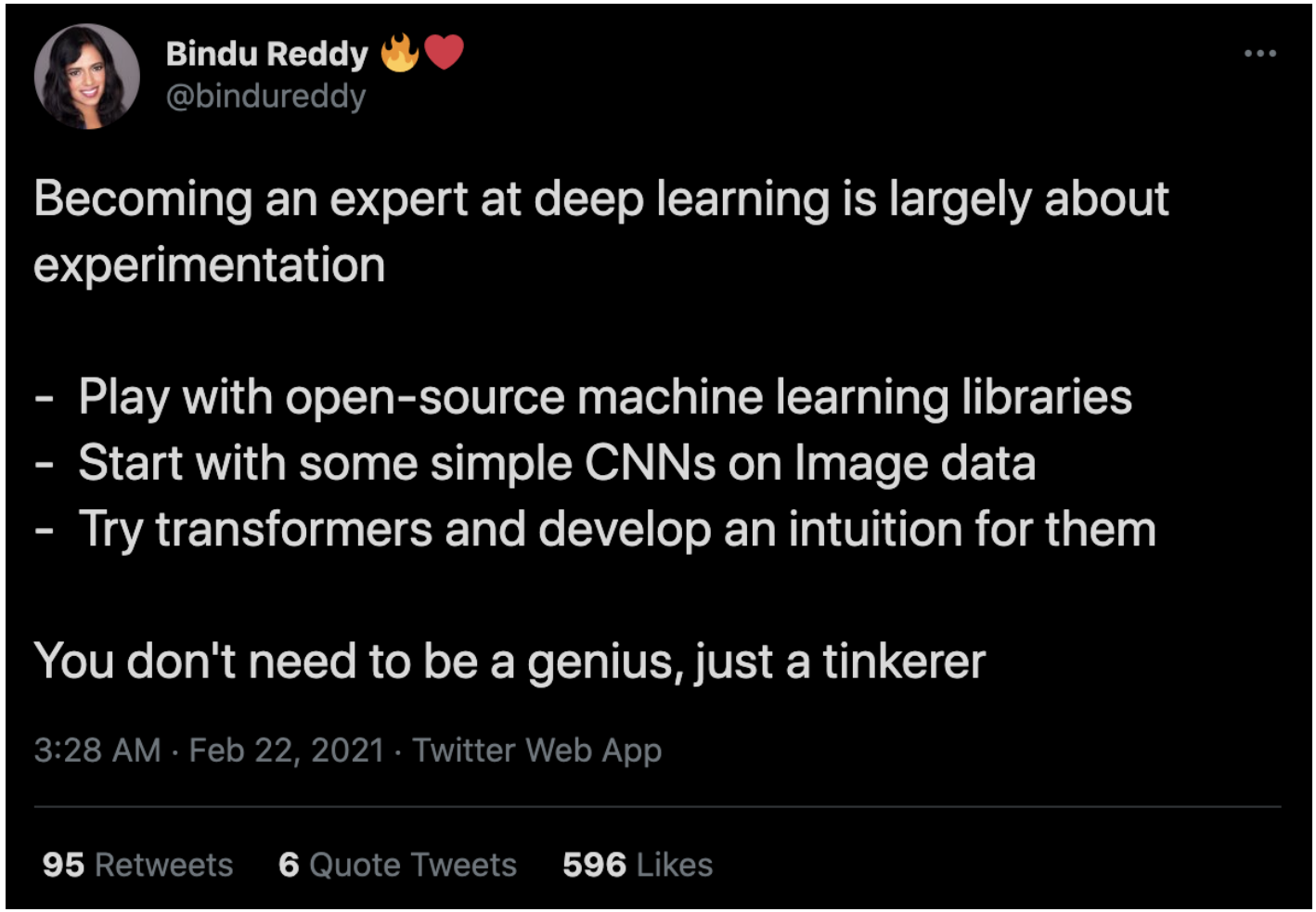
HPC-AI Lab of NUS Computer Science

ai.comp.nus.edu.sg

Outline

- **Review on Transformers**
- Transformers for Computer Vision
- ALBERT: an efficient BERT

Transformer is transforming everything in deep learning



- All the attention of deep learning is now on Attention!

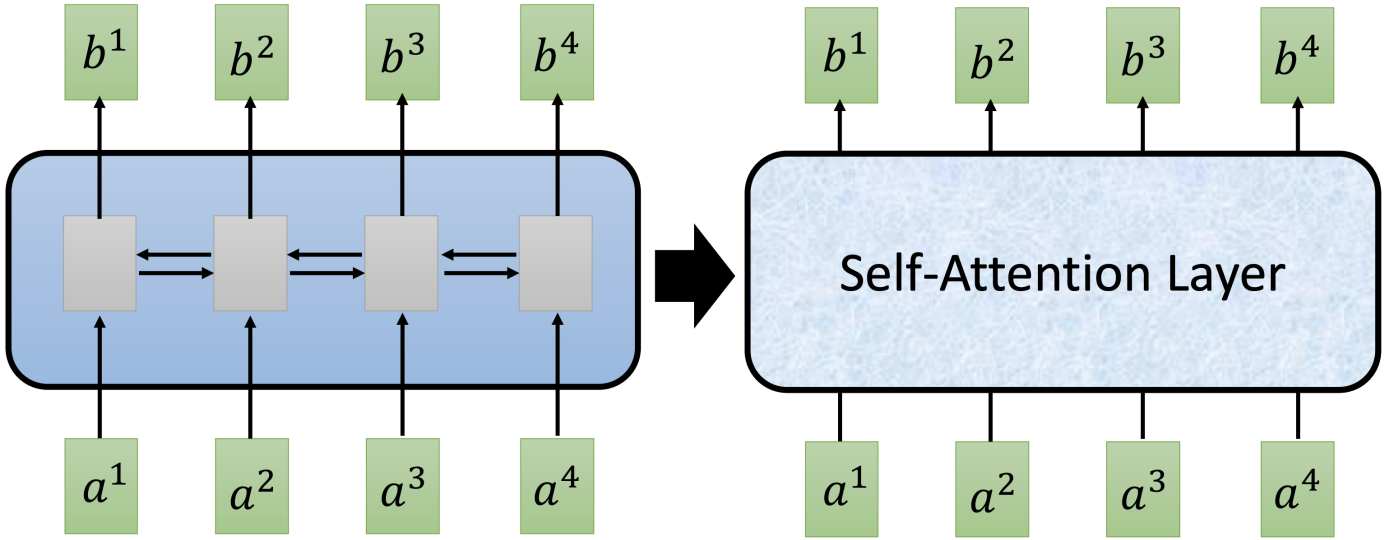
Why do we want to study Transformers?

- Transformers have become the default model-of-choice in NLP
  - The dominating approach is to pre-train on a huge dataset and then fine-tune on a smaller task-specific dataset
  - Belongs to self-attention-based architectures (Vaswani et al., 2017)
  - It is computationally efficient and scalable
    - With the models and datasets growing, there is still no sign of saturating performance
    - We now can train models of unprecedented size (>100B parameters)
    - e.g. BERT, GPT-3, PaLM, Switch-Transformers

Why do we want to study Transformers?

- Transformers are friendly to modern hardware like TPUs and GPUs
- Why?

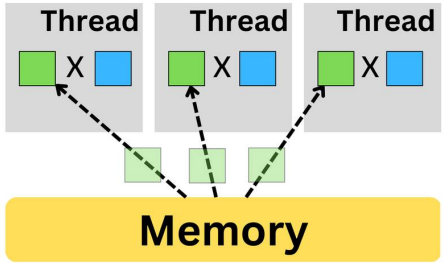
Self-Attention



- We can replace anything processed by RNN with Attention
  - In RNN, b^1, b^2, b^3, b^4 will be computed sequentially
  - In Attention, b^1, b^2, b^3, b^4 can be computed in parallel

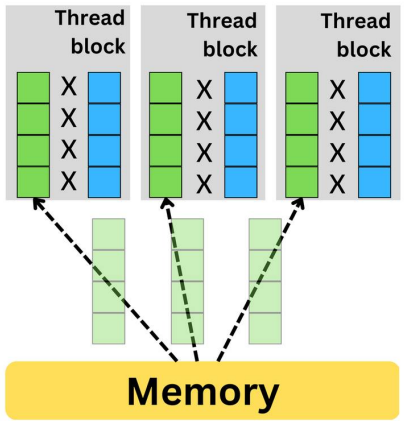
CPU vs GPU vs TPU

CPU



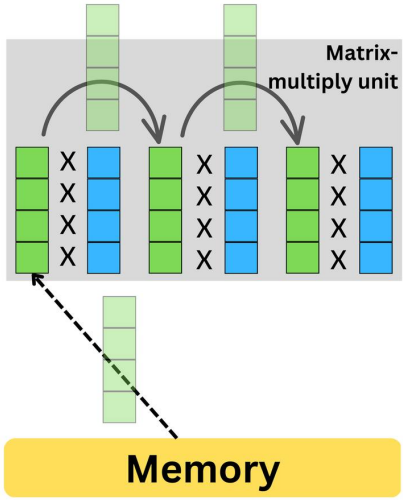
Parallelized scalar multiplication

GPU



parallelized vector multiplication

TPU

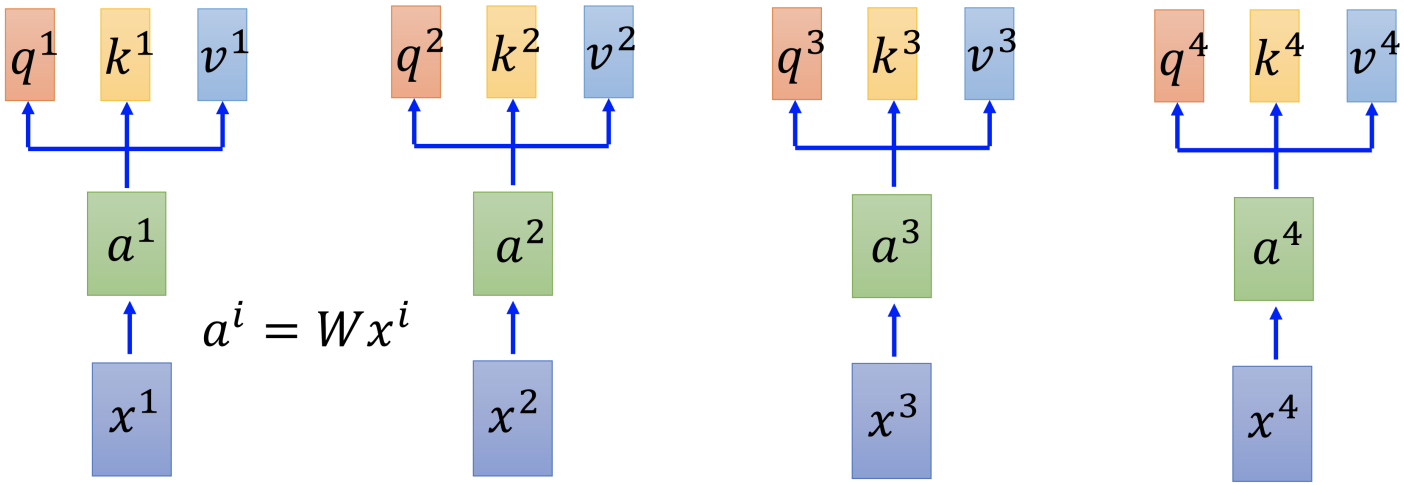


parallelized matrix multiplication

Why do we want to study Transformers?

- Transformers are friendly to modern hardware like TPUs and GPUs
  - We need a high parallelism to make full use of the computing units in TPUs and GPUs (e.g. thousands of threads)

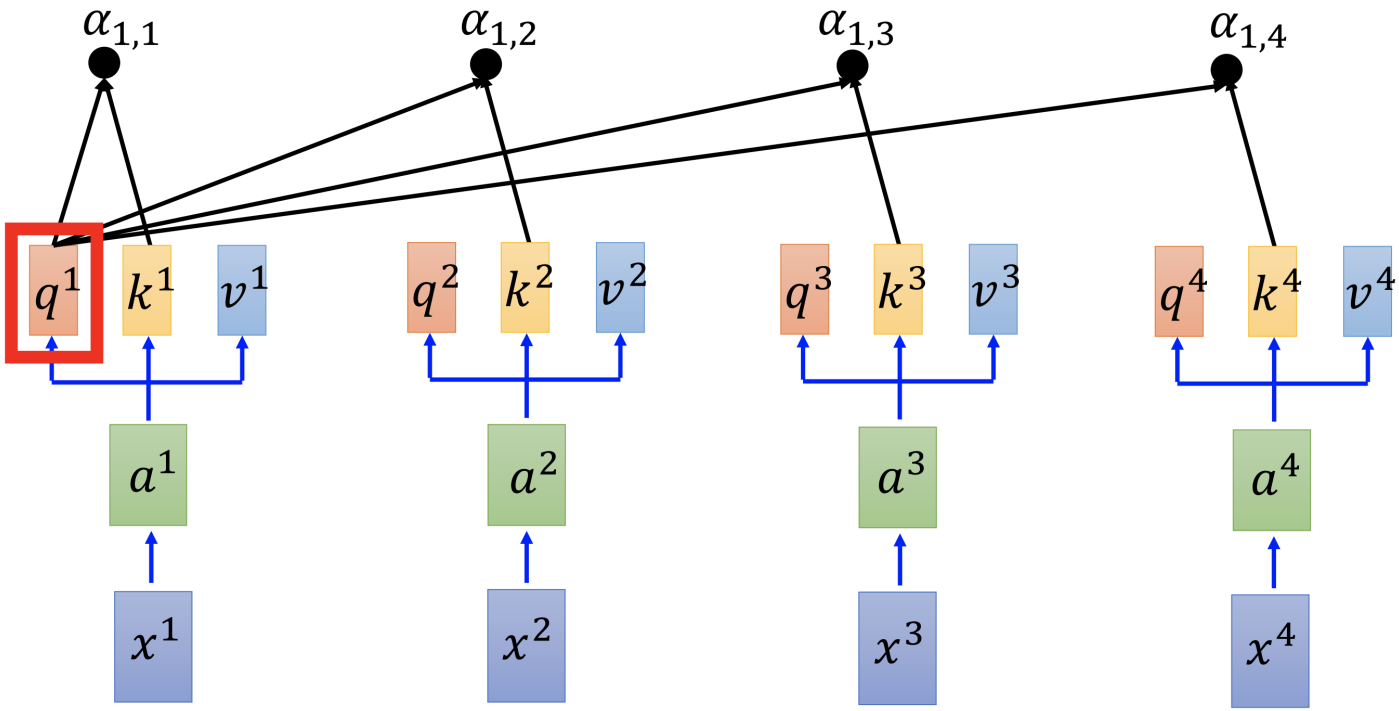
Self-Attention Architecture



- q: query (to match others)
  - $q^i = W^q a^i$
- k: key (to be matched)
  - $k^i = W^k a^i$
- v: value (to be extracted)
  - $v^i = W^v a^i$

