History of Transformers, 2017-…
The Shortest Version

Survey on Transformers” by Lin *et al*

Mike Erlihson, Salt Security

Agenda:

- Recap on Transformers
- Transformer Usages: Domains, Tasks, Architecture and, Configurations
- Transformers Shortcomings: What can be improved in Transformers?
- Attention Mechanism Improvements: Research Directions
- Position Encoding Scheme Improvements: Research Directions
- Transformers without Attention: Can it Work?
- Future of Transformers

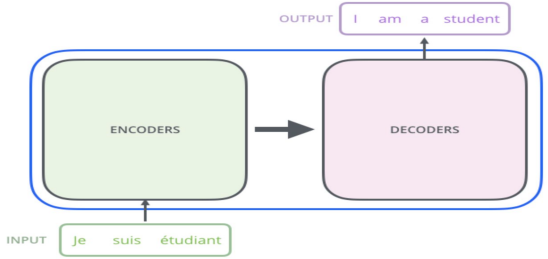
Transformer: A New Emperor of the NLP World

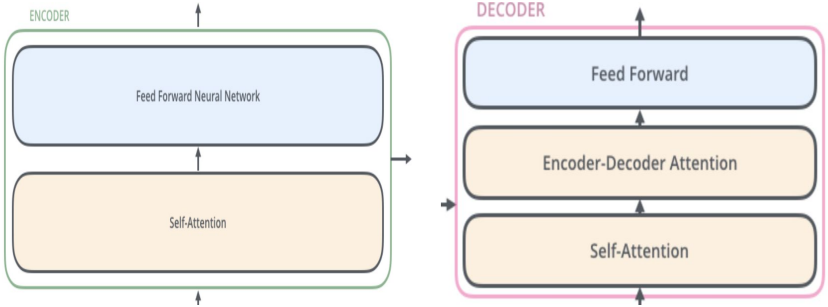
- NLP models until 2018 used different flavours of RNN for most types of tasks
- But then… "**Attention is all you need**" & "**BERT: ...**" papers were published and took the NLP world by **STORM!**
- They introduced a new network **Architecture,** called **Transformer,** relying on **attention mechanism**
- Since then **ALMOST ALL NLP** models are based on **Transformers/BERT** (RoBERTA, AlBERT, LinFormer, Reformer, PerFormer)

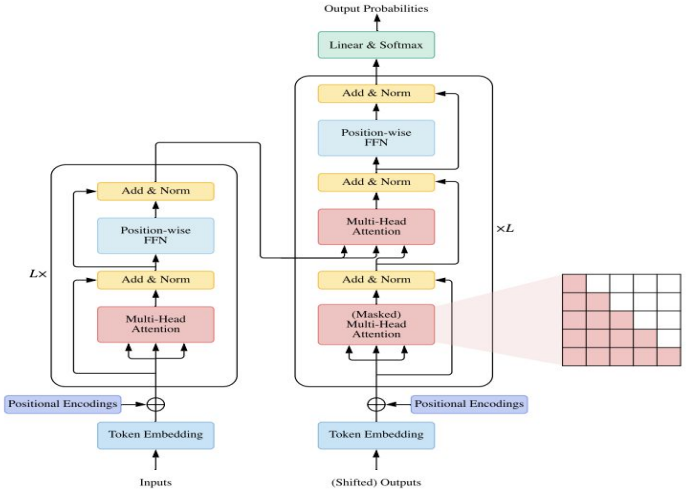
Transformers: High-Level Look

Text in Language A

Text in Language B







Illustrations from The Illustrated Transformer by Jay Alammar

But what is Self-Attention?

Attention Mechanism: An Essence

- **Attention** is a method of building sequence element representation based on its similarity (connection, relationship) with other sequence elements

- **Transformer** (vanilla) employs attention mechanism via converting an input element representation into **Query(Q), Key(K) & Value(V)** vectors

- Normalized **Attention Weights** are computed as a softmax of **K & Q** vectors dot product

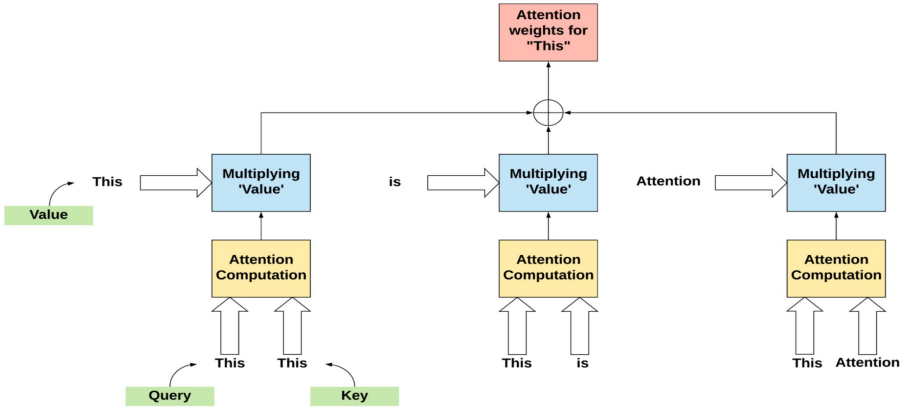
- **V-**vectors of all sequence elements are summed with its attention weights before to build **Contextualized Token Representation(Embedding)**

Attention with a Punch of Math

Weights = softmax((QX) * transpose(KX)) - attention weights

Attention Output = Weights * (VX), X is an input sequence representation

- Transformer can be seen as a **generalization** of a Feed-Forward Layer
- Self-attention can be seen as a FC Layer with **input-dependent** weights



Self-attention for
"This is attention" sentence

Pretrained Transformers is the Name of the Game

Key Observation: Transformer doesn't make **any assumption** about the data structure (as opposed to ConvNets & RNNs)

Pros: An universal architecture potentially capable of capturing dependencies of different ranges.

Cons: **Prone to overfitting** when the data is limited

What can be done? Pretrain on large datasets with different self-supervised objectives (e.g., predicting a masked word given its context).

