X-formers

Arulselvan Madhavan

Need for Transformer variants

- Vanilla transformers are inefficient at processing long sequences
  - Computation and memory complexity of self-attention is high
- Doesn’t use structural/inductive bias of input data
  - Variants: Introduce structural bias, regularization, pre-training on large-scale unlabeled data
- To handle specialized downstream tasks/applications

Taxonomy for X-former variants

- Many variants address multiple needs
  - Example: Sparse Transformers - addresses computational requirements and adds structural prior
- Taxonomy should be based on the ways in which they improve vanilla x-former
  - Architecture modification
  - Pre-training
  - Applications

Vanilla Transformer

- Seq2Seq model
  - Encoder - usually multiple stacked on top of each other
  - Decoder - - usually multiple stacked on top of each other
- Encoder
  - Multi-head self-attention
  - Position-wise feed-forward network(FFN)
  - Residual connection
  - Layer Norm
- Decoder
  - Multi-head self-attention (autoregressive in nature)
  - Cross attention with encoder state
  - FFN

Vanilla Transformer

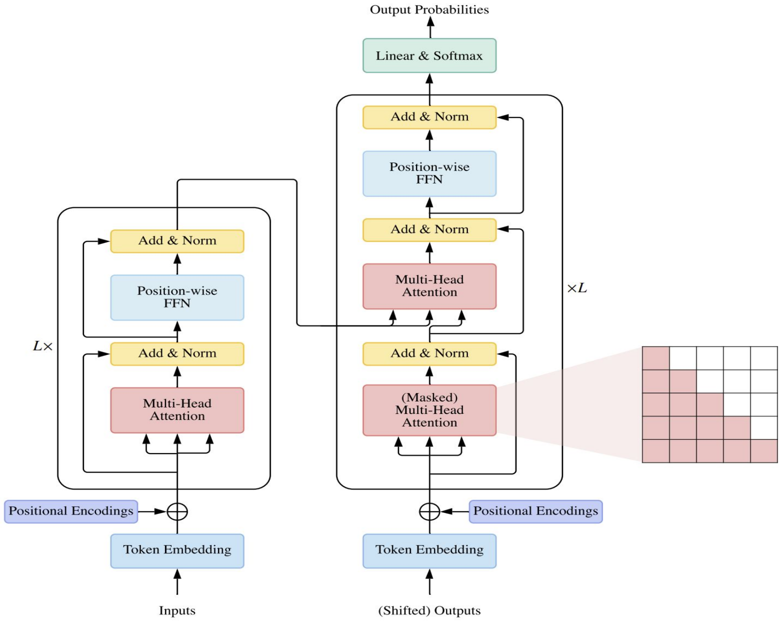


Fig. 1. Overview of vanilla Transformer architecture

Attention Module

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{D_k}}\right)\mathbf{V} = \mathbf{A}\mathbf{V},$$

$$\mathrm{MultiHeadAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_H)\mathbf{W}^O,$$

$$\text{where } \mathrm{head}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V).$$

Self vs Masked vs Cross Attention

- Self: Q = K = V = X
- Masked: Restrict attention to certain parts of input. Ex: Decoder restricting attention to previous positions
- Cross-Attention:
    - Queries are projected from the previous layer.
    - Keys and values are projected from the encoder output

Position wise FFN

$$\mathrm{FFN}(\mathbf{H}') = \mathrm{ReLU}(\mathbf{H}'\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2,$$

- Provides rich representation to the attention outputs

Residual connection + Layer Norm

$$H' = \mathrm{LayerNorm}(\mathrm{SelfAttention}(X) + X)$$

$$H = \mathrm{LayerNorm}(\mathrm{FFN}(H') + H'),$$

- Residual connections
  - helps with vanishing gradient problem
  - retain information about the original input

Transformer - model usage

- Encoder-Decoder
  - Full transformer architecture
  - Seq2seq modeling like Neural machine translation
- Encoder
  - classification / sequence labeling
- Decoder
  - Sequence generation

Model Analysis

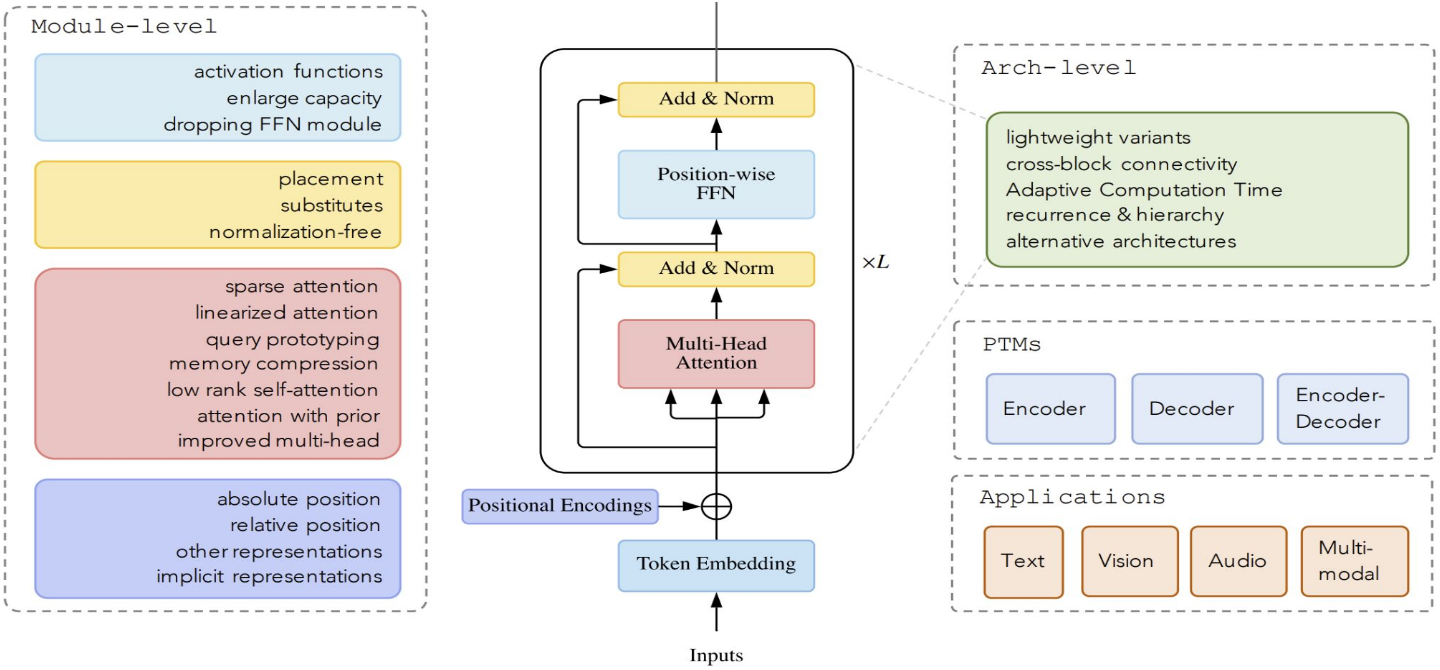
Module	Complexity	#Parameters
self-attention	$O(T^2 \cdot D)$	$4D^2$
position-wise FFN	$O(T \cdot D^2)$	$8D^2$