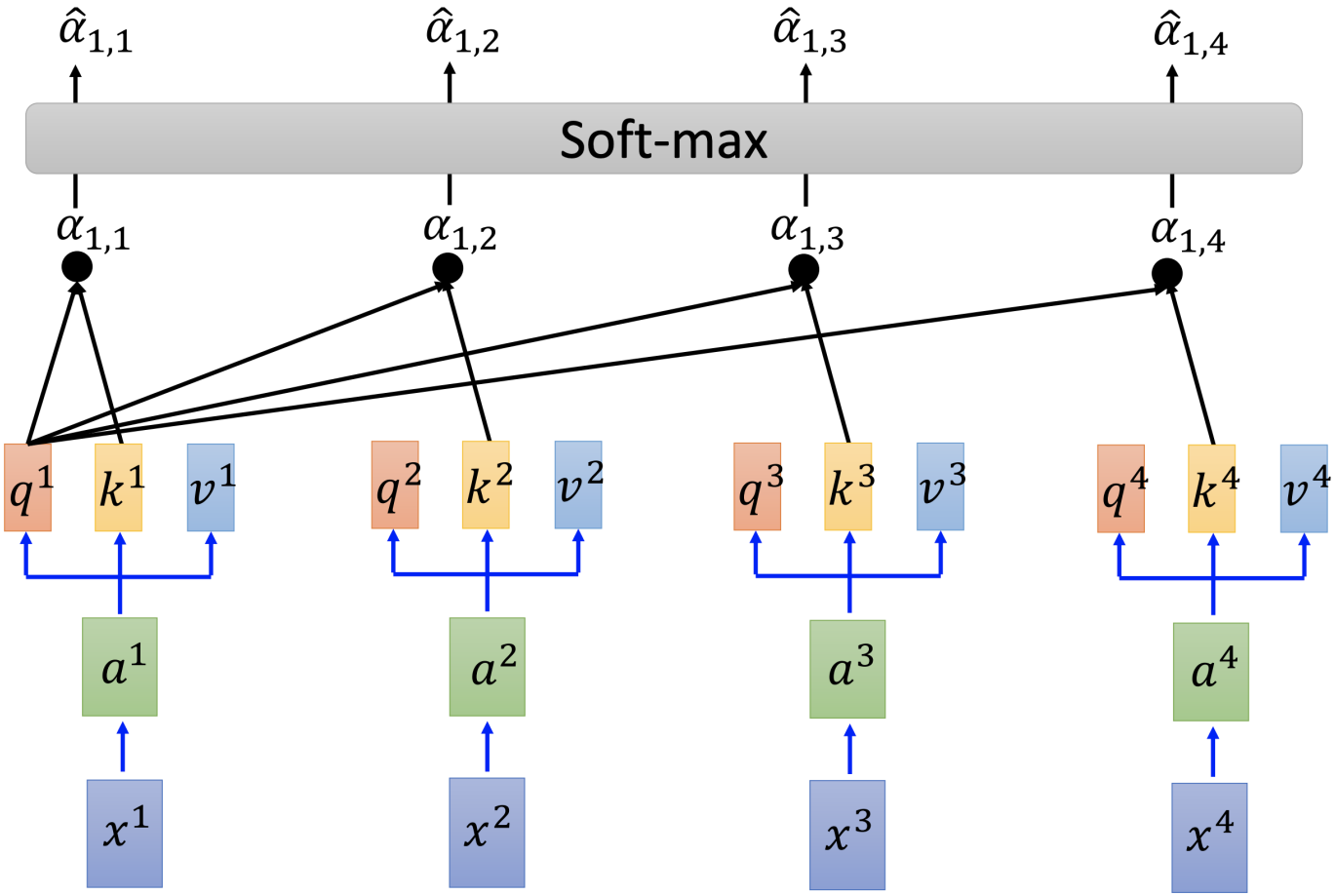
Computation of Self Attention

Scaled Dot-Product Attention: $\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\text{dot product}} / \sqrt{d}$



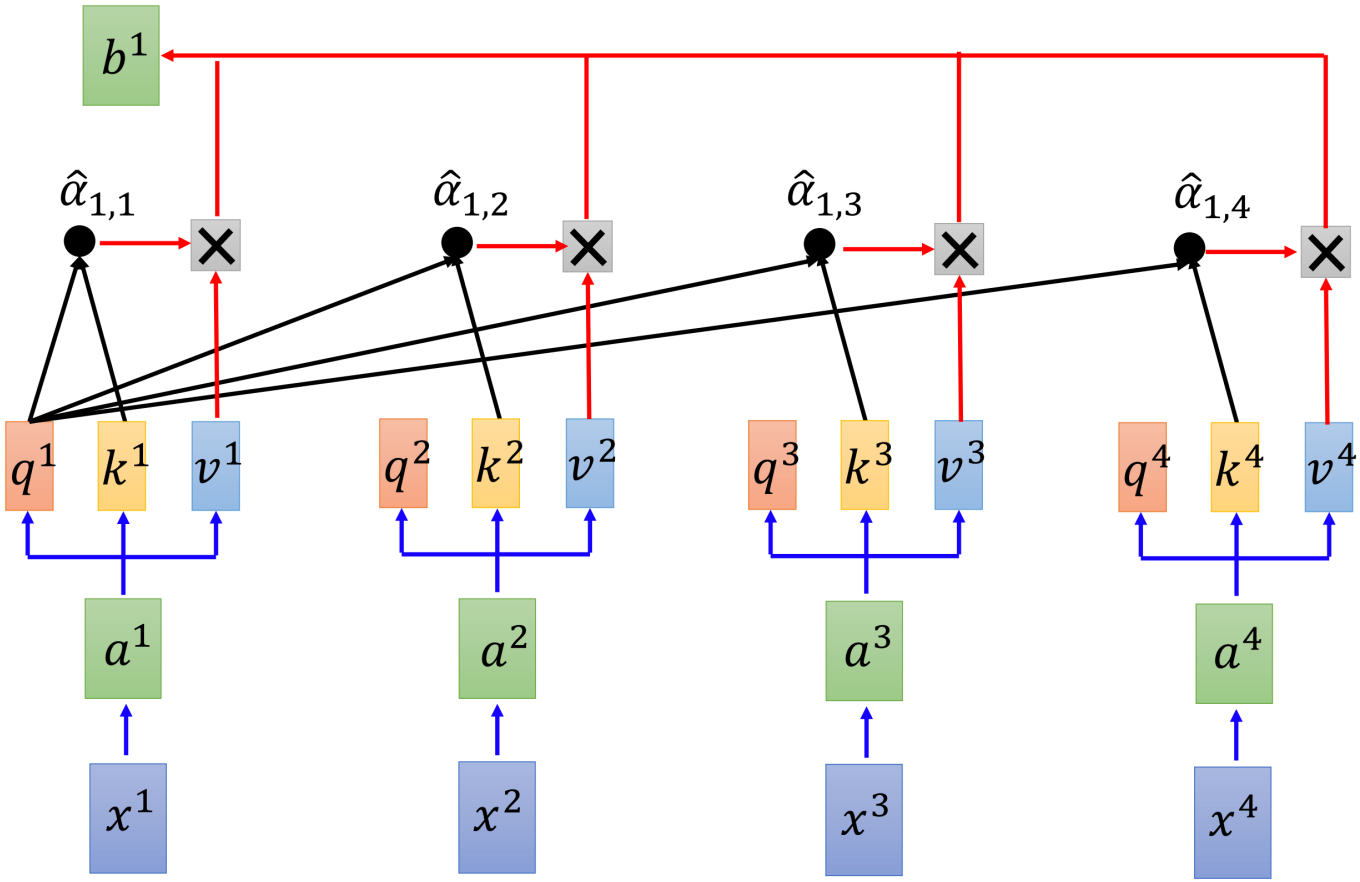
- d: the length of vector q or vector k
- We use each query q to match each key k
- $\alpha_{i,j}$: the strength of the relationship between x^i and x^j

Just doing a softmax



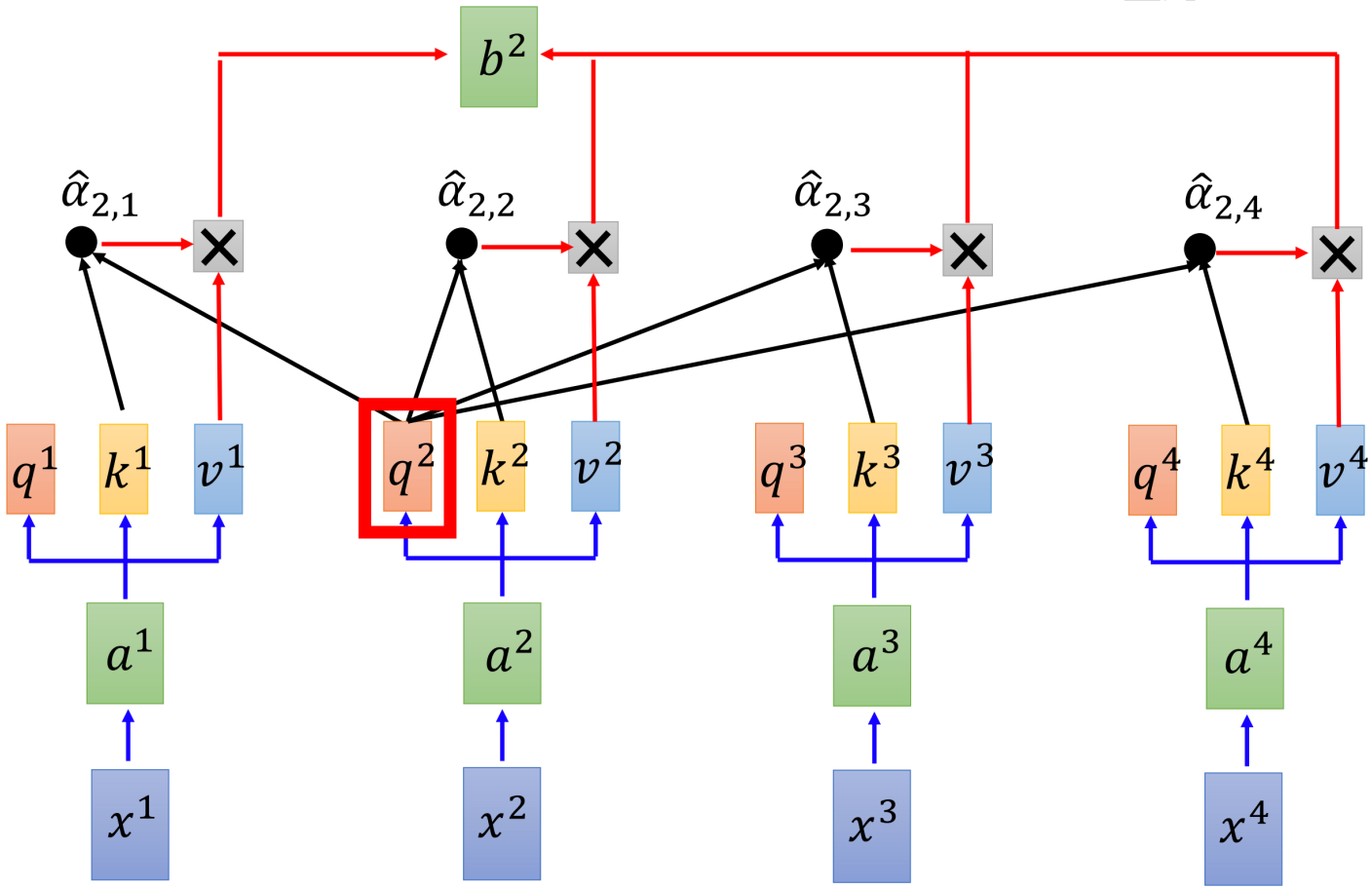
- $\hat{\alpha}_{1,i} = \frac{exp(\alpha_{1,i})}{\sum_j exp(\alpha_{1,j})}$: the strength of the relationship between x^1 and x^j
- $\hat{\alpha}_{i,j} \times v^j$: the attention of x^i on x^j
  - stronger relationship, stronger attention