- Learns powerful representations useful for downstream task (different domains)
- Finetune the model on downstream dataset, instead of training it from scratch

Transformers: Usage Modes

Encoder only: An encoder used as a backbone architecture for natural language understanding tasks (e.g. sentiment analysis, relation extraction, topic modeling etc)

Decoder only: A decoder is trained on language modeling tasks (e.g. text generation - GPT models)

Encoder-Decoder: Use both encoder-decoder as an overall architecture (e.g. for question answering, summarization - BART, T5)

Transformers: Variety of Domains & Tasks

NLP: Machine translation, language modeling, named entity recognition, question answering, summarization….

Computer Vision: Image classification, object detection, image generation, segmentation, video processing…

Audio Processing: Speech recognition, speech synthesis, speech enhancement, music generation…

Multimodal Applications: Visual question answering, visual commonsense reasoning, caption generation, speech-to-text translation text-to-image generation…

Biology/Chemistry: protein folding prediction, study of DNA & RNA sequences, drug discovery….


TRANSFORMERS
TRANSFORMERS EVERYWHERE
memegenerator.net

Main Transformers Shortcomings

Quadratic complexity in terms of an input length: especially **PAINFUL** for image and video applications

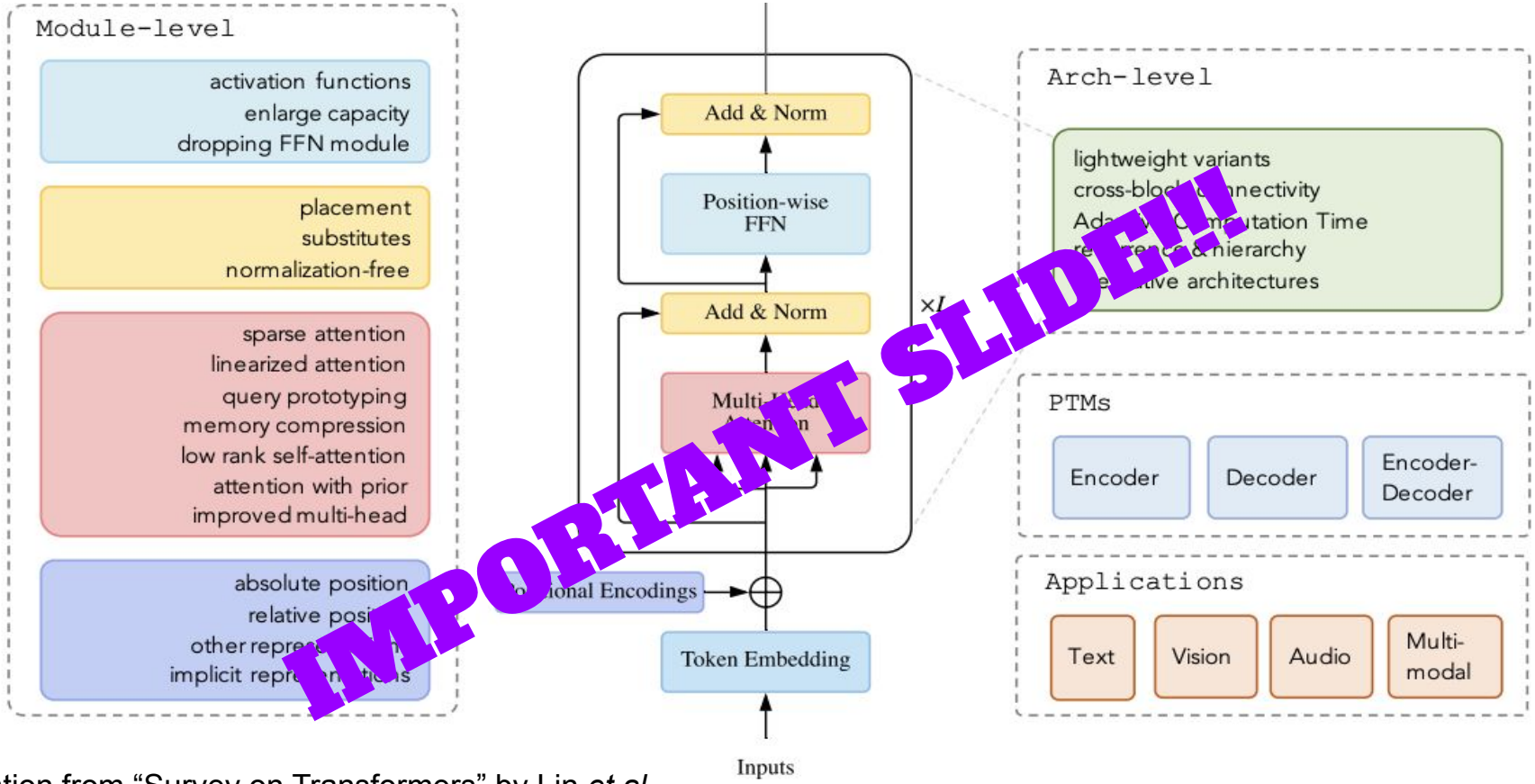
No "induction bias"/structural prior: no assumption of strong local dependencies as in ConvNets: may overfit for small/medium sized data

Positional encoding doesn't efficiently encode ALL info, contained in token position in the input

Module	Complexity	#Parameters
self-attention	$\mathcal{O}(T^2 \cdot D)$	$4D^2$
position-wise FFN	$\mathcal{O}(T \cdot D^2)$	$8D^2$

T - sequence length
D - hidden data representation dimension

Transformers Research Taxonomy



IMPORTANT SLIDE!!!