- T - Sequence length; D - Hidden dimension
- Short-sequence: Bottleneck is FFN
- Long-sequence: Bottleneck Self-Attention; In computation and space
    - Pixel level image generation, long text document modeling are infeasible

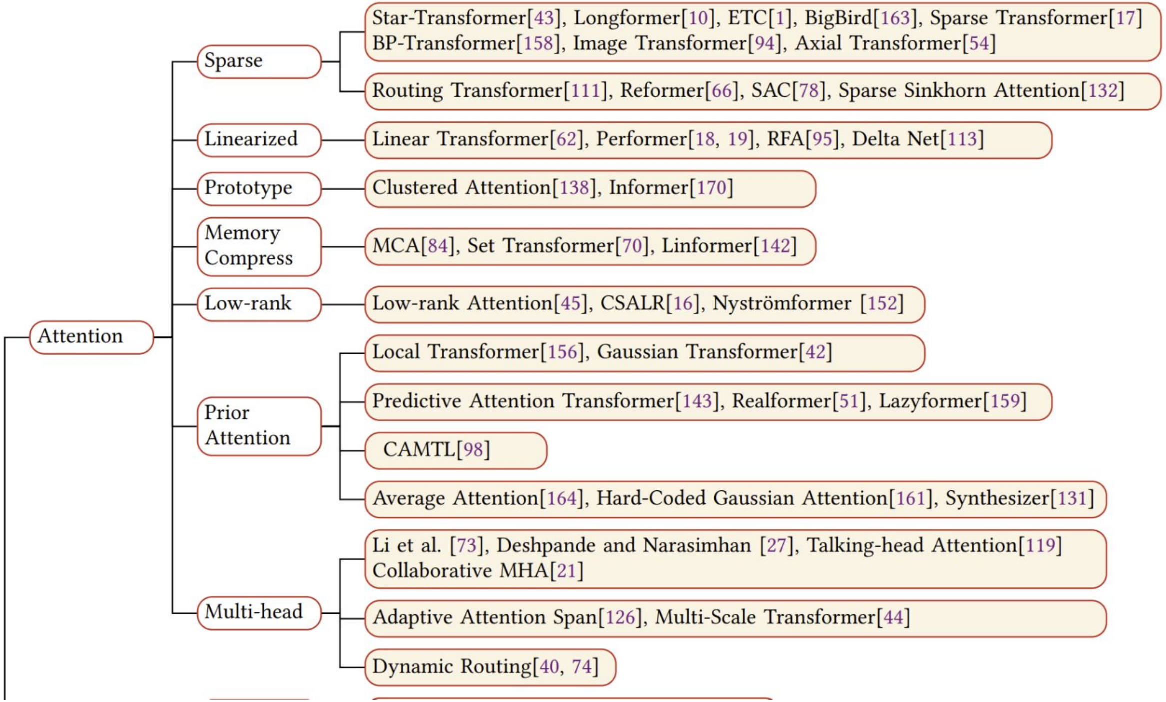
Self-Attention vs Conv vs FC vs Recurrent

- Conv vs SA: Convolutional network need to stack conv layers deep to get global receptive field. SA can get global receptive field with constant number of layers.
- FC vs SA: SA is flexible in handling variable-length inputs
- Recurrent vs SA: Constant sequential operations make SA more parallelizable and better at long-range modeling
- Inductive bias:
  - Conv layers impose translation invariance and locality with shared local kernel functions.
  - Recurrent layers impose temporal invariance and locality
  - TF make very little use of structural info and is prone to overfitting on small-scale data

Taxonomy of Transformers



Attention level variants



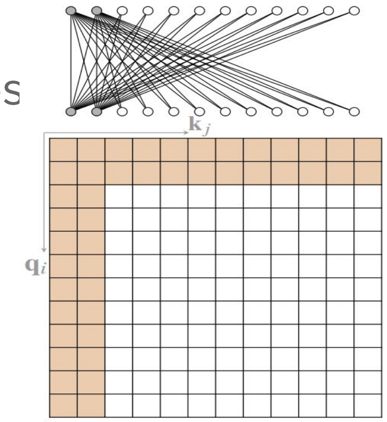
Sparse Attention

- Learned Attention matrix is often sparse across most data points
- Incorporate this information to reduce the number of query-key pairs computed

$$\hat{\mathbf{A}}_{ij} = \begin{cases} \mathbf{q}_i \mathbf{k}_j^\top & \text{if token } i \text{ attends to token } j, \\ -\infty & \text{if token } i \text{ does not attend to token } j, \end{cases}$$