Computing the first output



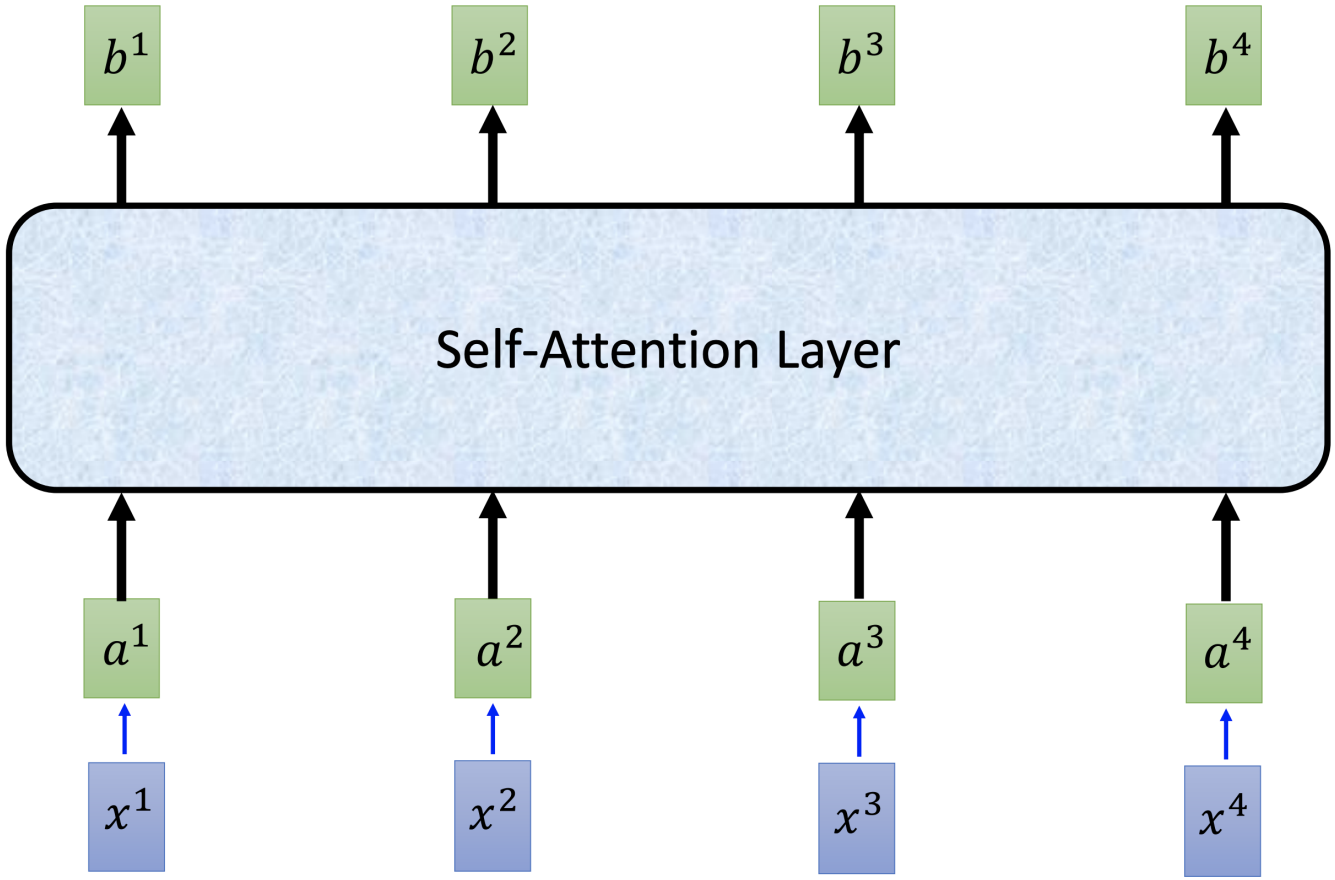
- $b^1 = \sum_i \hat{\alpha}_{1,i} \times v^i$: all the attentions of x^1 on other inputs and itself

Computing the second output



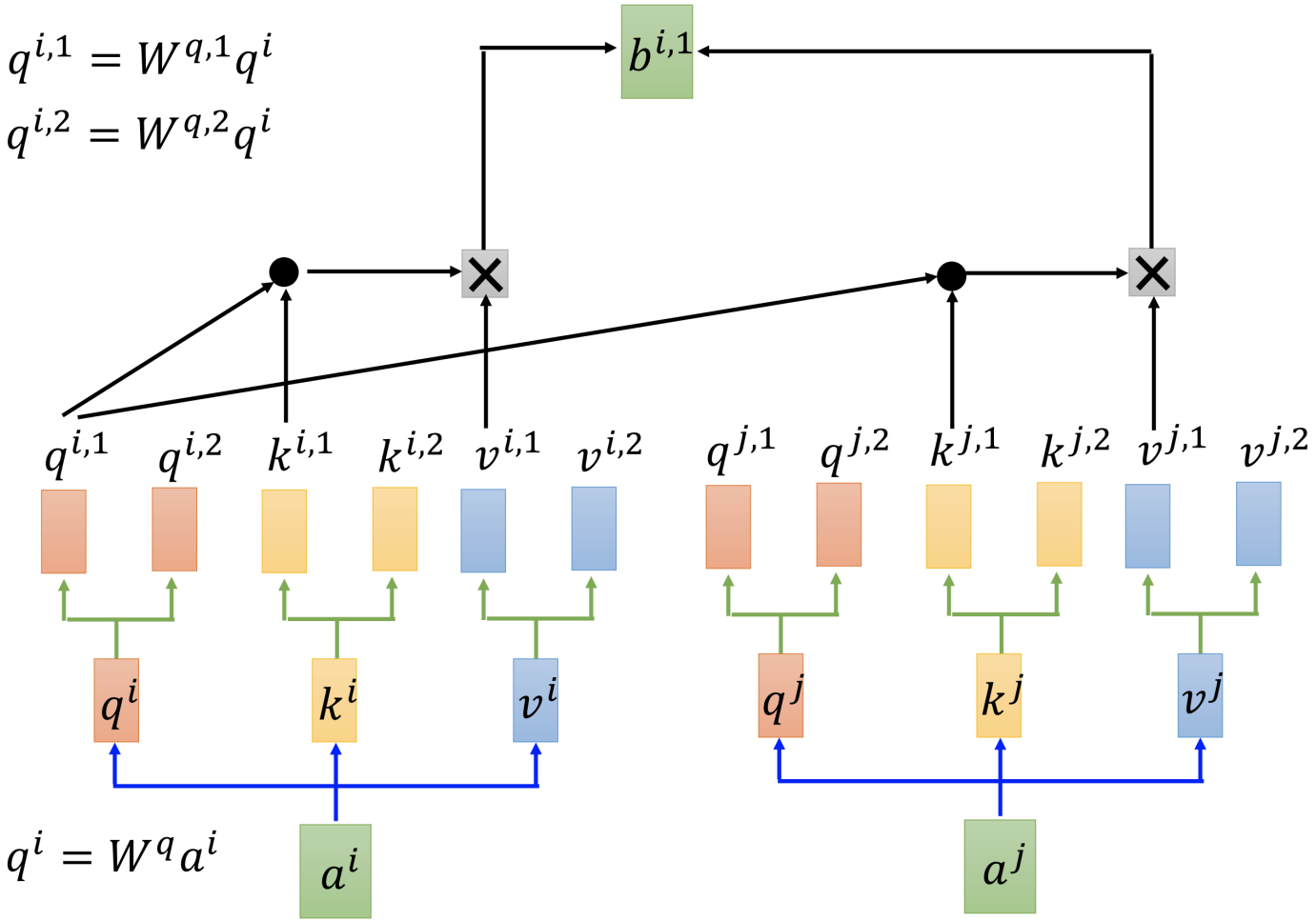
- $b^2 = \sum_i \hat{\alpha}_{2,i} \times v^i$: all the attentions of x^2 on other inputs and itself

Overview of Self-Attention



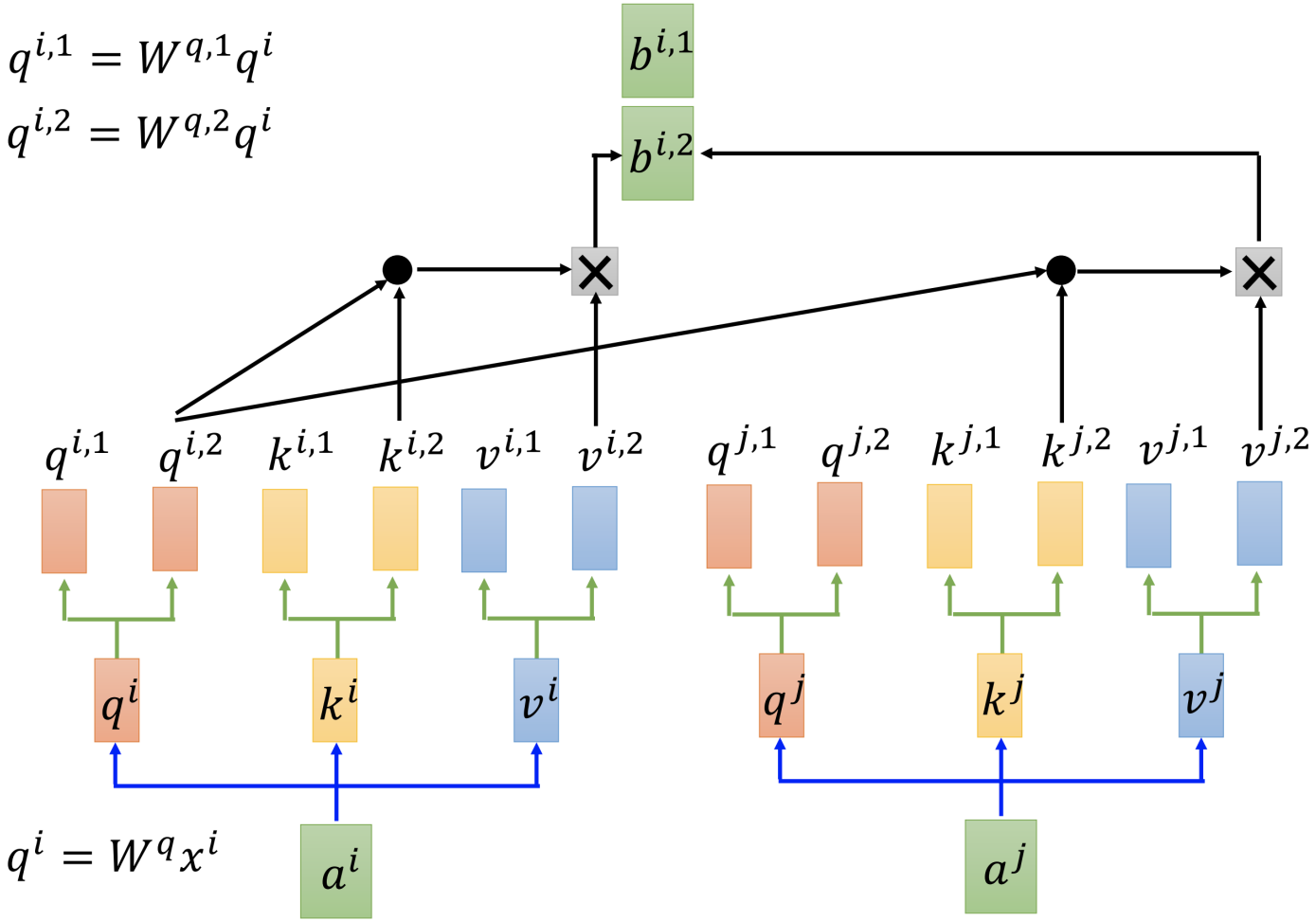
- b^1, b^2, b^3, b^4 can be computed in parallel (no data dependency)
- Idea: building a relationship between any 2 inputs (including itself)

Multi-Head Self-Attention



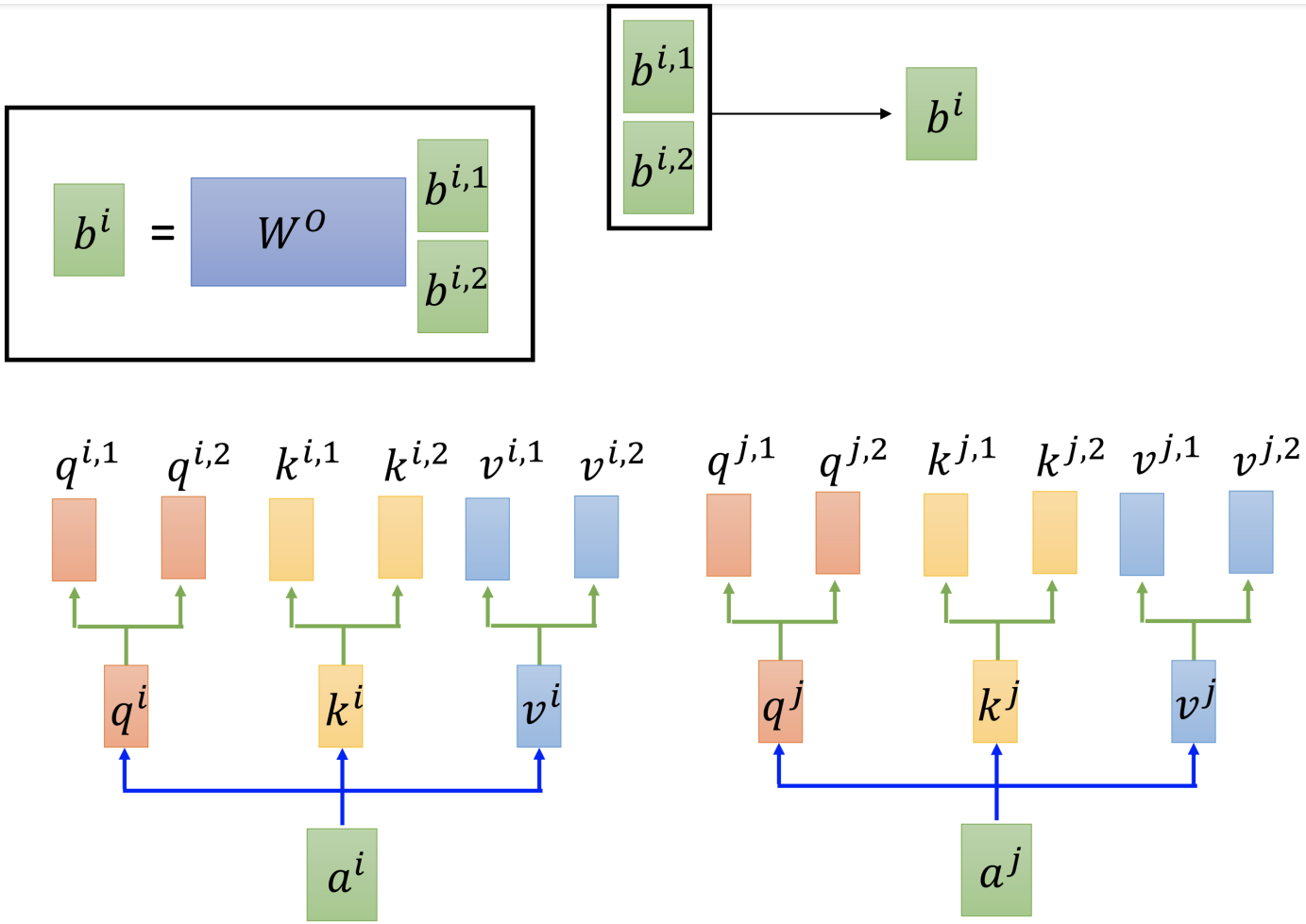
- Example of 2 heads
- They can only communicate within the same head group