Illustration from “Survey on Transformers” by Lin *et al*

Attention Scheme Modifications: Types

Sparse Attention: Sparsity bias into the attention mechanism, leading to reduced complexity

Linearized Attention: Linearly approximates softmax in attention weights computations. The attention is computed in reversed order with linear complexity

Prototype and Memory Compression: Reduces the number of queries (Q) or/and key-value (KV) memory pairs to reduce attention matrix dimension

Attention Scheme Modifications: Types, Cont

Low-rank Self-Attention: Leverages “low-rankness” property of self-attention mechanism

Attention with Prior: Considers the integration of prior attention distributions into regular attention mechanisms

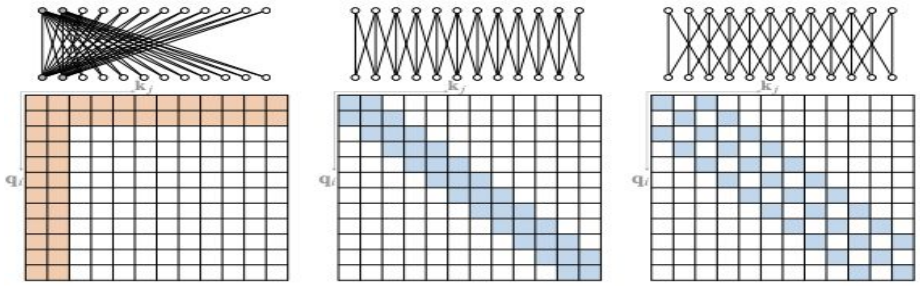
Improved Multi-Head Mechanism: Experiments with alternative multi-head attention approaches

Sparse Attention Variants: Atomic Patterns

Global Attention: Global nodes, ‘hubs” for info propagation between tokens

Sliding window/local attention (band attention): **Restrict query** to “attend to” its neighbor elements: for data types with inherent locality dependencies

Dilated (Strided) Attention: **Increase band attention** receptive field w/o increasing computation complexity

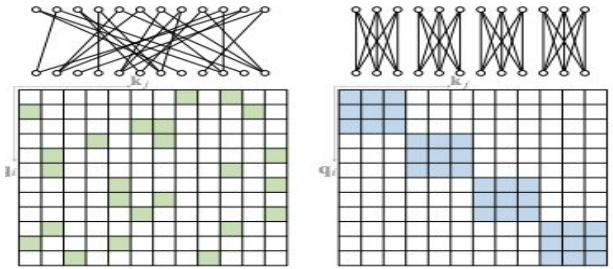


(a) global (b) band (c) dilated

Sparse Attention Variants: Atomic Patterns

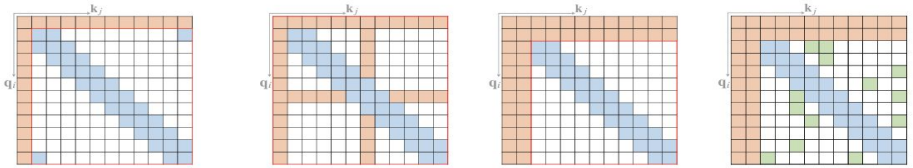
Random Attention: Few tokens randomly sampled for every query to increase "expressiveness" of non-local interactions

Block Local Attention: Split input sequence into several non-overlapping Q blocks; each one has its local memory block. Q-s attend to K-s in the corresponding cluster



(d) random (e) block local

Sparse Attention with Multiple Atomic Patterns: combine patterns

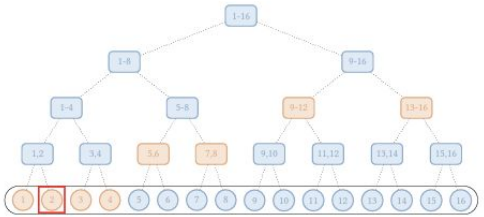


(a) Star-Transformer (b) Longformer (c) ETC (d) BigBird

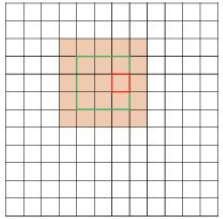
Fig. 5. Some representative compound sparse attention patterns. The red boxes indicate sequence boundaries.

Sparse Attention Variants:

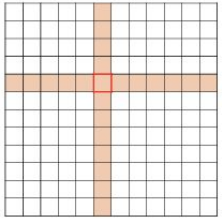
Extended Attention: extended sparse patterns for specific data types (text, image)



(a) BPT

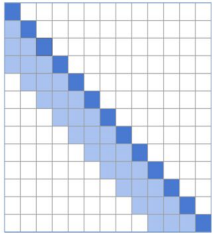


(b) block local (2D)

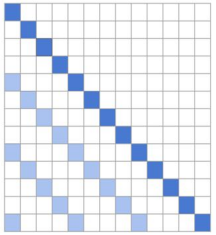


(c) axial (2D)

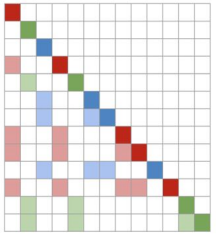
Content-based: select keys likely having high similarity scores with a given query



(a) Local attention



(b) Strided attention



(c) Routing attention

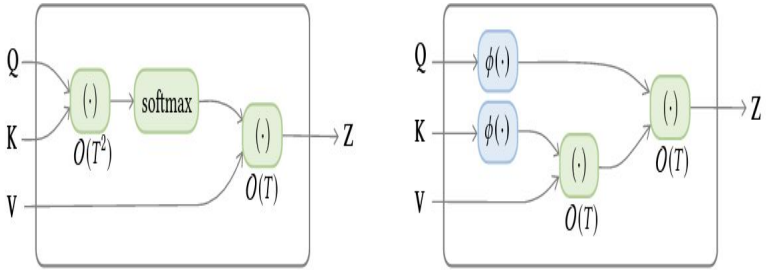
Routing Attention

Attention computed for Qs and Ks in the same K-means clusters

Linearized Attention

- Self-attention is quadratic in terms of the input sequence length T
- **Main Idea:** Decompose *softmax*(QK^T) into ($Q'K'^T$), **first compute** K'^TV & multiply by Q' => linear complexity in T
- Self-attention written as $\mathbf{z}_i = \sum_j \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{j'} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_{j'})} \mathbf{v}_j,$
- Choose kernelized form sim(**x**, **y**) = K(**x**, **y**) = $\phi(\mathbf{x})\phi(\mathbf{y})^\top$
- Use "inverse kernel trick": $\mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top} \mathbf{v}_j$

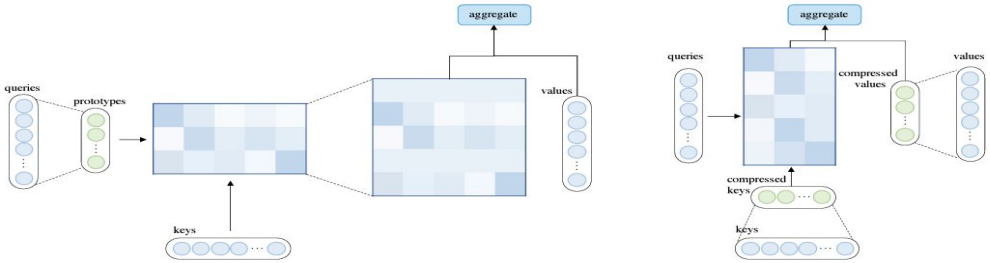
$$= \frac{\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j}{\phi(\mathbf{q}_i)\sum_{j'} \phi(\mathbf{k}_{j'})^\top},$$



(a) standard self-attention (b) linearized self-attention

Query Prototyping & Memory Compression

Reduce attention complexity by reducing the number of Q-s OR KV pairs



(a) Query prototyping (b) Memory compression

Fig. 8. Query prototyping and memory compression.