Sparse Attention - Global Attention

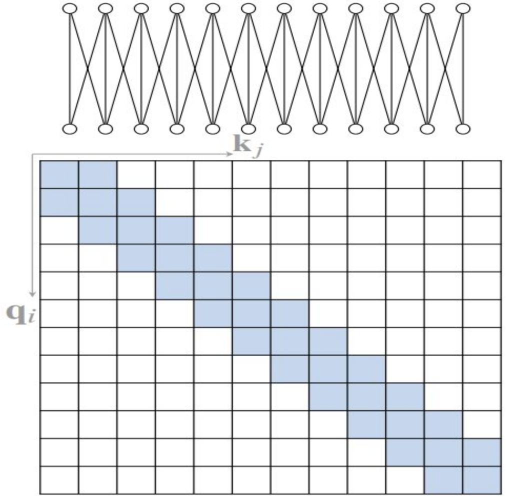
- Global nodes: Retain the ability to model long range deps
- Serve as the hub for information propagation between nodes
- Global nodes attend to all nodes in the sequence



(a) global

Sparse Attention - Band Attention

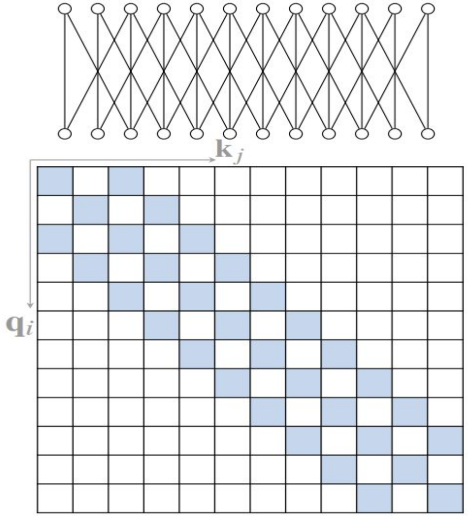
- Sliding window attention
- Aka local attention
- Restrict each query to attend to it its neighbor nodes



(b) band

Dilated Attention

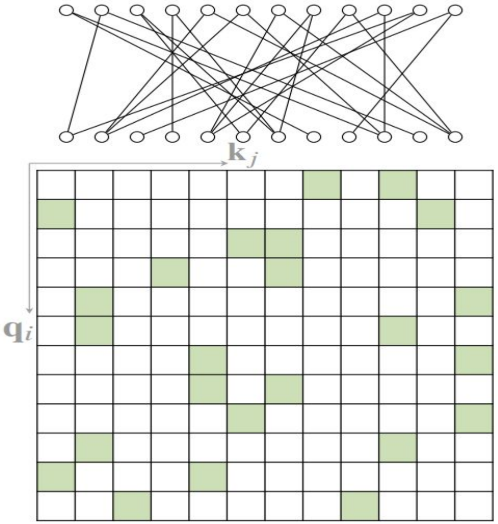
- Idea is similar to dilation in CNN
- Increased receptive field without increasing computation
- Aka strided attention



(c) dilated

Random Attention

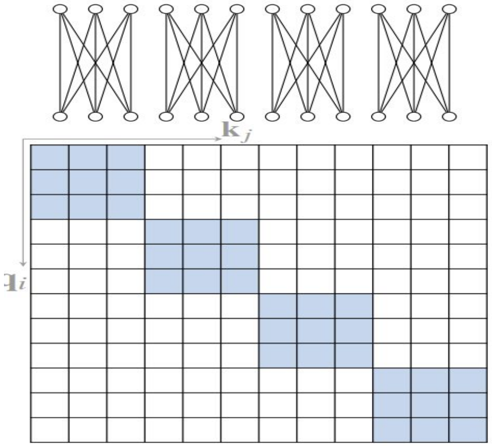
- Randomly sampled nodes for each query
- Random graphs can have similar spectral properties with complete graphs that leads to fast mixing time for random walking on graphs



(d) random

Block Local Attention

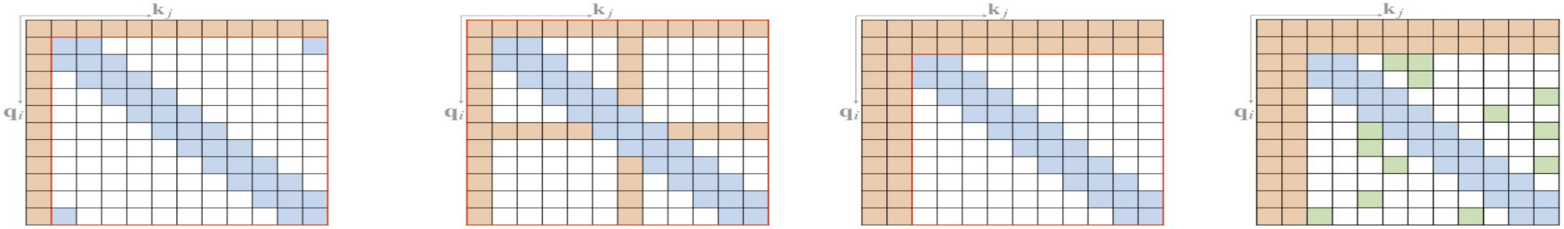
- Segments input sequence into several non-overlapping query blocks
- Queries in a query block attend to only the keys in the corresponding block



(e) block local

Compound Sparse Attention

- Combination of multiple attention patterns
- Star-Transformer: Band Attention + Global Attention
- Longformer: Band Attention + Internal global-node attention + dilated window attention in upper layers
- Extended Transformer Construction(ETC): Band + External-node global attention
- BigBird: Band + Global + Random Attention to approximate full attention