Multi-Head Self-Attention



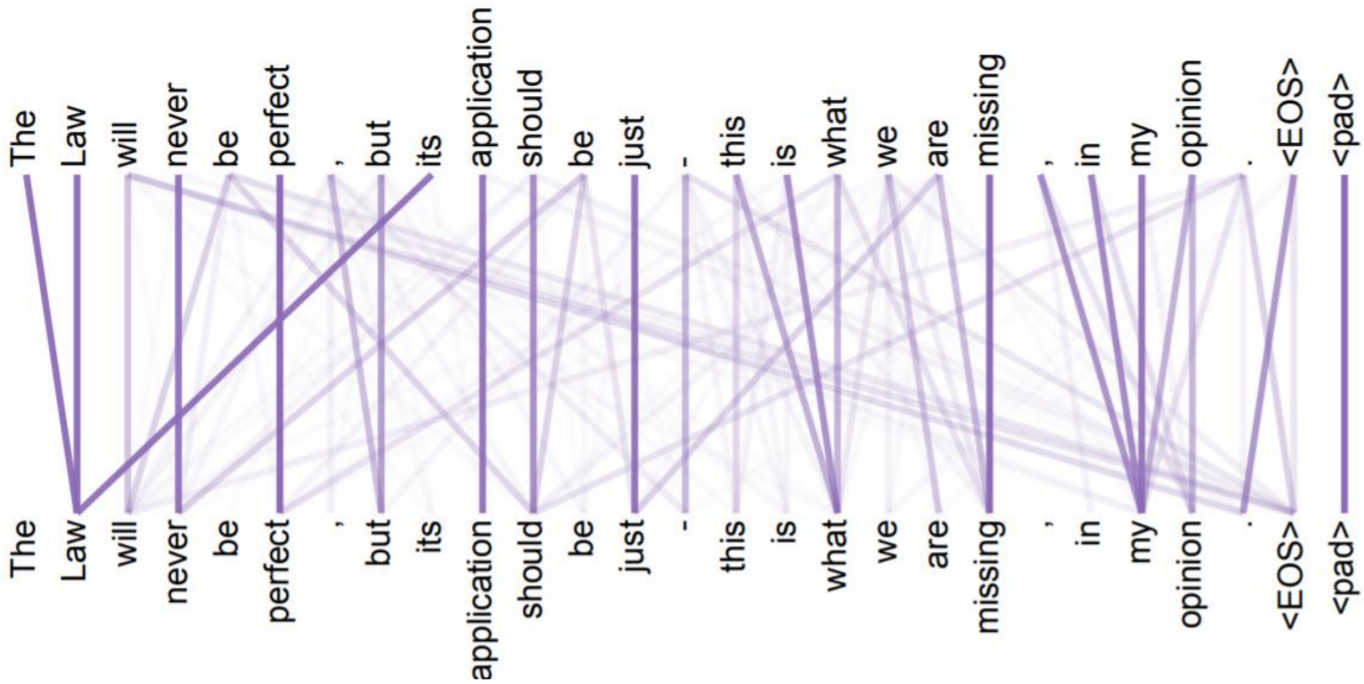
- Example of 2 heads
- They can only communicate within the same head group

Multi-Head Self-Attention

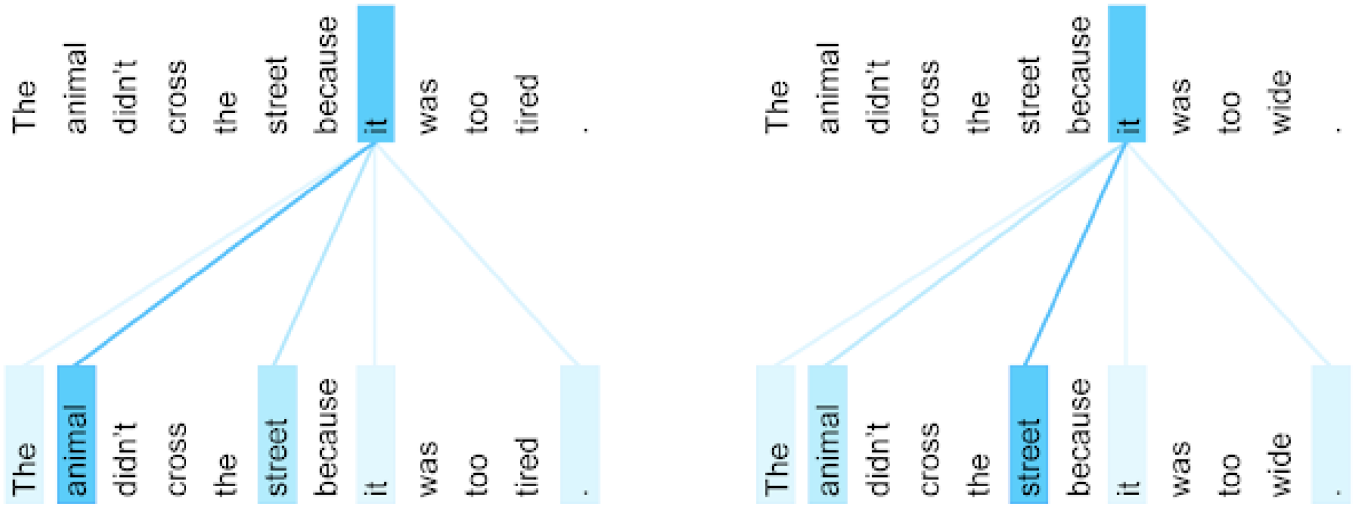


- We can merge the outputs the multiple heads into one output

Self-Attention Visualization

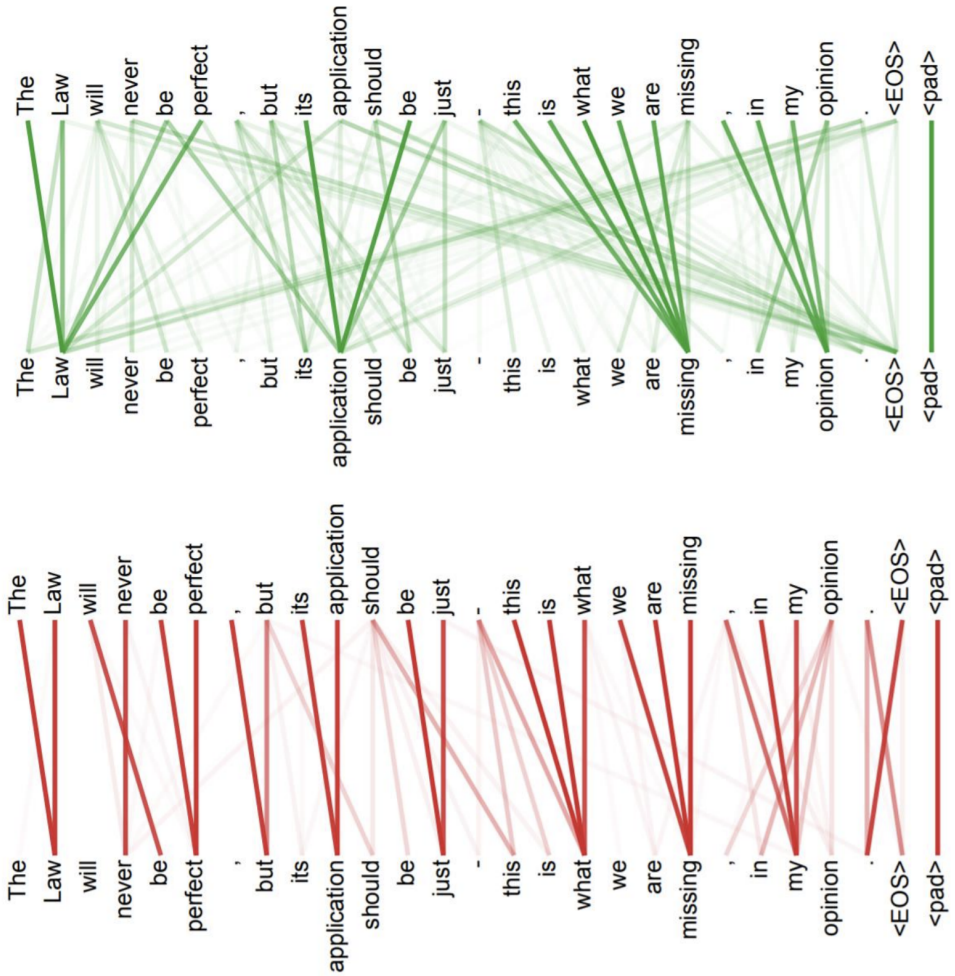


Self-Attention Visualization



- The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

Multi-Head Self-Attention Visualization

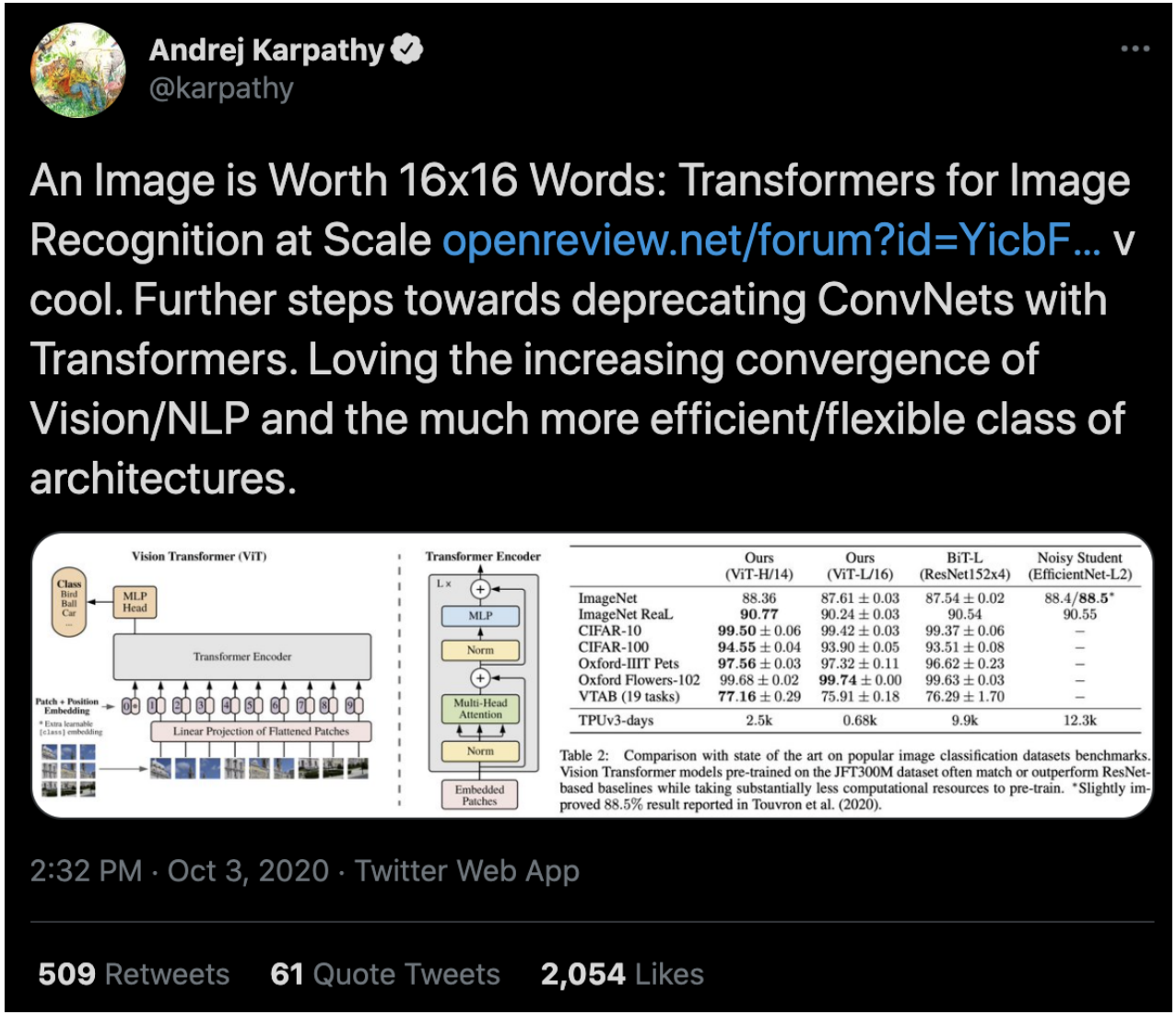


- Some heads focus on local information
- Some heads focus on global information

Outline

- Review on Transformers
- **Transformers for Computer Vision**
- ALBERT: an efficient BERT

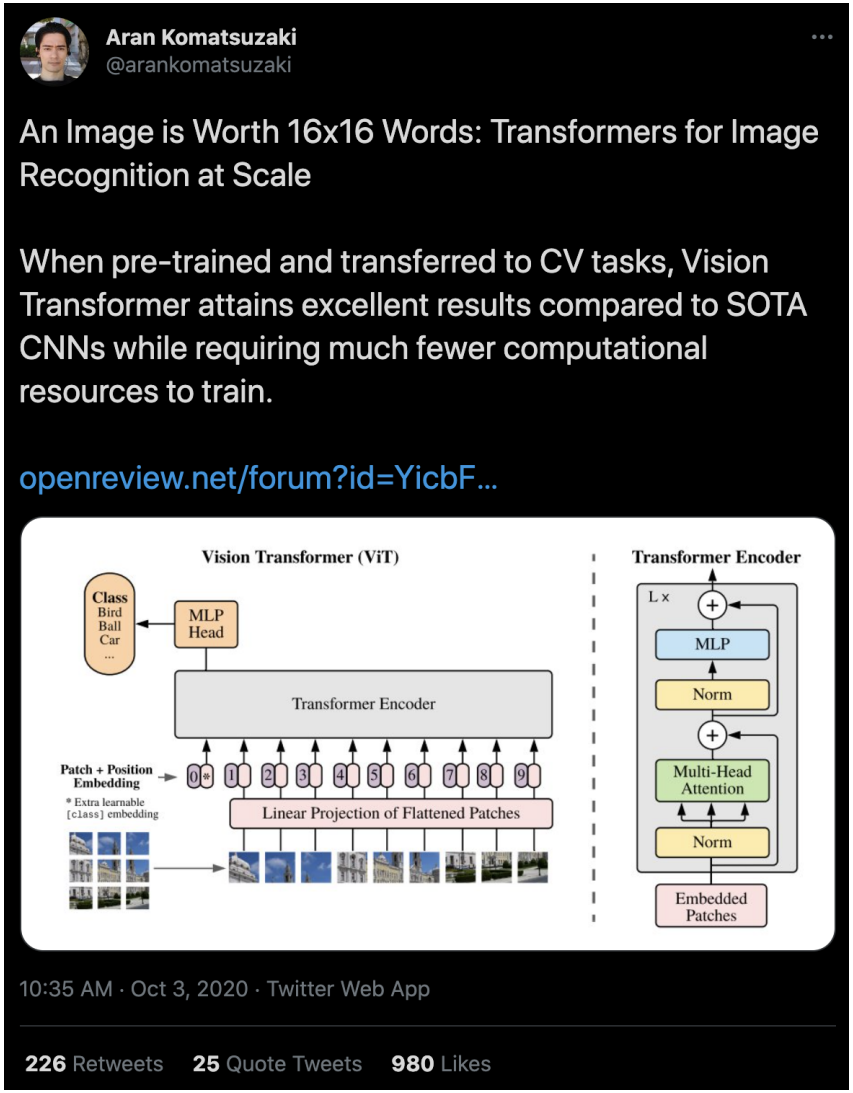
Transformers for Computer Vision



	Ours (ViT-H/14)	Ours (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.36	87.61 ± 0.03	87.54 ± 0.02	88.4/**88.5***
ImageNet ReaL	**90.77**	90.24 ± 0.03	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.63 ± 0.03	–
VTAB (19 tasks)	**77.16** ± 0.29	75.91 ± 0.18	76.29 ± 1.70	–
TPUv3-days	2.5k	0.68k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification datasets benchmarks. Vision Transformer models pre-trained on the JFT300M dataset often match or outperform ResNet-based baselines while taking substantially less computational resources to pre-train. *Slightly improved 88.5% result reported in Touvron et al. (2020).