Query Prototyping: compute attention vectors with several prototypes of Q-s

Clustered Attention: group Q-s into several clusters and compute attention vector for cluster centroids. All cluster Q-s share the attention vector of their centroid.

Low-rank Self-Attention

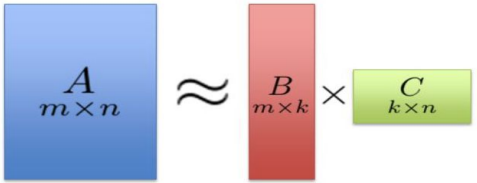
Key Assumption: self-attention matrix A (after softmax) is often **low-rank**. Then

- **A** can be modeled with **low-rank parameterization** or replaced by a low-rank **approximation**

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$\underset{m\times n}{A} \approx \underset{m\times k}{B} \; \underset{k\times n}{C}$$

where $k \ll \min\{m, n\}$



Low-rank Self-Attention, Cont'

Low-Rank Parameterization:

- If rank(A) < sequence length T for small T ⇒ embedding dimension D > T may lead to <u>overfitting</u> (and <u>not just overparameterization</u>)

- **Limit** the dimension of D to explicitly model self-attention low-rank property as an inductive bias

Low-Rank Approximation:

- Use a low-rank matrix approximation to reduce the complexity of self-attention.

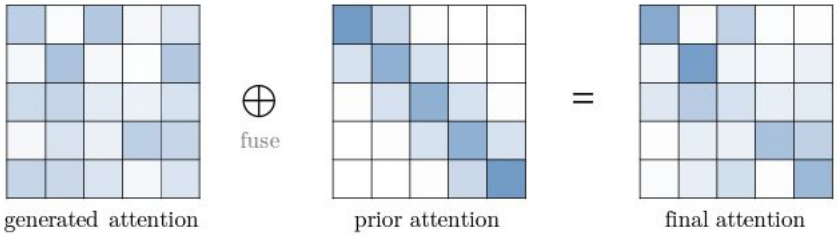
- Similar technique: low-rank approximation of kernel matrices (Performer)

Attention with Prior

Attention mechanism output (for a token E): a weighted sum of input tokens representations; the weights “describe relationships between E & other tokens”.

Attention weights distribution can also come from other sources: **attention prior**

Prior attention distribution: supplement or substitute for distribution generated from inputs (weighted sum of the prior and generated attention scores)

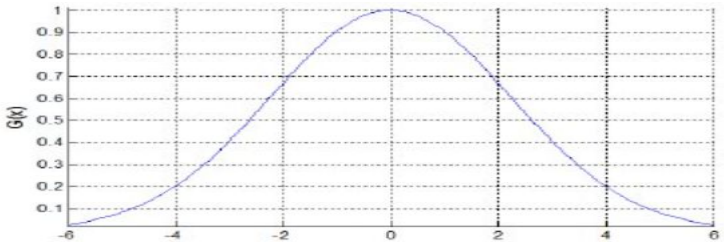


Attention with Prior

Prior that Models locality

- Data with strong locality properties are **"explicitly encoded** as" a prior attention; multiply generated attention scores by 1D Gaussian density filter coeffs

- <u>Attention + Bias G:</u> high Gij indicates a high prior probability of "stronger connection" between i-th and j-th inputs



Distance between i-th and j-th token

Prior as Multi-task Adapters: adapters are task-dependent, trainable modules attached in pretrained network for cross-task efficient parameter sharing

Multi-Head Attention: Recap & Main Issue

- Each head captures different contextual info by mixing tokens in a unique manner

- Expands the model's ability to <u>focus on different positions</u>
  - In translating a sentence "The **animal** didn't cross the street because it was too **tired**", we would want to know which word "it" refers to

- Gives the attention layer <u>multiple "representation subspaces"</u>
  - Multiple sets of Q-K-V matrices
  - Each Q-K-V set used to project token embeddings (vectors from lower encoders) into a different representation subspaces

BUT: No **guarantee** that different heads indeed capture **distinct features**

Head Behavior Modeling: Research Directions

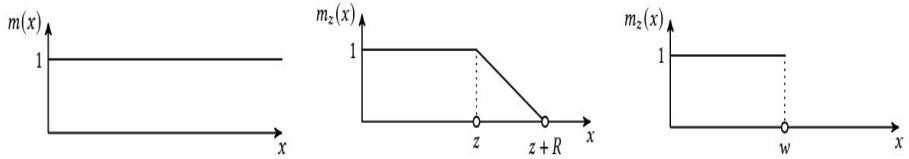
- **Mechanisms** guiding different attention heads behavior
- **Allow** interaction across attention heads

Refined Aggregation: some heads focus their attention distribution mainly in a local context while some others attend to broader contexts.

Restrict the attention spans of different heads to achieve:

Locality: Induces explicit local constraints - best suited for data with locality prior

Efficiency: Scales to very long sequences w/o requiring additional memory footprint & computational time.



(a) mask function for vanilla attention (b) mask function for adaptive span (c) mask function for fixed span

Sequence Position Representations:

Vanilla Transformer: Positional info is encoded as absolute sinusoidal position encodings.

For position index t, the **positional encoding** is a vector:

$$\mathrm{PE}(t)_i = \begin{cases} \sin(\omega_i t) & \text{if } i \text{ is even,} \\ \cos(\omega_i t) & \text{if } i \text{ is odd,} \end{cases}$$

where ω_i is a hand-crafted frequency for each dimension.