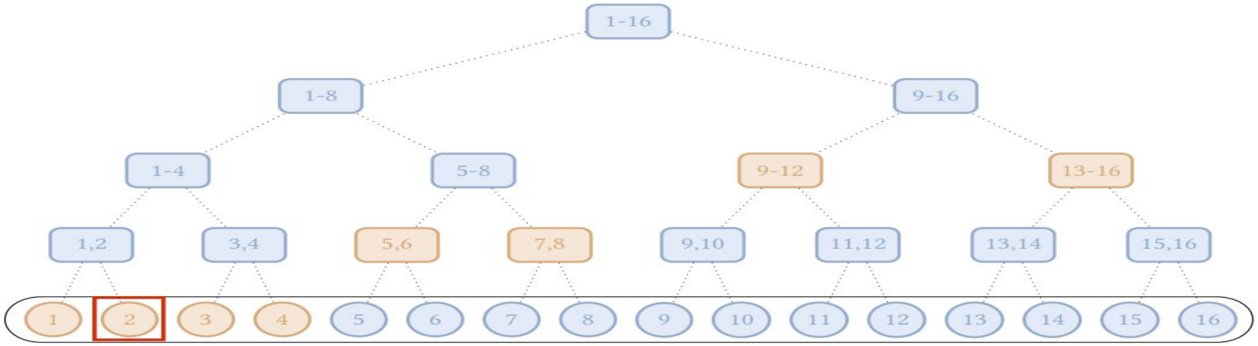
(a) Star-Transformer (b) Longformer (c) ETC (d) BigBird

Compound Sparse Attention

- Sparse Transformer: Different sparse patterns for different types of data
- Data with periodic structure(Images):
  - Composition of band attention and strided attention
- Data without periodic structure(text):
  - Block local attention + global attention

Extended Sparse Attention

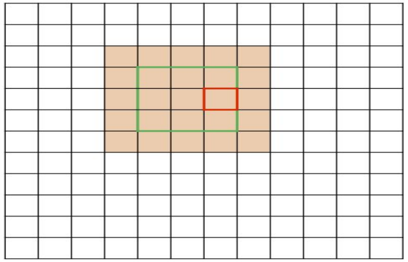
- For text: BP transformer
  - Binary tree with tokens as leaf nodes and internal nodes as span nodes containing many tokens



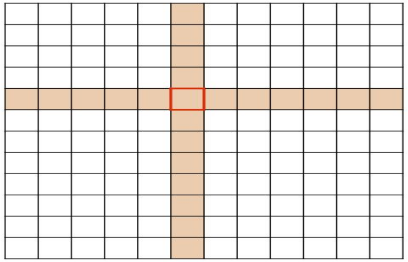
(a) BPT

Extended Sparse Attention

- Image Transformer:
    - flattens image pixels in raster-scan order and applies block local sparse attention
    - 2D block local attention with query and memory blocks arranged in 2D
- Axial Transformer:
    - Applies independent attention modules over each axis of the image
    - Mixes information along one axis while keeping information from the other axis independent



(b) block local (2D)



(c) axial (2D)

Content-based Sparse Attention

- Sparse graph is created based on the input content
- Idea: Select those keys that are likely to have large similarity scores with the given query
- Routing Transformer:
  - K-means clustering to cluster both queries and keys to the same set of centroid vectors
  - Each query attends only to the keys that belong to the same cluster
- Reformer
  - Locality sensitive hashing to select key-value pairs for each query
  - Each query attends only to the keys within the same hashing bucket
  - LSH function to hash queries and keys into several buckets with similar items fall in the same bucket with high probability

Content-based Sparse Attention

- Sparse Adaptive Connection(SAC)
  - Input sequence as a graph and learns to construct attention edges to improve task specific performances
  - Uses adaptive sparse connection using LSTM edge predictor to construct edges between tokens
  - Uses reinforcement learning to predict edges
- Sparse Sinkhorn Attention
  - Splits queries and keys into several blocks and assigns a key block to each query block
  - The assignment of key block to query block is done via sorting network which uses Sinkhorn normalization
  - Permutation matrix representing the assignment

Linearized Attention

- Attention computation is quadratic in sequence length T.
- O(T^2 D)
- How to get linear computation complexity?

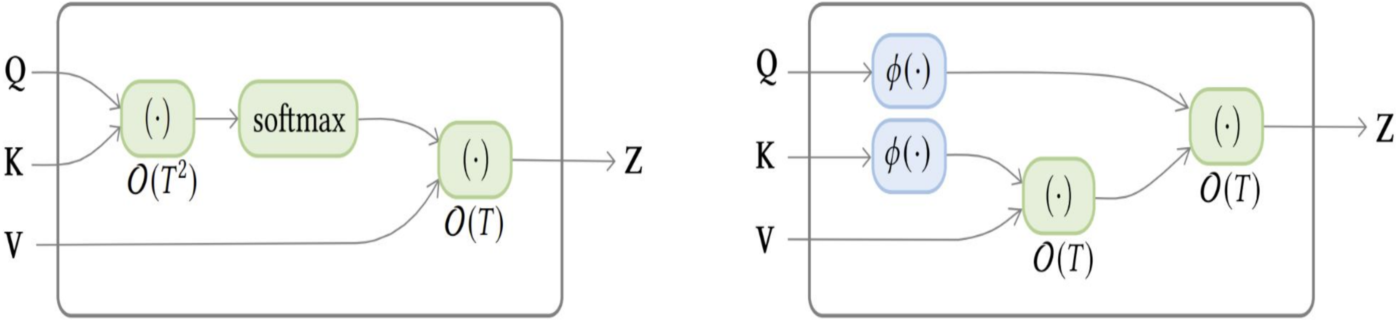
$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}}\right)\mathbf{V} = \mathbf{A}\mathbf{V},$$

- N, M - sequence lengths of queries and keys/values
- Dk; Dv - dimensions of queries/keys and values
- Q - N x Dk ; K - M x Dk; V - M x Dv;
- Q.K^T - N x M; (Q.K^T)V = N x Dv => O(T^2.D) => O(T^2) when T >> D
- Q(K^T.V) - O(T.D^2) => O(T) when T >> D

Linearized Attention

- Unnormalized attention - A_hat = exp(Q.K^T)
- Replace this unnormalized attention with a torch.matmul(phi(Q), phi(K)^T)
- Phi - feature map applied in row-wise manner
- Standard Self-Attention $Z = D^{-1}\hat{A}V, \text{ where } D = \mathrm{diag}(\hat{A}1_T^T);$
- Because A_hat = exp(Q.K^T) = phi(Q).phi(K)^T
- And, Dot product is associative
- phi(Q).(phi(K)^T . V)