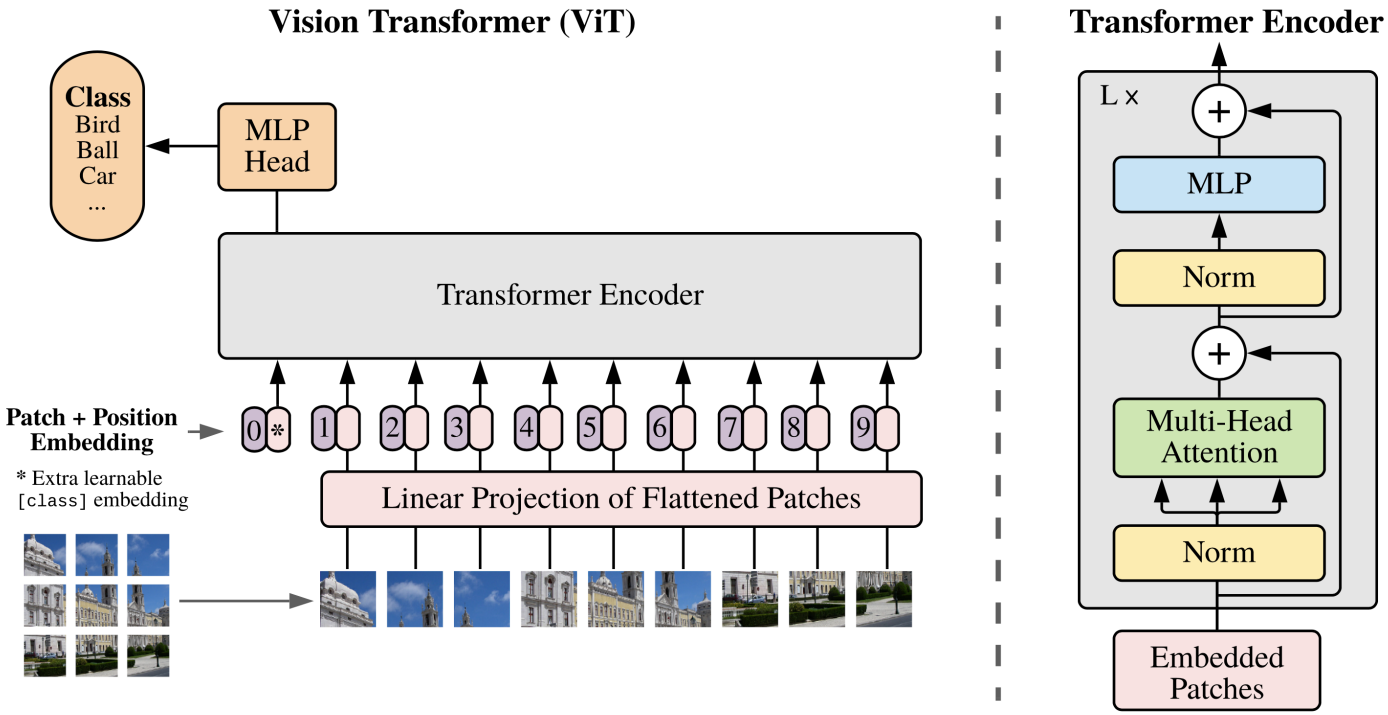
Transformers for Computer Vision



Early Attempts on Transformers for Computer Vision

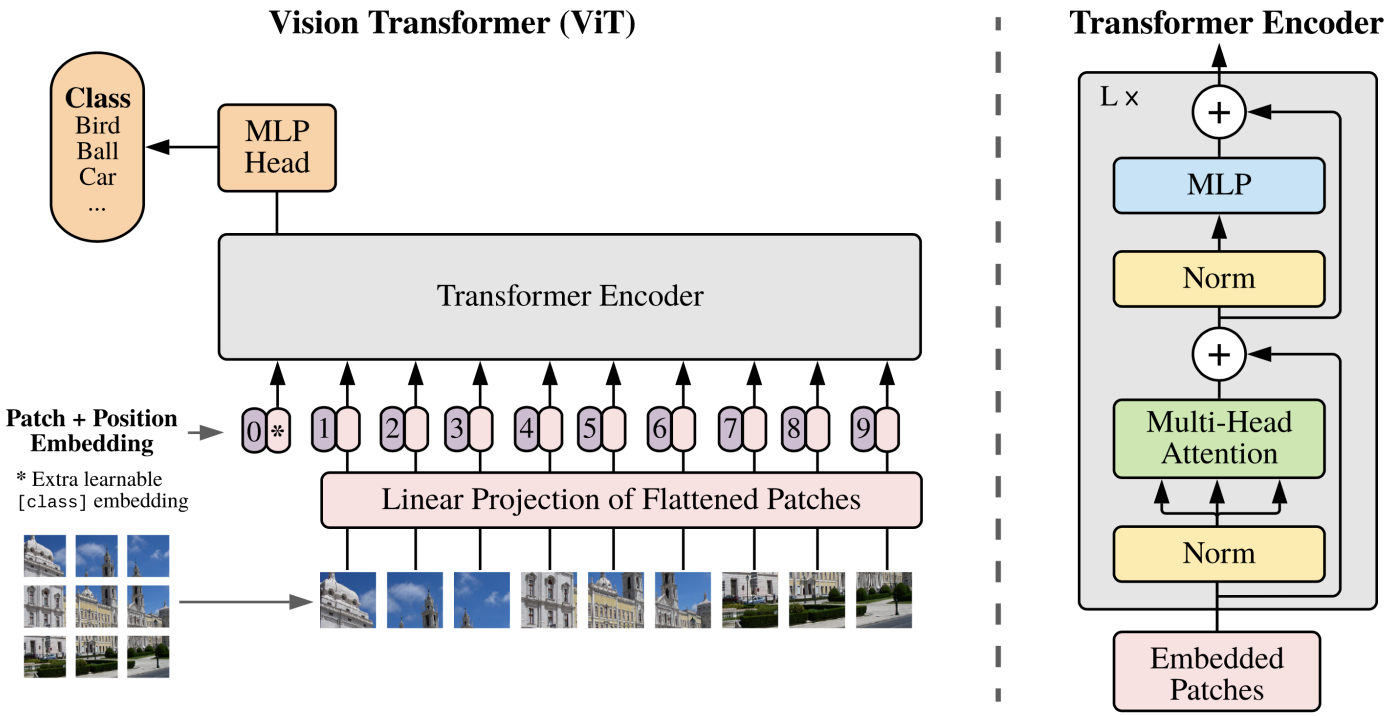
- Until recently, convolutional architectures remain dominant
  - Mahajan et al., 2018; Xie et al., 2020; Kolesnikov et al., 2020
- Combining CNN-like architectures with self-attention
  - Wang et al., 2018; Carion et al., 2020
- Replacing the convolutions entirely
  - Ramachandran et al., 2019; Wang et al., 2020
  - They did not scale effectively on modern hardware accelerators
    - Because of they used specialized attention patterns

Overview of ViT (Vision Transformer)



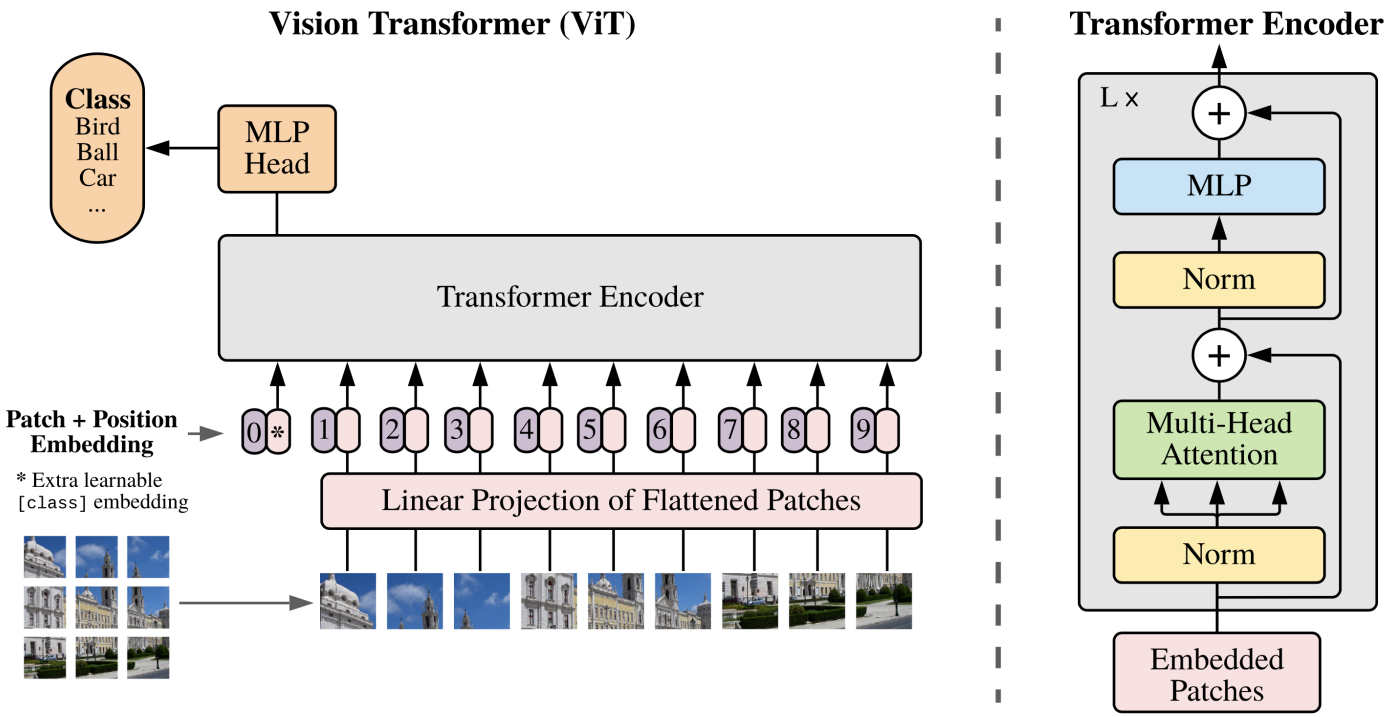
- Partition an image into fixed-size patches
- Linearly embed each of them
- Add position embeddings
- Feed the resulting sequence of vectors to a Transformer encoder

Overview of ViT (Vision Transformer)



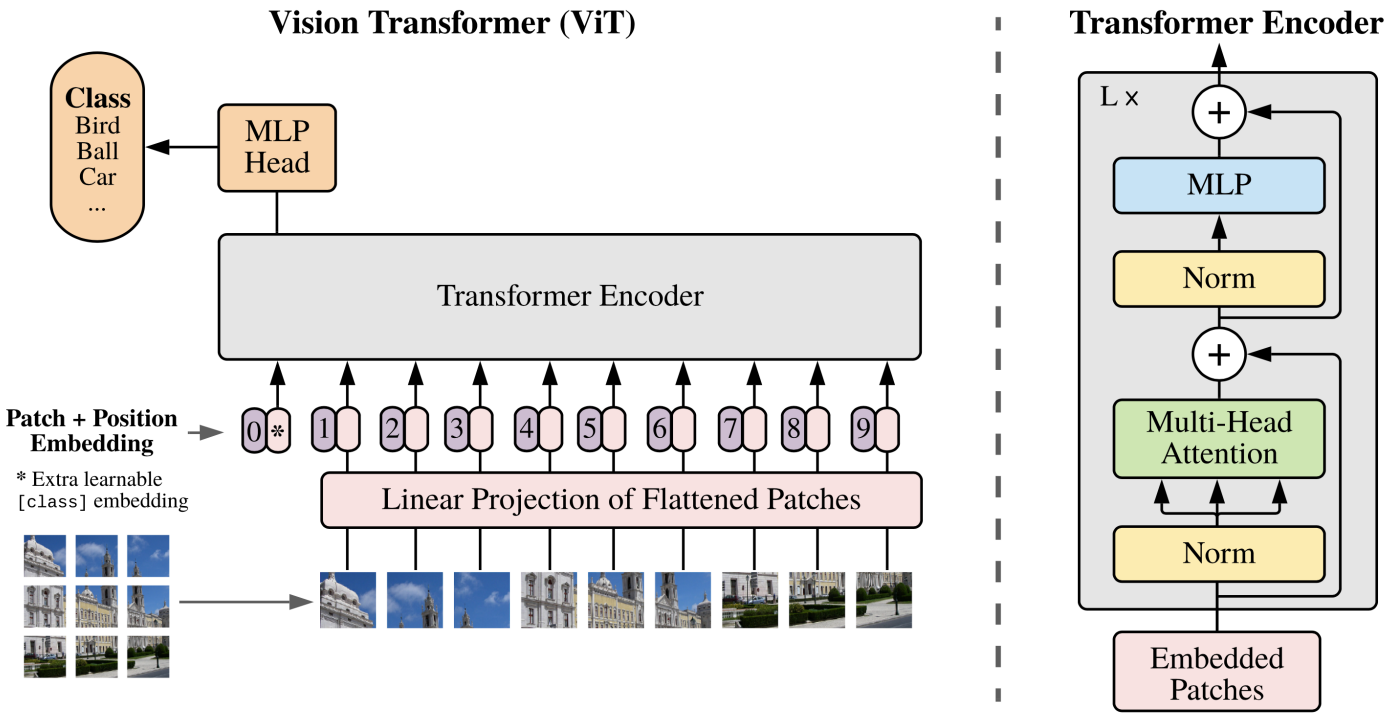
- To perform classification, they add an extra learnable "classification token" to the sequence
  - It is a standard approach