Position encodings **are added** to the **token embeddings** and fed to Transformer

Issue: Positional info get lost in the upper layers(propagation via transformers layers)

Remedy: add position representations to inputs of each Transformer layer

Sequence Position Representations: Ideas

Learnable Positional Embeddings: learn a set of positional embeddings for each position

Pros: Learned embeddings adapt position representations to tasks through backprop.

Cons: #embeddings is limited by a max sequence length used before training; <u>not able</u> to handle sequences longer than sequences seen in the training time

Relative Position Representations: pairwise positional relationships between input elements (direction, distance) can be more beneficial than positions of elements

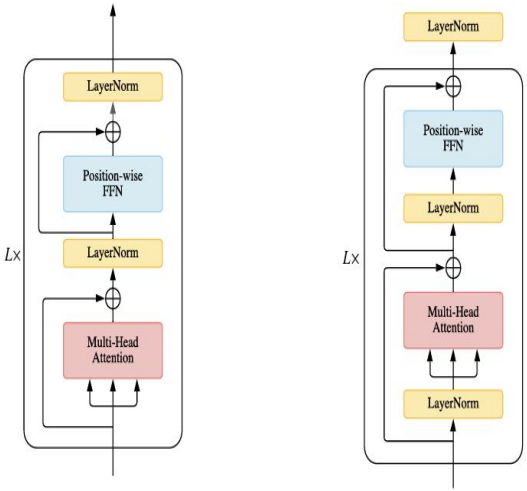
Implicit Position Encoding: encode positional info in word embeddings, by generalizing embedding to continuous (e.g. complex-valued) functions over positions

Layer Normalization(LN): Basic Schemes

- **LN (& residual connection)** stabilize deep NN training (copes with ill-posed gradients, covariance shift etc)
- **Vanilla Transformer:** LN lies between the residual blocks(post-LN)

LN Placement (pre-LN)

- Place LN layer inside the residual connection before the attention or FFN
- Add a LN after the final layer to control the magnitude of final outputs



(a) post-LN (b) pre-LN

PowerNorm

- Relaxes zero-mean normalization
- Input quadratic mean instead of the variance
- Running statistics for the quadratic mean

Layer Normalization(LN) Modifications

Replace LN with scaled ℓ2 normalization:

- Project d-dim input x onto $(d-1)$-sphere of learned radius g: z $= g$ // x //

Use AdaNorm

- Learnable LN parameters **don't work** in most experiments (even increase overfitting)
- Recenter mean/variance derivatives & rescale gradients; play significant role in LN
- **AdaNorm,** a normalization technique w/o learnable params

z $= C(1-ky) \odot$ y, y = $(x-\mu)/\sigma$; C, k are hyperparameters; μ, σ - mean and std of x.

Feed-forward (FF) Layers

Why FFs are an essential Transformer building block?

- Stacking self-attention modules causes a rank collapse problem, leading to attention scores-uniformity (all scores are nearly equal)
- FF layers (& residual connection) **significantly alleviates** this issue

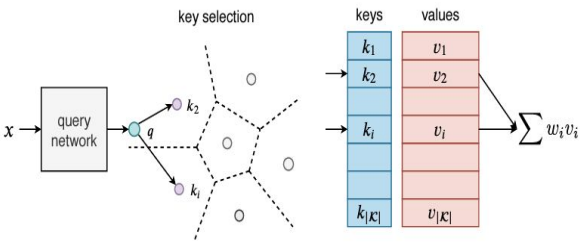
Activation Function Modifications: original Transformer uses ReLU activations

- Swish function $f(x) = x$*sigmoid(βx)
- Gaussian Error Linear Unit (GELU) are the default choice for many pretrained language models.
- Gated Linear Units (GLU) and its variants

Increase FF Layers capacity (expressiveness)

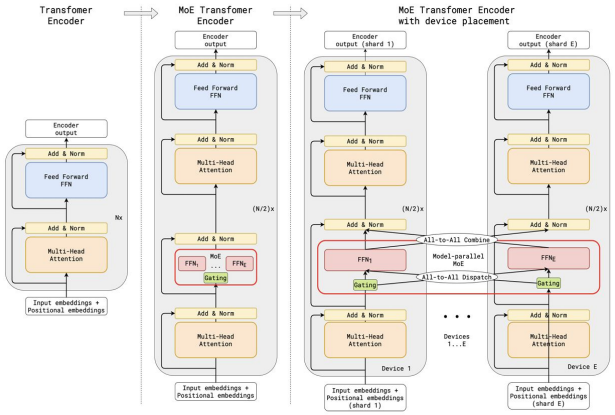
Key Idea: replace FF layers with similar structures having much more parameters

Product-key memory layers:



Sparsely-gated MoE layers (on parallel devices):

MoE layer consists of several FFs (experts) each one similar to vanilla Transformer FF.

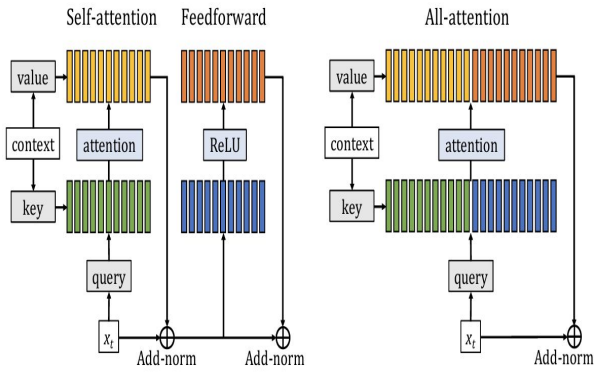


Dropping FF layers: Replace by Attention

Replace FF layers by MORE ATTENTION

FF with ReLU **replaced** with Softmax & with dropped bias is an attention module (attending to **constant** “KV memory”)

Drop the FFs & add set of learnable key-value pairs, concatenated with K-s & V-s generated by input



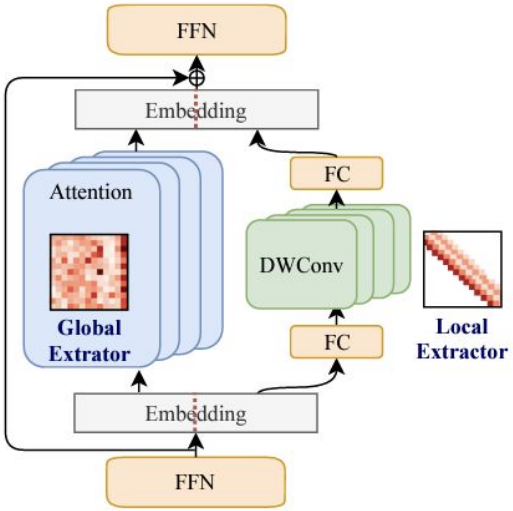
FFs in the Transformer decoder aren’t efficient & can be removed safely with a slight/no loss of performance

Significantly boosts the training and inference speed.