Linearized Attention



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

Linearized Attention

- Feature map used in Linear Transformer: phi(x) = elu(x) + 1
- The feature map doesn't approximate dot product attention
- Empirically performs on par with standard transformer
- Performer
  - Uses random feature maps that approximate the scoring function of Transformer

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}[f_1(\omega_1^\top \mathbf{x}), \cdots, f_m(\omega_m^\top \mathbf{x}), \cdots, f_l(\omega_1^\top \mathbf{x}), \cdots, f_l(\omega_m^\top \mathbf{x})],$$

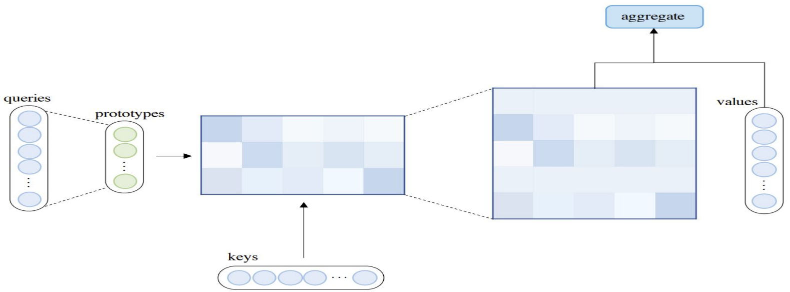
$f_1, \cdots, f_l : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^D \to \mathbb{R}$.

Performer variants

- They vary in the number of functions the feature maps take and h(x)
- Some variants
  - L = 2; f1 = sin; f2 = cos; h(x) = exp(l2norm(x))/2
  - L=1; f1 = exp; h(x) = exp(-l2norm(x))/2
  - L=1; f1 = RELU; h(x)=1 (Protein sequence modeling and NMT find this effective)

Query Prototyping and Memory Compression

- Idea: Reduce the number of queries or key-value pairs
- Prototype queries:
  - Several prototype queries serve as the main source to compute attention matrix
  - Clustered Attention: groups queries into clusters and compute attention distribution for cluster centroids. All queries in the cluster share the attention distribution of the centroid
  - Informer selects prototype queries using sparsity measurement
    - KL divergence between query's attention distribution and discrete uniform distribution
    - Select Top-u queries as prototype



(a) Query prototyping

Memory Compressed Attention

- Idea: Reduce the number of key-value pairs before applying attention
- MCA:
  - Uses strided convolution with local attention
  - Can capture global context
  - Reduces number of keys and values by a factor of kernel size k
- Set Transformer
  - External trainable global nodes
    - Summarizes information from inputs
    - Summarized representations serve as a compressed memory that the inputs attend to
    - Shown to perform with linear computational complexity in sequence length

Other Compressed variants

- Linformer
  - Linear projections to project keys and values from length n to smaller length k
  - Assumes an input sequence length
  - Cannot be used autoregressively
- Poolingformer
  - Two-level attention
    - Sliding window
    - Compressed memory attention
  - Compression achieved by max pooling, pooling with dynamic convolution

Low rank Self-Attention

- Idea: Self-Attention matrix is often low-rank
- Low-rank parameterization:
  - Model the low-rank property as inductive bias by limiting the dimension of query/keys Dk
  - Self-attention matrix - decomposed into low-rank attention module with small Dk to capture long-range non-local interactions and a bank attention module to capture local dependencies
- Low-rank Approximation:
  - Performer variants were inspired from kernel approximation with feature maps
  - Decompose attention matrix A into C.G.C where feature maps are used to approximate G

Nyström based Attention approximation

- Another form of low rank approximation
- Idea:
  - Select m landmark nodes from T inputs with down-sampling(strided average pooling)

$$\tilde{\mathbf{A}} = \operatorname{softmax}\left(\mathbf{Q}\tilde{\mathbf{K}}^{\top}\right)\left(\operatorname{softmax}\left(\tilde{\mathbf{Q}}\tilde{\mathbf{K}}^{\top}\right)\right)^{-1}\operatorname{softmax}\left(\tilde{\mathbf{Q}}\mathbf{K}^{\top}\right)$$

- Inverse doesn't always exist
- Identity matrix could be added to make sure that the inverse always exists
- Nyström-former used Moore-Penrose pseudoinverse in place of inverse

Attention with Prior

- Attention matrix = softmax(QK^T)
  - Generated using input sequences
- Prior: Attention distribution stemming from source other than input, referred as prior
- Fusion of two attention distribution matrices can be done by computing a weighted sum of the scores corresponding to the prior and generated attention before applying softmax

Priors that model locality

- Text data usually exhibits preference for locality
- This can be modelled as a prior attention
- Multiply the generated attention distribution with some Gaussian density
  - Gaussian density Gij indicates higher probability that the i-th input attend to j-th input
  - Pi predicts a central position for each query Qi using a FFN

$$G_{ij} = -\frac{(j - p_i)^2}{2\sigma^2},$$

- Gaussian Transfomer
  - $G_{ij} = -|w(i-j)^2 + b|,$

Priors from lower modules

- Observation: Attention distributions are similar in adjacent layers
- Idea: Use attention distribution from previous layer as a prior

$$\hat{\mathbf{A}}^{(l)} = w_1 \cdot \mathbf{A}^{(l)} + w_2 \cdot g(\mathbf{A}^{(l-1)}),$$

- Predictive Attention Transformer
  - Apply a 2D convolution to previous attention scores
  - Final attention = convex combination of generated attention score and convoluted scores
- Realformer
  - Directly adds previous attention scores resembling a residual skip connection
- Lazyformer
  - Shares attention maps between a number of adjacent layers
  - Attention maps computed once but reused several times as prior

Multi-task adapters

- Adapters: Task-dependent modules attached in specific locations of a pre-trained network for cross-task efficient parameter sharing
- Idea: Use trainable attention priors that depends on task encoding

$$M(\mathbf{z}_i) = \bigoplus_{j=1}^{m} A'_j(\mathbf{z}_i), \qquad A'_j(\mathbf{z}_i) = A_j \gamma_i(\mathbf{z}_i) + \beta_i(\mathbf{z}_i)$$