ViT (Vision Transformer) Input Processing



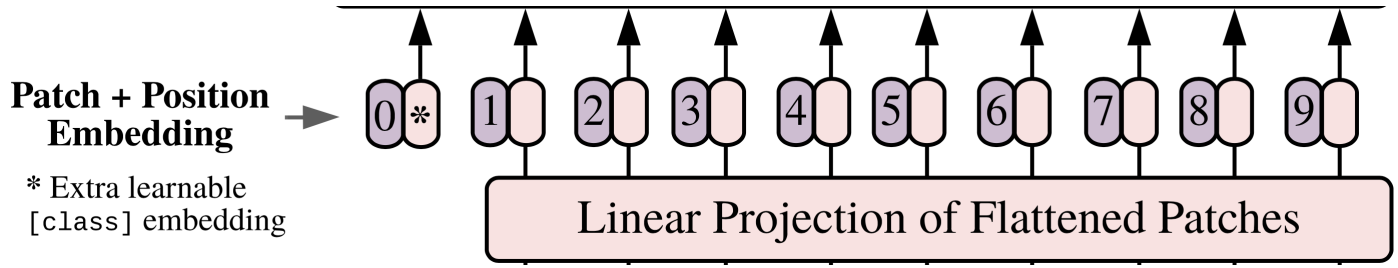
- The input of a standard NLP Transformer is a 1D sequence of token embeddings
  - To handle 2D images, we reshape the image $x \in R^{H \times W \times C}$ into a sequence of flattened 2D patches $x_p \in R^{N \times (P^2 C)}$
  - (H, W): resolution of the original image & C: number of channels
  - (P, P): resolution of an image patch & $N = HW/P^2$: number of patches
    - N is the effective input sequence length for the Transformer

ViT (Vision Transformer) Input Processing



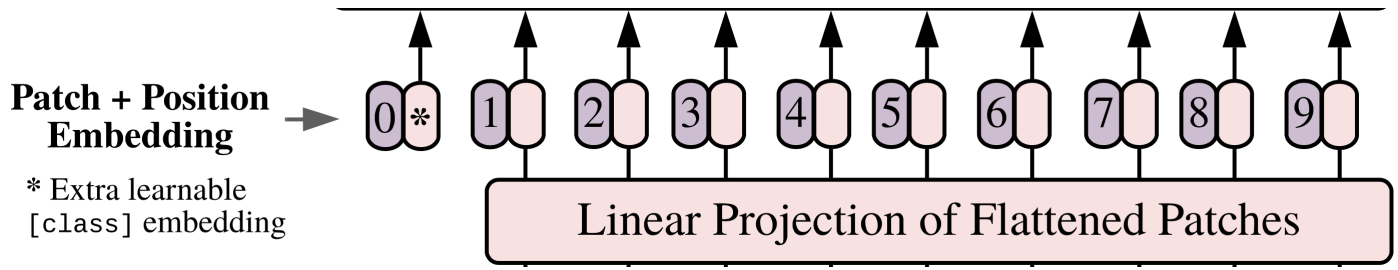
- ViT uses constant latent vector size D through all of its layers
  - They flatten the patches and map to D dimensions with a trainable linear projection
    - The output of this projection is called as the patch embeddings

Position Embedding



- Transformer does not have the position information
  - We need to use position embedding to add position information to the model
- The purple box is the position embedding
- The pink box is the patch embedding
  - The flattened patch after linear projection
- ViT put position embedding and patch embedding together to give the model postion information

Learnable Embedding



- The pink box with a star is not an image patch
  - It is a learnable embedding (denoted as x^*)
  - It is similar to the [class] token t^* in BERT
    - After encoder, [class] token t^* represents the whole sentence
  - Similarly, after encoder, x^* represents of the whole image
- Why BERT or ViT has this additional token t^* or patch x^*?
  - If we pick an existing image patch (x') or token (t') to represent the whole image or whole sentence, it will likely only represent x' or t'
  - t^* or x^* has nothing to do with a specific token or patch, can represent the whole sentence or image

Inductive Bias

- Inductive Bias: the assumption on a problem by an algorithm
  - CNN assumes the image has the property of "locality", so CNN puts neighboring features together (i.e. the idea of CNN kernel)
  - ViT has a weaker Inductive Bias than CNN
    - Different patches are equal, no matter they are far away or close
- In ViT, only MLP layer is local, while self-attention layer is global
  - ViT can minimize the usage of 2D neighborhood structure
    - In the beginning of the model: cutting the image into patches
    - At fine-tuning time: adjusting the position embeddings for images of different resolution
  - The position embeddings at initialization time carry no information about the 2D positions of the patches
    - It is just a 1D sequence

Hybrid Architecture

- As an alternative to raw image patches, the input sequence can be formed from feature maps of a CNN
- In this hybrid model, the patch embedding projection E is applied to patches extracted from a CNN feature map
- The classification input embedding and position embeddings are added using the same way as a regular ViT

Fine-Tuning and Higher-Resolution

- Typically, we pre-train ViT on large datasets, and fine-tune to (smaller) downstream tasks
  - For this, we remove the pre-trained prediction head and attach a zero-initialized $D \times K$ feedforward layer
    - K is the number of downstream classes
- It is beneficial to fine-tune at higher resolution than pre-training
  - Using a lower resolution in pre-training to reduce the training cost
- When feeding images of higher resolution, we keep the patch size the same, which results in a longer effective sequence length
  - The Vision Transformer can handle arbitrary sequence lengths
  - But the pre-trained position embeddings may no longer work
    - How can we solve this problem?
    - 2D interpolation of the pre-trained position embeddings[1]

[1] The resolution adjustment and patch extraction are the only points at which an inductive bias about the 2D structure of images is manually injected into the Vision Transformer

Dataflow of a ViT encoder layer

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

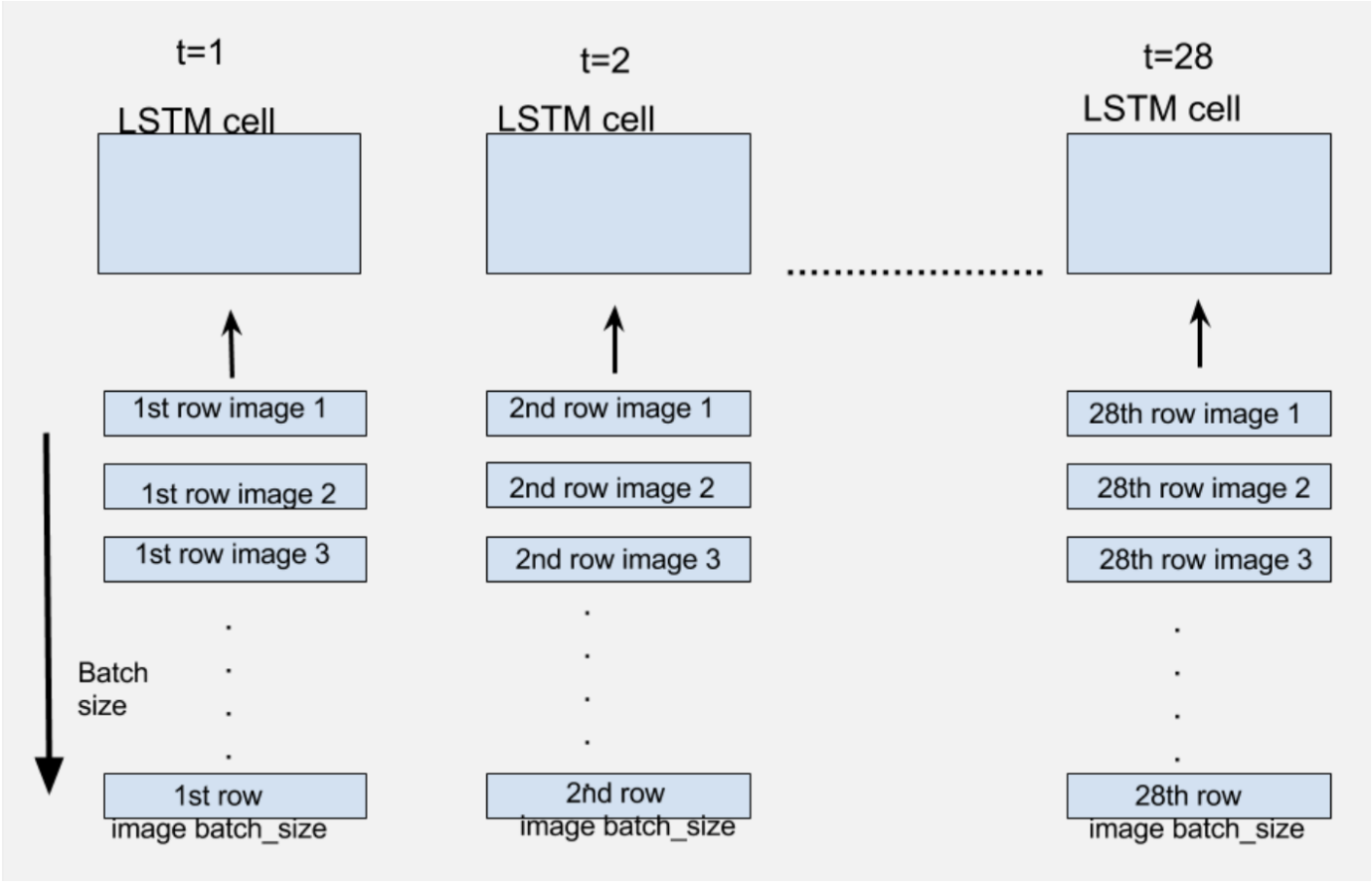
$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

- For l-th layer of the encoder
  - z_{l-1} is the input
  - z_l is the output
- In each layer, we have two main parts
  - MSA (Multi-head Self-Attention)
  - MLP (Multi-Layer Perceptron)

A similar idea before ViT



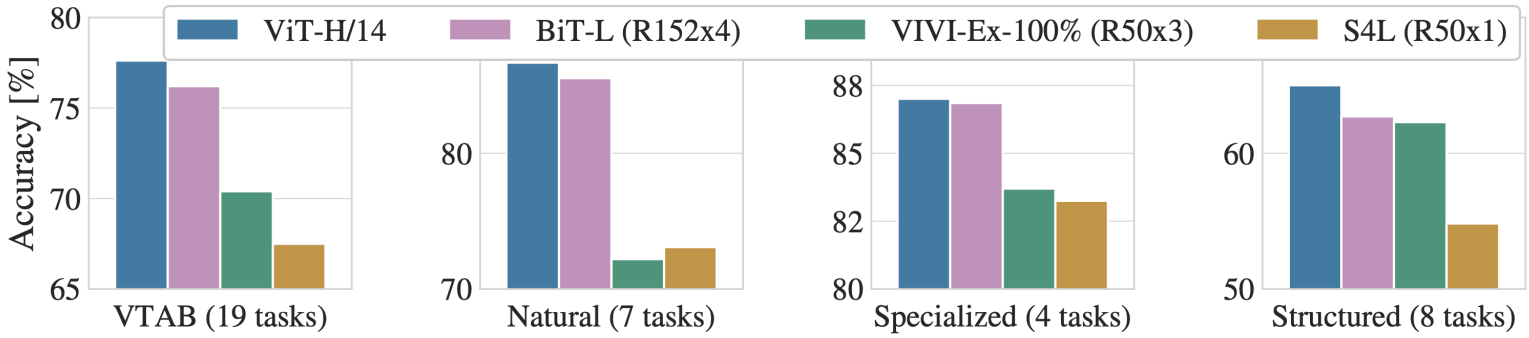
- LSTM for MNIST (image classification)
- https://jasdeep06.github.io/posts/Understanding-LSTM-in-Tensorflow-MNIST/

Different versions of ViT models

Model	Layers	Hidden size D	MLP size	Heads	Parameters
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

- ResNet-50: roughly 24M parameters

VTAB results



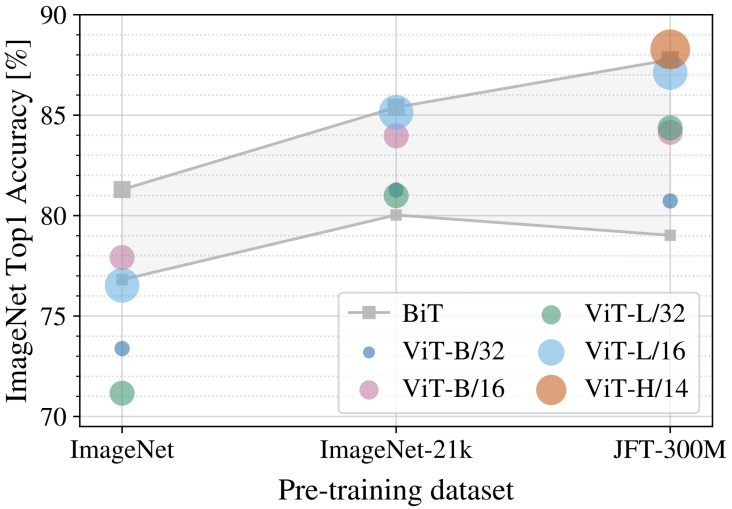
- VTAB (Visual Task Adaptation Benchmark) is an evaluation protocol designed to measure progress towards general and useful visual representations
- VTAB consists of a suite of evaluation vision tasks that a learning algorithm must solve

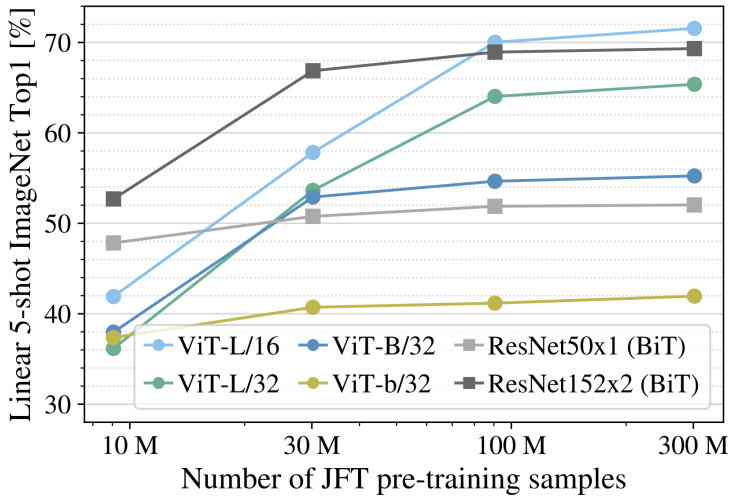
Compared to state-of-the-art

	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** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

- Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train.
- ViT pre-trained on the smaller public ImageNet-21k dataset performs well too.