Architecture-Level: Lightweight Transformers

Replace attention with 2-branch structure: Lightweight in terms of model size & computation (for mobile devices)

- Branch 1: uses attention to capture long-range contexts
- Branch 2: uses depth-wise convs & linear layers to capture local dependencies

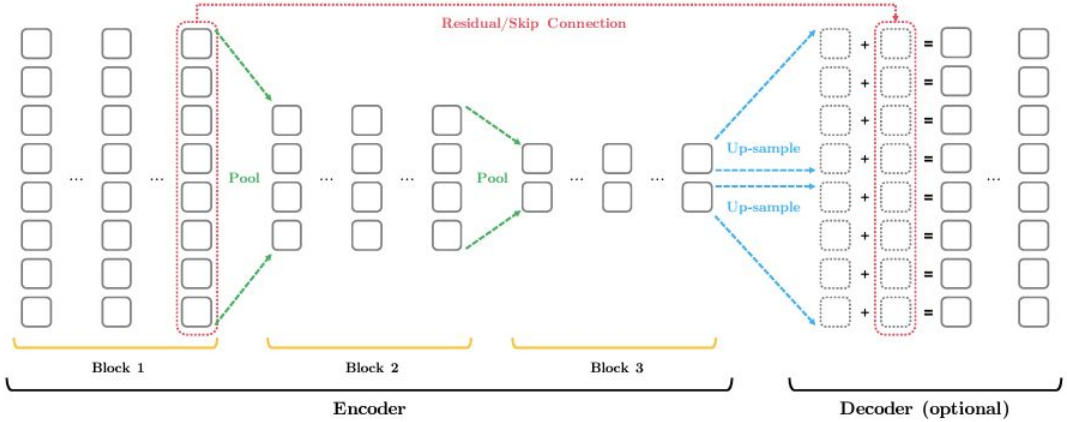


Architecture-Level: Lightweight Transformers

Funnel-like encoder architecture reducing FLOPs & memory

- Hidden representation length of self-attention module output is gradually reduced using pooling along the sequence dimension & recovered using upsampling

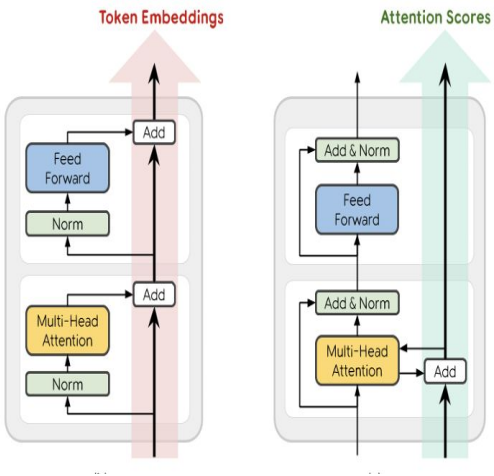
- Can be used to build a deeper or wider model using the same computation budget



Strengthening Cross-Block Connectivity

Observation 1: In vanilla T-r blocks take outputs from the previous block as inputs & outputs a sequence of hidden states

Make tighter inter-block connection: reuse attention distributions from the previous block to guide attention of current block



Observation 2: Decoder cross-attention modules **only** utilize the encoder final outputs, so the error signal traverses along the encoder depth

Possible Problems: optimization issues (vanishing gradients)

Remedy: Decoder representation is a sum of encoder outputs at all encoder layers (and embedding layer) in all cross-attention modules with learned weights

Adaptive Computation Time (ACT):

Key Observation: Vanilla Transformer needs **fixed computation time** to process an input

Approach: make Transformer computation time conditioned on the input (adaptive)

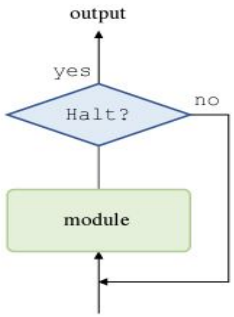
Feature refinement for hard examples: For “hard examples”, a shallow representation may be not enough. Apply more computations to acquire a deeper & more refined representation.

Efficiency for easy examples: For easy examples, a shallow representation may be enough. The network can learn to extract features using reduced computation time.

ACT Transformers: Dynamic Halting

Per-position dynamic halting mechanism computing a halting probability P(symbol) for each token at every time step

- Iteratively refines all tokens representations using self-attention mechanism
- If P(symbol) > threshold, symbol's representation remains unchanged for next timesteps
- Network run stops when all symbols halt or predefined maxSteps is reached



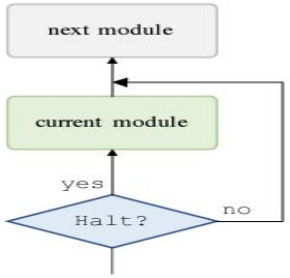
(a) dynamic halting

ACT Transformers: Conditional Skipping

- **Skip module invocation** if it is not needed

- Adds a control network g(.) at each self-attention & FF layer to decide whether to skip the current layer

- An auxiliary loss encouraging model to adjust g(.)-s to match computation cost to the available computation budget



(b) conditional skipping

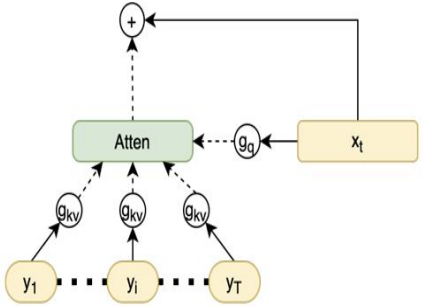


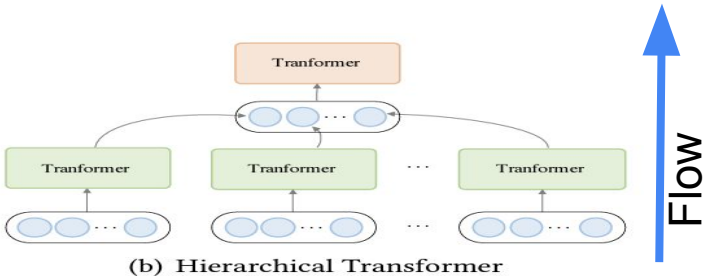
Figure 2. Conditional Computation Attention Layer.