- Zi depends on task specific encoding
- Gamma and Beta are Feature wise linear modulation functions
- This trainable prior is added to attention scores of upper layers in pre-trained transformers

Attention with only prior

- Idea: No generated attention. Only prior attention distribution
- Average Attention network:
  - Discrete uniform distribution as the only source of attention distribution
  - A FFN gating layer on top of the average attention module
  - Used in decoder side
  - Decoder like RNN and avoids O(T^2) complexity
- Gaussian distribution as prior
  - Achieves comparable performance to baseline machine translation
- Synthesizer
  - Replace generated attention with learnable, randomly initialized attention scores
  - Replace attention score with a FFN that is conditioned on the querying input

Multi-head Attention variants

- Multi-heads: Allows to attend to information from different representation subspaces at different positions
- Do the attention heads capture distinct features?
- Attention heads don’t interact with each other
- Variants
    - Encourage distinction learned features between attention heads
    - Add mechanisms to guide attention head behaviors
- Regularization terms to loss function to encourage diversity among attention heads
- Observation: Multiple attention heads pay attention to CLS and SEP tokens
    - Auxillary loss: Frobenius norm between attention distribution maps and predefined attention patterns

Multi-head Attention variants

- Talking-head Attention
  - Linearly projects the generated attention scores from hk to h heads
  - Applies softmax
  - Projects output to hv heads for value aggregation
    - This moves information between attention heads in a learnable fashion
- Collaborative Multi-head Attention
  - Shared query and key projection
  - In the normal attention shown below, Wq and Wk are different for each head

$$\mathrm{MultiHeadAttn}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_H)W^O,$$

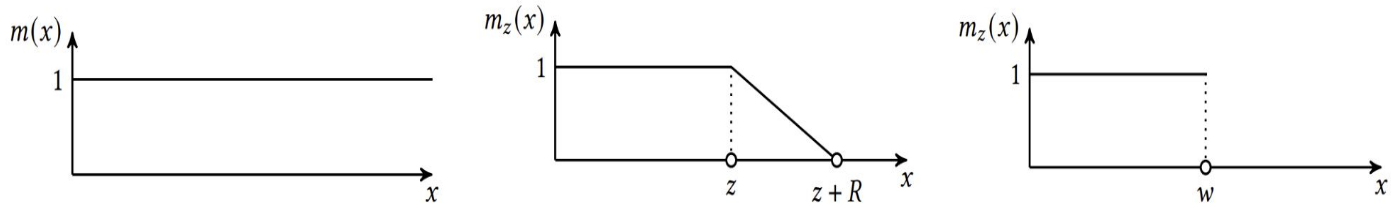
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

Multi-head with Restricted spans

- Full attention span: A query can attend to all key-value pairs
- Observation: Some heads focus on local context and some heads attend to broader contexts
- Restricting attention spans on some heads can save memory footprint and computational time while allowing to extend to long sequences
- Learnable attention span
    - Restricted attention span are implemented via attention masks
    - Learnable scalar z represents a distance/span value
    - Character-level language modeling show adaptive-span models outperform baseline models

Multi-head with Restricted span

- Multi-scale Transformer
    - Fixed attention span with different heads in different layers using a different max span
    - Scales are designed from an intuitive linguistic perspective and empirical observation
    - Higher layers have more scale/span length
    - Lower layers are confined to smaller scale



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

Multi-head with Refined Aggregation

- Multi-head attention usually concatenates attention head outputs and projects them to final output representation

$$\text{MultiHeadAttn}(Q, K, V) = \sum_{i=1}^{H} \text{Attention}(Q\mathbf{W}_i^Q, K\mathbf{W}_i^K, V\mathbf{W}_i^V\mathbf{W}_i^O).$$

- Idea: This aggregation doesn't fully exploit the expressiveness of multi-head attention
- Alternate aggregation:
  - Dynamic routing
  - EM routing
- Computational overhead is alleviated by applying complex aggregation to lower layers

Multi-head variants

- Multi-query attention
    - Key-value pairs are shared among attention heads
    - Faster decode operations with minimal quality loss
- Usual Head size: Key dimension / H
    - Variant: Attention head size to be same as input sequence length

Module-level variant - Position

- Self-Attention and position-wise FFN are permutation equivariant
  - Same set of inputs regardless of permutation will product same set of outputs with permutation matching the input permutation
- Position information has to be added
- Variants:
  - Absolute position representations
    - Position encoding for index t of size Dm is generated
    - Position encoding is added to token embeddings and fed to transformer

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

Position Encoding variants

- Learned position embeddings for each position
  - Number of embeddings is limited to a maximum sequence length
  - Loses inductive bias
- Sinusoidal position representation with each frequency learned from data
  - More robust compared to hand-crafted position representation
- FLOATER
  - Positional representation as continuous dynamical system and adopts Neural ODE to enable end-to-end training
  - Inductive and flexible with a fully learnable approach
- Add positional encodings to each layer input and not just the first layer since positional information gets lost after first few layers

Relative Position Representation

- Represent positional relationships between tokens as opposed to position of tokens
- Pairwise positional relationships between input elements could be more beneficial than position of elements
- Music Transformer uses relative position representation
- Transformer-XL use a sinusoidal encoding to represent positional relationship but fuses contents and position information
- DeBERTa utilizes position embeddings similar to Transformer-XL

Other Representations

- TUPE - Transformers with Untied Position Encoding
  - absolute position-to-position term
  - Relative content-to-content term
  - Bias term representing relative positional relationships
- Roformer
  - Rotary Position Embedding to represent the position of a token by multiplying the affine transformed embedding of the input by a rotatory matrix
  - Combines absolute and relative information