Larger Models and Datasets



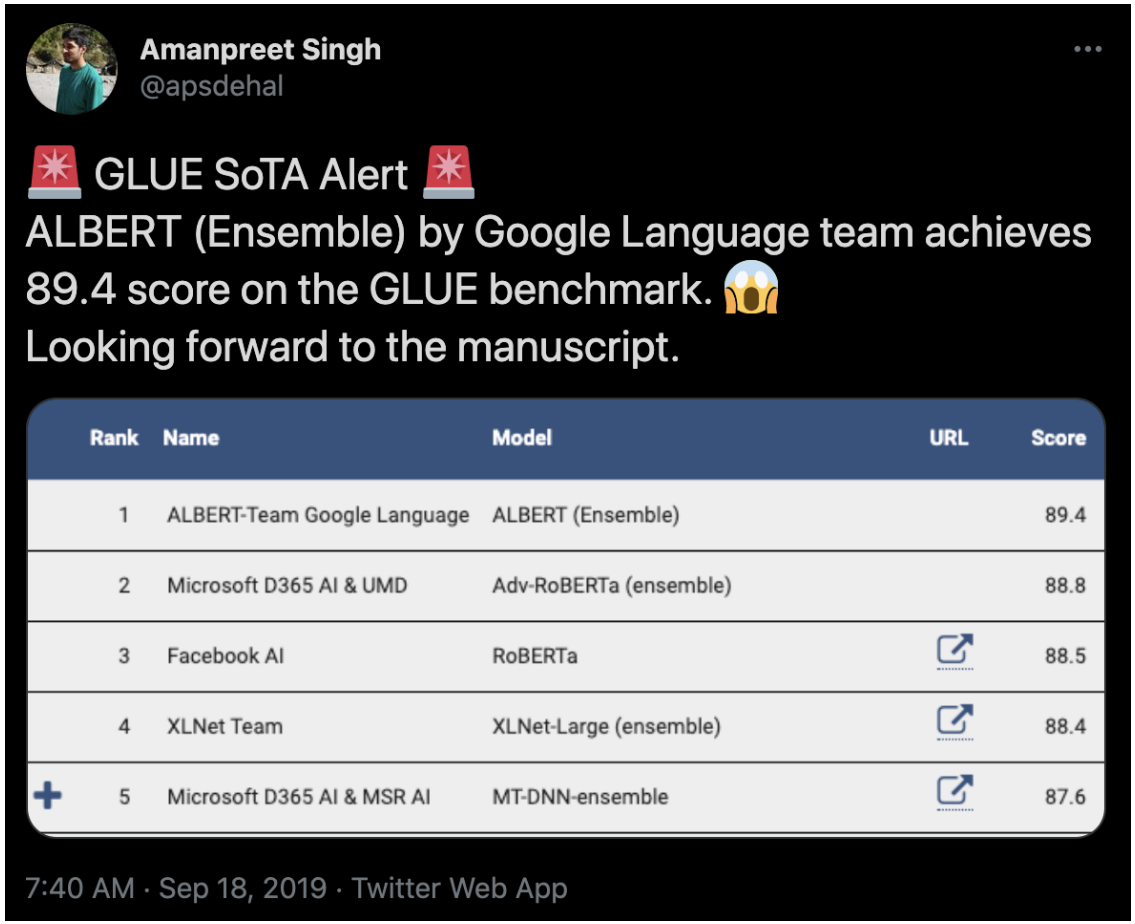


- Figure 1: while ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they become better when pre-trained on larger datasets.
- Figure 2: ResNets perform better with smaller pre-training datasets but plateau sooner than ViT.

Outline

- Review on Transformers
- Transformers for Computer Vision
- **ALBERT: an efficient BERT**

ALBERT (A Lite BERT)



Rank	Name	Model	URL	Score
1	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4
2	Microsoft D365 AI & UMD	Adv-RoBERTa (ensemble)		88.8
3	Facebook AI	RoBERTa		88.5
4	XLNet Team	XLNet-Large (ensemble)		88.4
5	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6

- ALBERT: A Lite BERT for Self-supervised Learning of Language Representations
- The most cited paper of ICLR 2020

Does a better NLP model means a larger model?

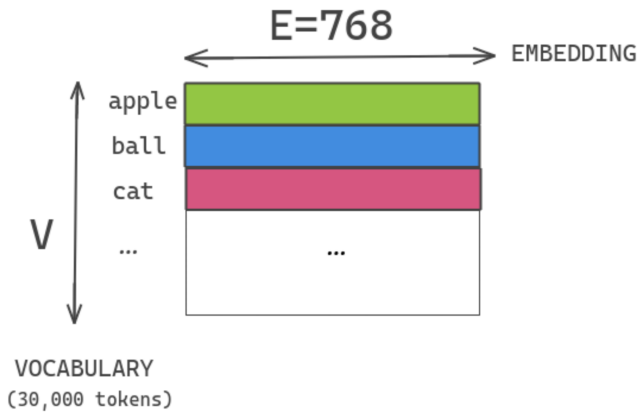
- The memory is limited on available hardware
  - Current state-of-the-art models often have hundreds of millions or even billions of parameters
  - It's easy to hit these limitations as we scale our models
- Training speed can also be significantly hurt in distributed training.
  - Communication overhead is proportional to the number of parameters in a model (data parallelism)
    - Regular distributed training: averaging the gradients.
- System-level solutions: model parallelism & memory management
  - Can't reduce communication overhead
- New solution: ALBERT
  - Significantly reduce the number of parameters in BERT

Model Architecture of ALBERT

- The backbone of the ALBERT architecture is similar to BERT
  - Transformer encoder with GELU nonlinear activation function
- Settings based on BERT
  - H: the hidden size
  - E: the vocabulary embedding size
  - L: the number of encoder layers
  - $4H$: the feed-forward/filter size
  - $H/64$: the number of attention heads

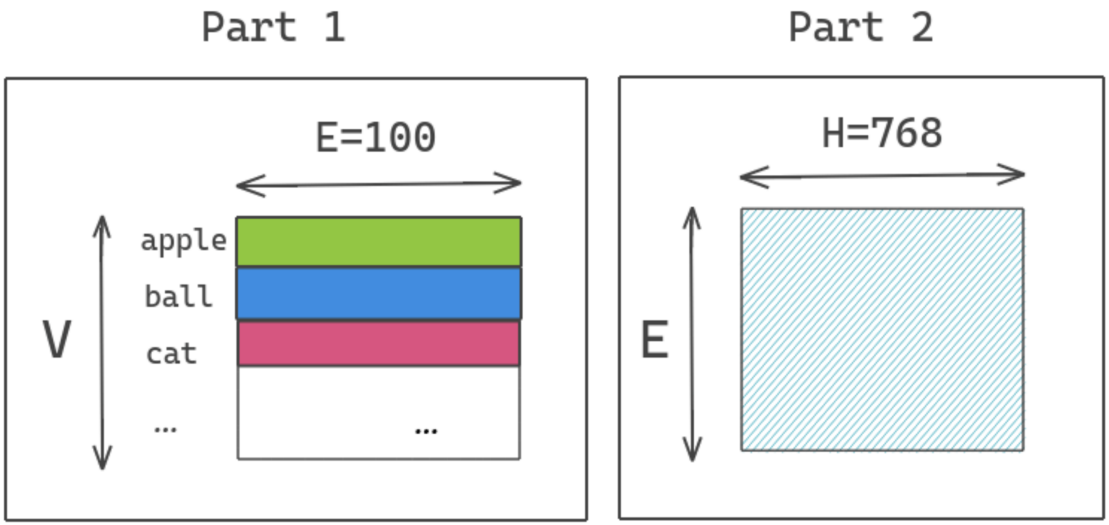
Method 1: factorized embedding parameterization

- In BERT, WordPiece embedding size E is tied with hidden layer size H, i.e., $E \equiv H$
- This decision is suboptimal for modeling and practical reasons
  - Goal of WordPiece embeddings: learn context-independent representations
  - Goal of hidden-layer embeddings: learn context-dependent representations
  - Untying the WordPiece embedding size E from the hidden layer size H allows us to make a more efficient usage of the total model parameters as informed by modeling needs ($H \gg E$)
  - In NLP, the vocabulary size V is typically large.
  - If $E \equiv H$, increasing H increases the embedding matrix size ($V \times E$)
    - This can easily result in a model with billions of parameters



Method 1: factorized embedding parameterization

- ALBERT factorized the embedding matrix into 2 smaller matrices
  - Decomposing a V-by-H matrix into V-by-E and E-by-H matrices
  - It reduced the embedding matrix from $\Theta(V \times H)$ to $\Theta(V \times E + E \times H)$
  - This parameter reduction is significant when $H \gg E$

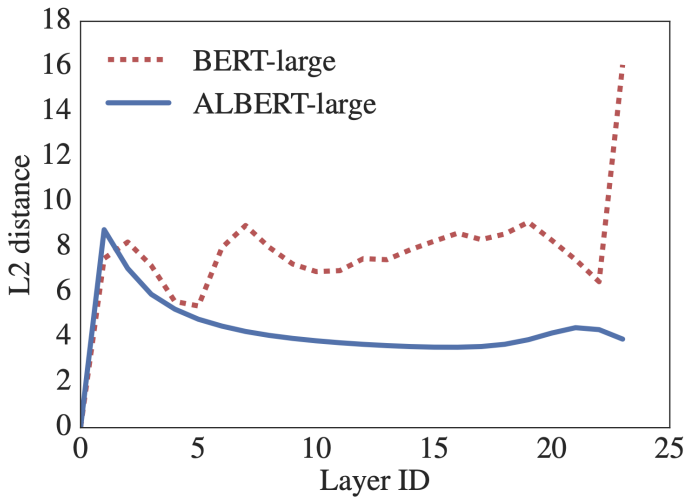


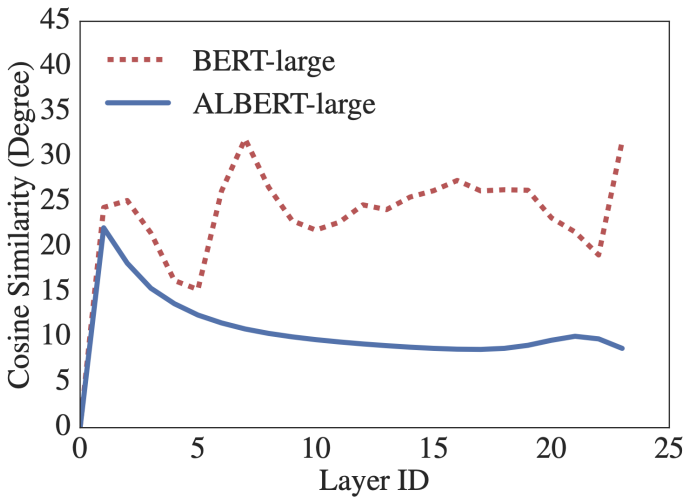
Method 2: cross-layer parameter sharing

- There are multiple ways to share parameters
  - Only sharing feed-forward network (FFN) parameters
  - Only sharing attention parameters
- The default decision for ALBERT: share all parameters across layers

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Method 2: cross-layer parameter sharing