Divide-&-Conquer(DaC) Transformers

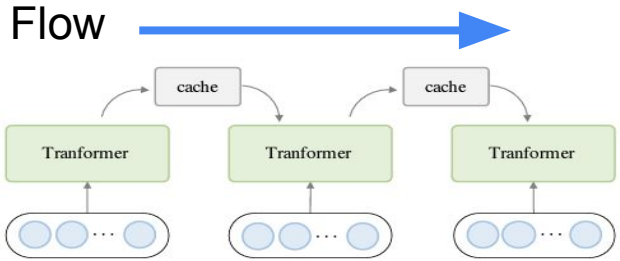
An essence: tackle Transformer quadratic complexity with **divide-&-conquer** strategy

How? split an input sequence into finer segments & process by smaller Transformers

- **Recurrent:** cache memory is maintained to incorporate the history info
- **Hierarchical:** decompose inputs hierarchically into pieces of finer granularity; feed outputs to Transformer encoder, aggregate & process by high-level Transformer



(b) Hierarchical Transformer



(a) Recurrent Transformer

DaC Transformers: Recurrent, Example 1

Extends the cache with two levels of memory

- Keeps a fine-grained memory of past activations, which are compressed (cached) into coarser compressed memories
- Cached segments are used to augment the current segment (activations from older segments are discarded)
- Compression operations on older activations stores them in compressed memory

Extends **theoretical** max **Transformer-XL** context length from N_{mem} to $L(N_{\text{mem}} + cN_{\text{cm}})$; c is the compression rate, N_{cm} compressed memory length

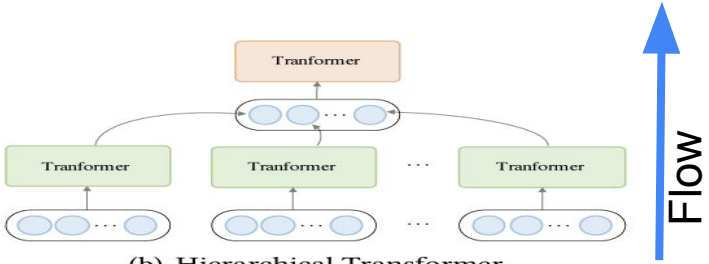
DaG Transformers: Hierarchical

An essence:

- Decompose inputs hierarchically into pieces of finer granularity; feed the outputs to Transformer encoder, aggregate & process by high-level Transformer

Advantages:

- Hierarchical modeling allows the model to handle long inputs with limited resources
- Has the potential to generate richer representations beneficial to many tasks.



(b) Hierarchical Transformer

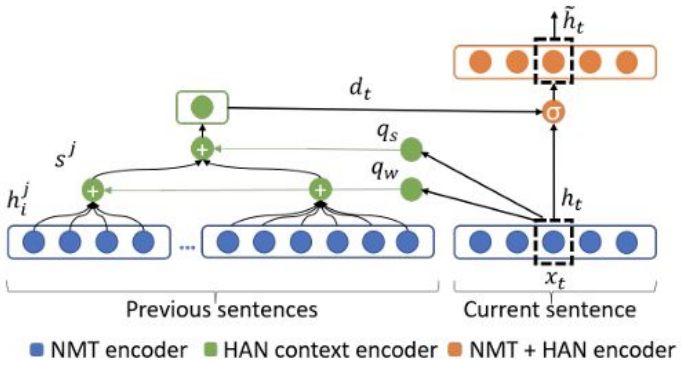
Hierarchical for long sequence inputs, Example 1.

Key Idea: Use hierarchical Transformers to model long-range dependencies for long input lengths

Neural Machine Translation, 2 levels of abstraction

- **Word-level:** summarizes info from EACH previous sentence j into a vector s_j
- **Sentence-level:** aggregates info from ALL previous sentences up to sentence t

.

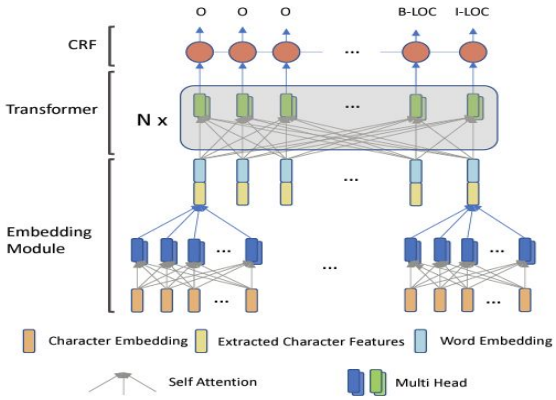


Hierarchical for richer representations, Example 1

Named Entity Recognition Task

- Low-level Transformer encoder encodes char-level features, concatenates with word embeddings & feed them to high-level Transformer encoder.

- Richer features (+character-level), alleviates data sparsity problem & out-of-vocabulary (OOV).



Is there life without Attention?