Implicit encoding of position

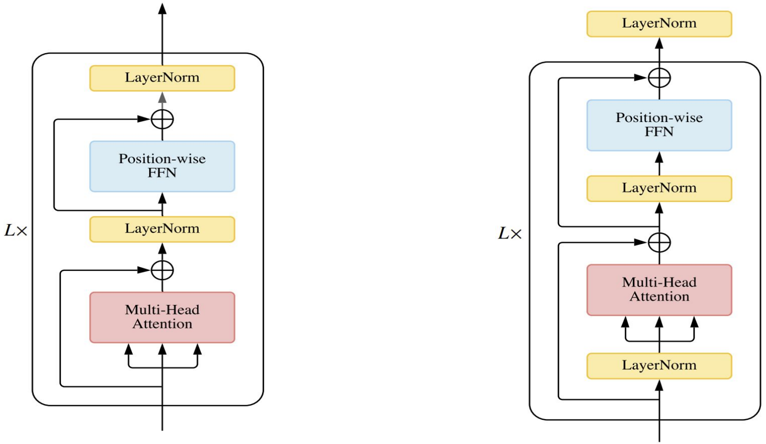
- Encode positional information in word embeddings
- Embedding function generalized to continuous function over positions
- R-Transformer
    - Model locality of sequential data with a local RNN
    - Inputs fed to local RNN and then to MHSA
- Conditional Positional Encoding:
    - Generate conditional position encodings with a 2-D convolution with zero-padding at each layer input
    - Convolution implicitly encodes position information

Position Representation on Decoder

- Decoders use cross attention and masks (to limit attention to previous inputs)
- Masked self-attention is not permutation equivariant
- A model that only uses the decoder is capable of sensing position information without explicit position representation
- Removing explicit position information on decoders improve performance

Layer Norm

- Layer Norm + Residual connection stabilize training of deep networks
- Where to apply Layer Norm?
  - Vanilla Transformer used Post-LN
  - Later transformers used Pre-LN



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

Layer Norm based variants

- Empirical results show post-LN without learning warm-up lead to unstable training
    - Because of large gradients near output layer at initialization
    - Pre-LN doesn’t suffer from this problem
- Post-LN outperforms pre-LN variants after convergence
- post-LN variants don’t suffer from gradient imbalance
    - Amplification effect is the cause
        - Heavier dependency on residual branch leads to larger output shift in post-LN transformers
        - Remedy: Initialize parameters based on activation variations of sample data

Layer Norm substitutes

- Layer Norm has learnable parameters which aren’t very useful and increase risk of overfitting
- AdaNorm - a normalization technique without learnable parameters
- Scaled L2 normalization
- PowerNorm
  - Modifies BatchNorm
    - Relaxes zero-mean normalization
    - Uses quadratic mean of the signal instead of variance
    - Uses running statistics of quadratic mean instead of using per-batch statistics

Normalization-free Transformers

- ReZero
  - Replaces LN module with a learnable residual connection
  - H’ = H + alpha * F(H)
  - alpha is a learnable parameter with zero-initialization
  - Better dynamic isometry for input signals and leads to faster convergence

Position-wise FFN

- Simply stacking self-attention modules causes a rank-collapse
  - Token-uniformity inductive bias
- FFN mitigates this issue
- Variants:
  - Replace ReLU with Swish function f(x) = x * sigmoid(beta * x)
  - GPT replaces ReLU with GELU
  - Mini-dalle replaces ReLU with GLU

Replacing FFN

- Idea: Replace FFN with similar structures with more parameters thereby improving model capacity
- Product-Key memory layers
    - A query network
    - A key selection module containing two sets of sub-keys
    - Value lookup table
    - Project input to a latency space using query network
    - Compare generated query to keys that are cartesian product of two sub-keys
    - Get K-nearest neighbors; Aggregate them to produce final output
    - Experiments on large-scale language modeling improved performance with negligible overhead

Replacing FFN

- Mixture-of-Experts(MoE)
    - Sparsely-gated MoE instead of FFN
        - Experts are several FFNs
        - Outputs: weighted sum of FFN outputs using gate values computed by routing function
        - Routing function is learnable. Assigns tokens to experts
        - Only the experts with top-k gate values are activated in forward pass
- Switch Transformer
    - Routes using only a single expert with the largest gate value
    - Auxillary loss function to load balance between experts
- Top-k routing with expert prototyping
    - Top-k prototype groups and top-1 routing within group
- Instead of learnable routing function, hash function can be used
    - Tokens are hashed to fixed number of buckets(experts)

Dropping FFN

- Observation: Replacing ReLU activation in FFN with softmax and dropping bias term turns FFN into an attention module where position-wise inputs attend to a global key-value memory
- Idea: Drop FFN by adding a set of learnable global key-value pairs
- In the decoder of a transformer, FFNs contribute very little with their large number of parameters.
    - Empirical results show improvement in inference and training with little loss of performance

Architecture-level variants

- Lite-Transformer
  - Replaces attention module with a two-branch structure
    - Attention to capture long-range contexts
    - Depth-wise convolution and linear layers to capture local dependencies
- Funnel Transformer
  - Length of the hidden sequence is gradually reduced using pooling along the sequence dimension and recovered using up-sampling
  - Reduces FLOPs and memory
- DeLighT Transformer
  - Replaces transformer block with three sub-modules
    - Expand-and-reduce - to learn wider representations with low computation requirements
    - Single-head self-attention to learn pairwise interaction
    - Reduce-and-expand FFN
  - Leads to a deeper network with fewer parameters and FLOPs

Cross-Block connectivity

- In a deep encoder-decoder transformer
  - Cross-attention module in the decoder only utilize the final outputs of the encoder
  - The error/feedback signal has to traverse along the depth of the encoder
  - Transformers are more susceptible to optimization issues like vanishing gradients
- Transparent Attention:
  - Weighted sum of encoder representations at all encoder layers
  - Use that as input in the cross attention module
  - This shortens the path for the error signal to the encoders and eases the optimization of deeper Transformers
- Feedback Transformer
  - Vanilla Transformer: Decoders were attending history representations of lower layers
  - Feedback mechanism where each position in the decoder can attend to history representation from all layers