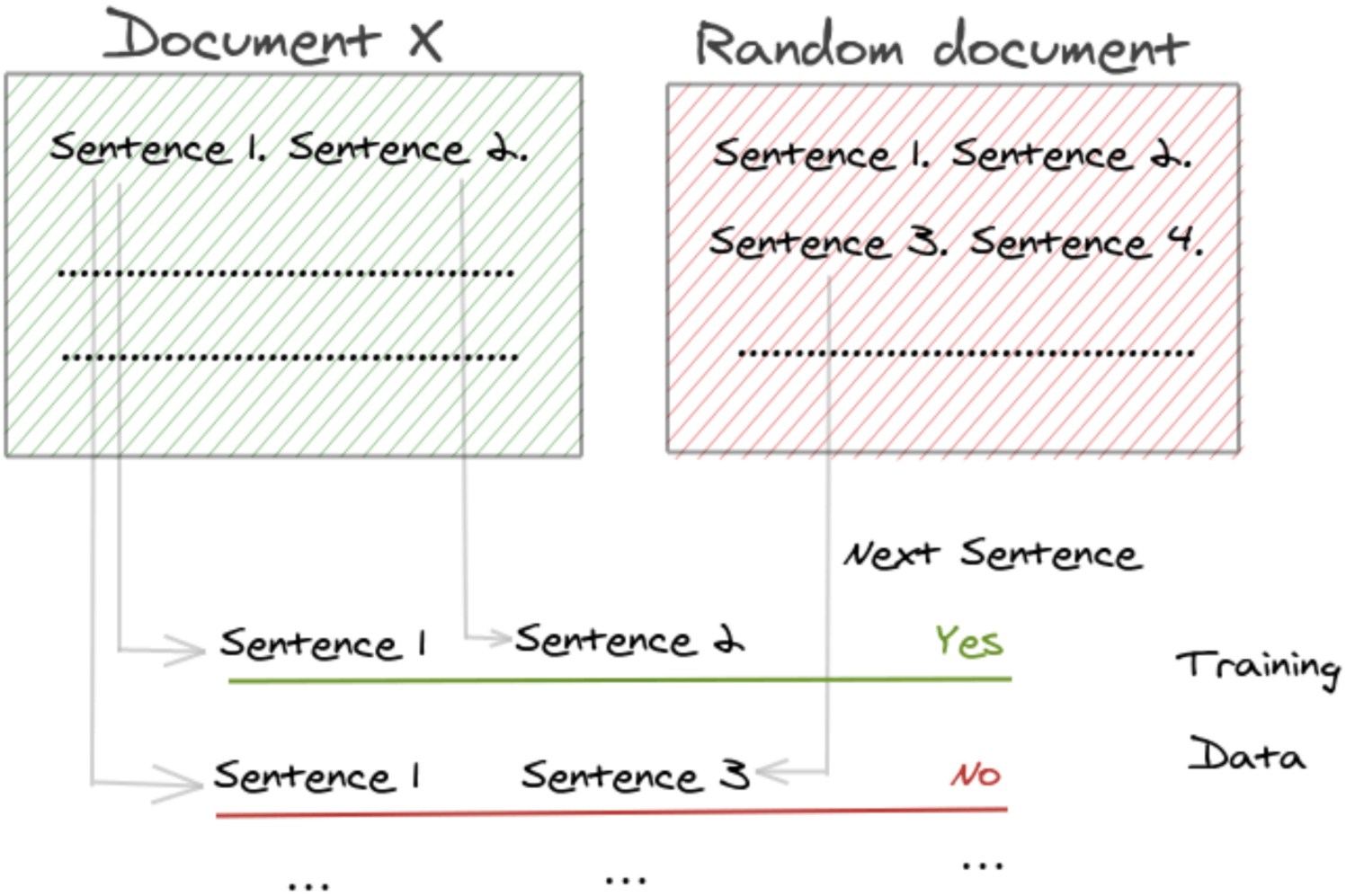


- The transitions from layer to layer are much smoother for ALBERT than for BERT.
  - The weight-sharing has an effect on stabilizing network parameters.
- Although there is a drop for both metrics compared to BERT, they do not converge to 0 even after 24 layers.
  - This shows that the solution space for ALBERT is very different from the one found by DQE (Bai et al., 2019).
  - DQE reaches a balance point where the input and output embedding of a certain layer stay the same.

Method 3: inter-sentence coherence loss

- BERT's loss function has two parts:
  - Masked Language Modeling (MLM) loss
  - Next-Sentence Prediction (NSP) loss
- NSP is a binary classification for predicting whether two segments appear consecutively in the original text
  - Positive examples: consecutive segments from the same document
  - Negative examples: pairing segments from different documents
  - Positive and Negative examples are sampled with equal probability
  - It was designed to improve performance on downstream tasks
    - These tasks (e.g. natural language inference) require reasoning about the relationship between sentence pairs
- Weakness of NSP (Yang et al., 2019)
  - NSP's impact is unreliable, so XLNET decided to remove it
  - XLNET can get a better performance after removing it

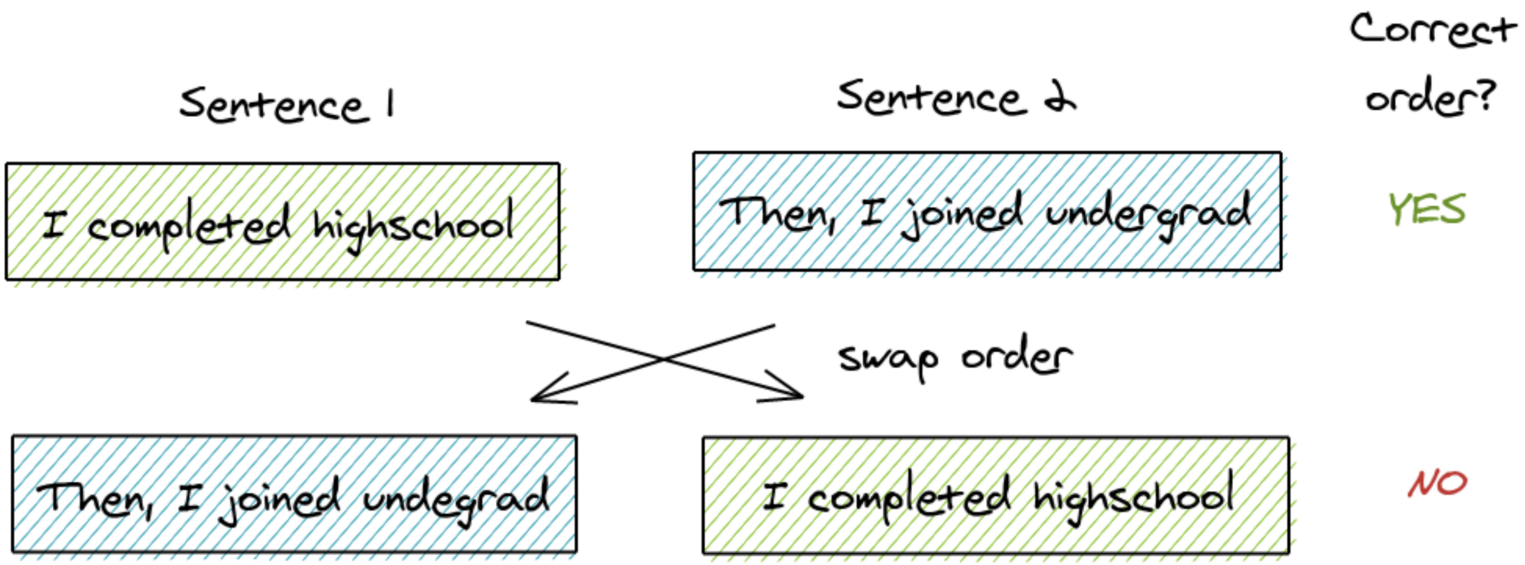
Next-Sentence Prediction (NSP)



Method 3: inter-sentence coherence loss

- The main reason behind NSP's ineffectiveness
  - Compared to MLM, it lacks of difficulty as a task
  - NSP combines topic prediction and coherence prediction
    - Topic prediction is easier than coherence prediction
    - Topic prediction also overlaps more with MLM
- ALBERT does inter-sentence modeling, but focuses on coherence
  - ALBERT used a sentence-order prediction (SOP) loss
  - It avoids topic prediction and focuses on inter-sentence coherence
  - Positive examples: consecutive segments from the same document
  - Negative examples: consecutive segments but with swapped order
- NSP cannot solve the SOP task at all
  - NSP learns easier topic-prediction signals, and performs at random baseline level on the SOP task
- SOP can solve the NSP task to a reasonable degree
  - It is probably based on studying misaligned coherence signals

Sentence-Order Prediction (SOP)



BERT vs ALBERT

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

- Dev set results for models pretrained over BOOKCORPUS and Wikipedia for 125k steps
- The Avg column is computed by averaging the scores of the downstream tasks to its left

The effect of vocabulary embedding size

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

The effect of cross-layer parameter-sharing

	Model	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

The effect of controlling for training time

Models	Steps	Time	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
BERT-large	400k	34h	93.5/87.4	86.9/84.3	87.8	94.6	77.3	87.2
ALBERT-xxlarge	125k	32h	**94.0/88.1**	**88.3/85.3**	87.8	**95.4**	**82.5**	**88.7**

The effect of removing dropout

	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
With dropout	94.7/89.2	89.6/86.9	90.0	96.3	85.7	90.4
Without dropout	**94.8/89.5**	**89.9/87.2**	**90.4**	**96.5**	**86.1**	**90.7**

- Results by ALBERT-xxlarge
- ALBERT is the first to show that dropout can hurt performance in large Transformer-based models

State-of-the-art results on the GLUE benchmark

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

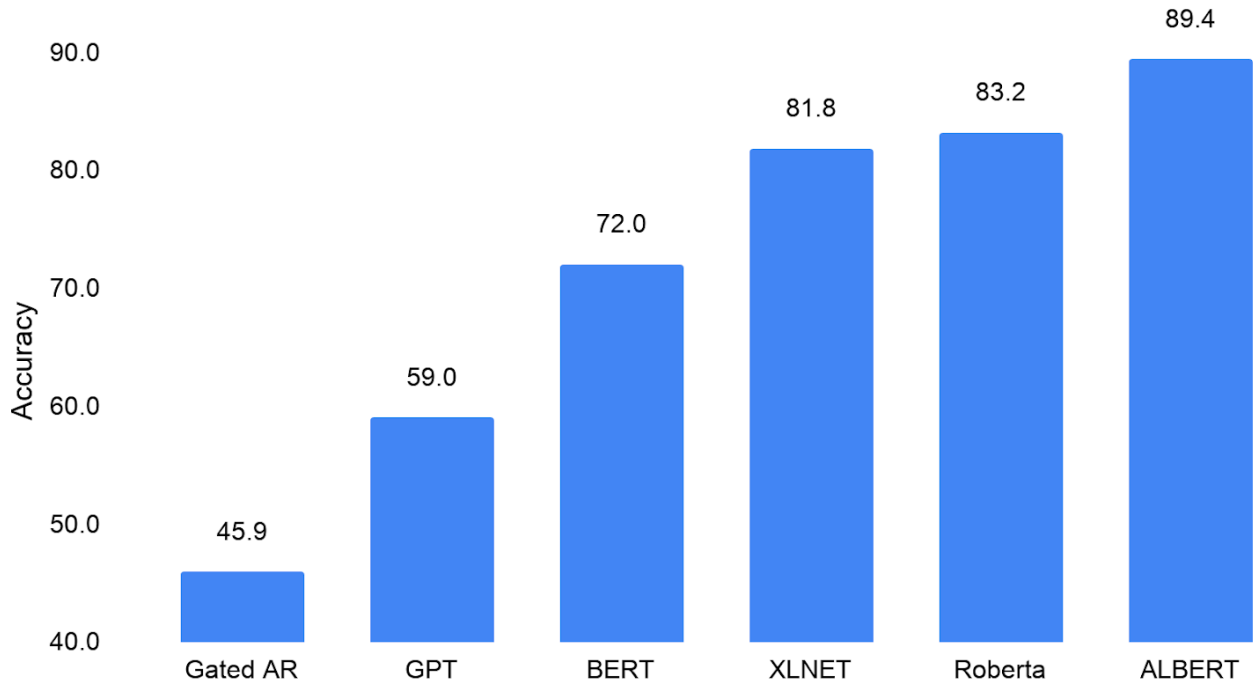
- ALBERT at 1M steps and at 1.5M steps

State-of-the-art results on SQuAD & RACE benchmarks

Models	SQuAD1.1 dev	SQuAD2.0 dev	SQuAD2.0 test	RACE test (Middle/High)
Single model (from leaderboard as of Sept. 23, 2019)				
BERT-large	90.9/84.1	81.8/79.0	89.1/86.3	72.0 (76.6/70.1)
XLNet	94.5/89.0	88.8/86.1	89.1/86.3	81.8 (85.5/80.2)
RoBERTa	94.6/88.9	89.4/86.5	89.8/86.8	83.2 (86.5/81.3)
UPM	-	-	89.9/87.2	-
XLNet + SG-Net Verifier++	-	-	90.1/87.2	-
ALBERT (1M)	94.8/89.2	89.9/87.2	-	86.0 (88.2/85.1)
ALBERT (1.5M)	**94.8/89.3**	**90.2/87.4**	**90.9/88.1**	**86.5 (89.0/85.5)**
Ensembles (from leaderboard as of Sept. 23, 2019)				
BERT-large	92.2/86.2	-	-	-
XLNet + SG-Net Verifier	-	-	90.7/88.2	-
UPM	-	-	90.7/88.2	
XLNet + DAAF + Verifier	-	-	90.9/88.6	-
DCMN+	-	-	-	84.1 (88.5/82.3)
ALBERT	**95.5/90.1**	**91.4/88.9**	**92.2/89.7**	**89.4 (91.2/88.6)**

- ALBERT at 1M steps and at 1.5M steps

Highlight of ALBERT



- Machine performance on the RACE challenge (SAT-like reading comprehension).
  - A random-guess baseline score is 25.0.
  - The maximum possible score is 95.0.
- ALBERT established new state-of-the-art results on the well-known GLUE, SQuAD, and RACE benchmarks for natural language understanding.

Summary of ViT

- Applying a standard Transformer directly to images
  - Inspired by the Transformer scaling successes in NLP
  - With the fewest possible modifications to NLP Transformers
  - They partition an image into patches and use the sequence of linear embeddings of these patches as an input to Transformers
    - Image patches are treated the same way as tokens (words) in NLP
  - They train models on image classification in supervised fashion
- When trained on mid-sized datasets (e.g. ImageNet)
  - Transformer has a lower accuracy than ResNet of comparable size
  - Transformer lacks some of the inductive biases inherent to CNNs
    - e.g. Translation equivariance[2] and locality
  - Transformer does not generalize well when trained on insufficient amounts of data
- When trained on larger datasets (e.g. 14M-300M images)
  - Large-scale training outperforms inductive bias
  - ViT gets great results when pre-trained at sufficient scale and transferred to tasks with fewer data points

[2] https://aboveintelligent.com/ml-cnn-translation-equivariance-and-invariance-da12e8ab7049

Summary of ALBERT

- Two parameter reduction techniques
  - **Factorized Embedding Layer**
    - By decomposing the large vocabulary embedding matrix into two small matrices, ALBERT separated the size of the hidden layers from the size of vocabulary embedding.
    - This separation allows it to grow the hidden size without significantly increasing the parameter size of the vocabulary embeddings.
  - **Cross-Layer Parameter Sharing**
    - The parameter size doesn't grow with the depth of the network.
  - Both techniques significantly reduce the number of parameters for BERT without seriously hurting performance, thus improving parameter-efficiency.
  - They also act as a form of regularization that stabilizes the training and helps with generalization.
- A self-supervised loss for sentence-order prediction (SOP)
  - SOP primary focuses on inter-sentence coherence.
  - It is designed to address the ineffectiveness of the next sentence prediction (NSP) loss proposed in the original BERT.

References

- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000-6010. 2017.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171-4186. 2019.
- Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).
- Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations." In International Conference on Learning Representations. 2020.
- Dr. Hung-yi Lee's tutorial on Transformers
  - https://speech.ee.ntu.edu.tw/~tlkagk/talk.html
- https://amitness.com/2020/02/albert-visual-summary/