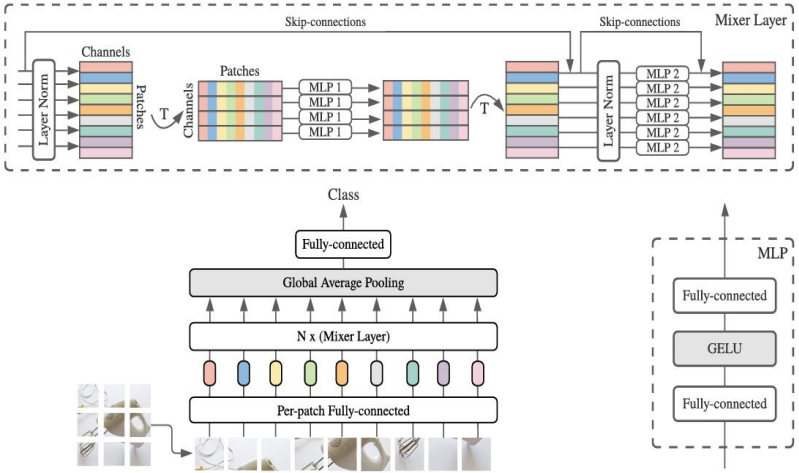
Transformer's Revolution Reasons: an attention mechanism (seasoned with FF and res-connections sauce)

Question: can we ACHIEVE similar performance WITHOUT attention mechanism

Attention Replacement, Example 1

MLP Mixer, 2021 (& Pay Attention to MLPs, 2021): Attention is Replaced by:

Token-mixing MLP & Channel-mixing MLP, each consisting of 2 fully-connected layers and a GELU nonlinearity



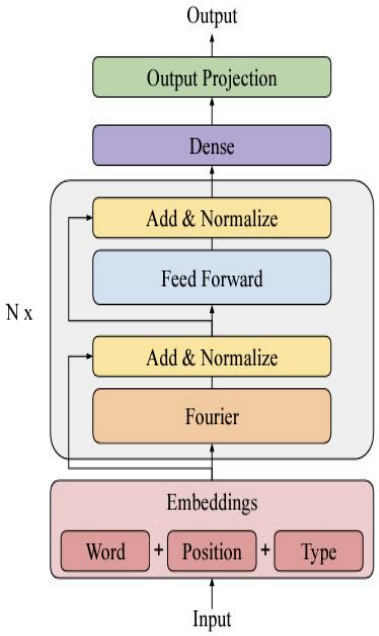
	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
Pre-trained on ImageNet-21k (public)						
• HaloNet [51]	85.8	—	—	—	120	0.10k
• Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
• ViT-L/16 [14]	85.30	88.62	94.39	72.72	32	0.18k
• BiT-R152x4 [22]	85.39	—	94.04	70.64	26	0.94k
Pre-trained on JFT-300M (proprietary)						
• NFNet-F4+ [7]	89.2	—	—	—	46	1.86k
• Mixer-H/14	87.94	90.18	95.71	75.33	40	1.01k
• BiT-R152x4 [22]	87.54	90.54	95.33	76.29	26	9.90k
• ViT-H/14 [14]	88.55	90.72	95.97	77.63	15	2.30k
Pre-trained on unlabelled or weakly labelled data (proprietary)						
• MPL [34]	90.0	91.12	—	—	—	20.48k
• ALIGN [21]	88.64	—	—	79.99	15	14.82k

Attention Replacement 2

FNet 2021: Attention is Replaced by:

- Simple token mixing mechanism: parameterized matrix multiplications
- Mixing over the sequence dimension & over the embedding dimension
- Mixing: 2 discrete Fourier transforms over 2 different dimensions

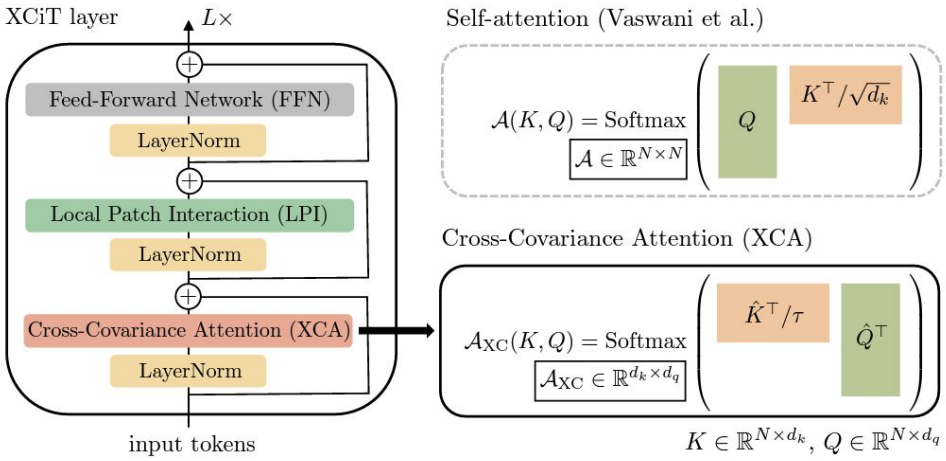
$$y = \Re\left(\mathcal{F}_{\text{seq}}\left(\mathcal{F}_{\text{h}}(x)\right)\right)$$



Attention Replacement 3

XCiT: Cross-Covariance Image Transformers, 2021 - Attention is Replaced by:

- "Transposed" self-attention operating across feature channels rather than tokens
- Interactions are based on the cross-covariance matrix between keys and queries
- Has linear complexity in #tokens; efficient processing of high-resolution images



Transformers: Future Research Directions 1

Theoretical Analysis of Immense Transformers Capabilities:

- Transformers were “empirically” shown to have a larger capacity (expressiveness) than CNNs and RNN & hence to effectively “learn” huge amount on data

- When Transformer trained on sufficiently large and diverse data, its performance is better than CNNs/RNNs.

- **An intuitive explanation:** Transformer makes **NO** assumptions on the data structure and thus is more flexible.

.

But we still DO NOT have a Full

Theoretical Explanation for the Transformer Power

Transformers: Future Research Directions 2

Better Global Interaction Mechanism beyond Attention.

- Many studies have shown that full attention is unnecessary for most token; it is inefficient to indistinguishably compute attention between **ALL** tokens

- **Question:** how to model global interactions efficiently?

- **Way 1:** Self-attention module is FFN with dynamic weights, aggregating non-local info with dynamic routing - **other dynamic routing mechanisms??**

- **Way 2:** The global interaction can be also modeled by other neural networks types, such as **memory-enhanced models**

Transformers: Future Research Directions 3

Unified Framework for Multimodal Data:

- **Integrating multimodal data** is proven to be useful for task performance improvement

- General AI needs to capture the **semantic relations across different modalities**.

- Transformers excel in text, image, video & audio domains, so what about building unified framework fully leveraging **interconnections of multimodal data?**

Transformers' History: More to Follow…

Stay Tuned...