Adaptive Computation Time

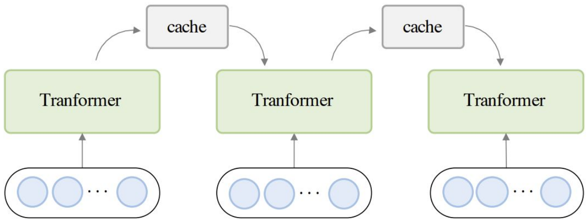
- Idea: Computation time conditioned on the inputs
- More computation for harder data and less computation for easy data
    - Efficiency for easy examples
    - Feature refinement for hard examples
- Universal Transformer
    - Recurrence-over-depth mechanism that iteratively refines representations for all symbols using a module that is shared over depth
    - Per-position dynamic halting mechanism that calculates a halting probability for each symbol at every time step
    - If halting prob > threshold, representation for subsequent time steps is unchanged
    - Recurrence is stopped when all symbols halt or max number of steps is reached

Adaptive Computation Time

- Conditional Computation Transformer
  - Adds a gating module to each self-attention and FFN to decide whether to skip the current layer
  - Auxillary loss that encourages the model to adjust the gating modules to match the practical computation cost to the available computation budget
- Adapt the number of layers to each input in order to achieve good speed-accuracy trade-off
  - Choosing to exit early
  - Add an internal classifier at each layer and jointly train all classifiers
  - Classifier output says yes or no to the exit question
  - Criteria:
    - Entropy of output probability distribution of current layer
    - Number of times predictions remain unchanged to decide to whether to exit
    - Window-based uncertainty criterion to achieve token-level partial exiting in sequence labeling tasks
    - Voting-based exit strategy that considers each layer predictions of all past internal classifiers to infer the correct label and to decide whether to exit

Divide and Conquer

- Decompose input sequence into finer segments
- Transformers are reused for different input segments
- Recurrent Transformers
  - A cache memory is maintained to incorporate history information
  - Cache is an additional input to the network
  - Network writes hidden state to cache
  - Example: Transformer-XL
- Compressive Transformer
  - Extends the cache with two levels of memory
    - Activations from previous segment
    - Compressed activations from older segments
- Memformer
  - Introduces cross attention to encoder
    - Allows encoder to attend to the memory
    - Memory slot attention on top of the encoder output to write the memory for next segment



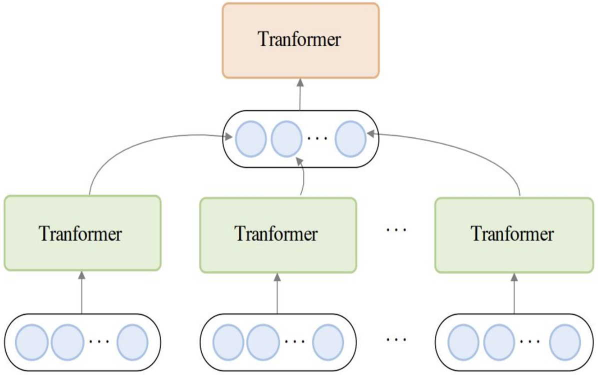
(a) Recurrent Transformer

Recurrent Transformer

- Extended context:
  - Transformer-XL: L x Nmem; Nmem is the length of cached memory sequence
  - Compressive Transformer: L x (Nmem + c Ncm); c is compression rate; Ncm is length of compressed memory
- Notes to avoid expensive BPTT:
  - Local loss functions where original memories are constructed from compressed memories
  - Memformer uses: Memory Replay Back-Propagation algorithm which replays the memory at each timestep to accomplish back propagation over long unrolls
- Alternate ways to add recurrence to transformers
  - ERNIE-Doc
  - Compressed single vector representation of divided segments

Hierarchical Transformer

- Decompose inputs hierarchically into elements of finer granularity
- Low-level features are first fed to a Transformer encoder
    - Outputs are aggregated to form a high level feature -> High-level transformer
- Allows modeling of long inputs with limited resources
- Potential to generate richer representations that are beneficial to tasks



(b) Hierarchical Transformer

Hierarchical Transformers

- Document translation
  - Previous sentences from both source and target are used in translating a new sentence
  - Attention is used to summarize low-level information
- HIBERT
  - Encodes a document of text by first learning sentence representations
  - Then uses sentence representations to encode document-level representations
- Hi-Transformer
  - Sentence Transformer
  - Document Transformer
  - Learns hierarchically document context aware sentence representations
  - Document context-aware sentence representations are fed to another sentence transformer to improve sentence context modeling

Hierarchical TF for richer representation

- Task specific rich representations
- TENER
  - Low-level TF encoder to encoder character features which is concatenated to word embeddings as input to high-level TF encoder
- Vision Transformer
  - Divide input image into several patches that serve as basic input elements to TF
  - Loses pixel information within patches
  - TNT: Fixes this issue by using inner TF block that transforms pixel representations and an outer TF block that takes fused vectors of patch representations and pixel representations as input

Alternate Architectures

- Macaron Transformer
  - Replaces each transformer block with FFN-attention-FFN variant
- Sandwich Transformer
  - Reorganizes attention modules and FFN modules such that attention modules are mainly located in lower layers and FFN modules in upper layers
  - Improves perplexity on multiple language models without increasing parameters
- Mask Attention Network
  - Prepends a dynamic mask attention module to self-attention module in each TF block
  - Mask is conditioned on token representations
  - Effectively models model locality in text data
- Evolved Transformer
  - Uses Evolution based architecture search to come up with a TF architecture
  - Shows consistent improved performance on language tasks
- DARTSformer
  - Uses Differentiable architecture search for searching architecture and outperforms ET

Pre-trained Transformer Variants

- Transformers are suitable for learning universal representations
- Transformers are used for producing pre-trained that can be later fine-tuned for downstream tasks
- Pre-training is usually combined with self-supervised tasks
- Encoder only
  - BERT with Masked Language Modeling and Next Sentence Prediction as self-supervision
  - RoBERTa: Removed NSP from BERT
- Decoder only
  - GPT, GPT-2, GPT-3
  - Impressive few shot performance
- Encoder-Decoder
  - BART - model is able to perform both natural language understanding and generation

References

- Survey of Transformers - https://arxiv.org/pdf/2106.04554.pdf