COMP [56]630– Machine Learning

Lecture 13 – Deep Learning (Transformers, GNNs)

Logistics

- Midterm on March 1, 2024
  - During class hours (available until Sunday for Distance section)
  - 50 minutes
  - 45 1-point questions
  - Similar to quizzes
- Assignment 2 will be posted
  - Due 03/11/2024 11:59 pm

Transformers

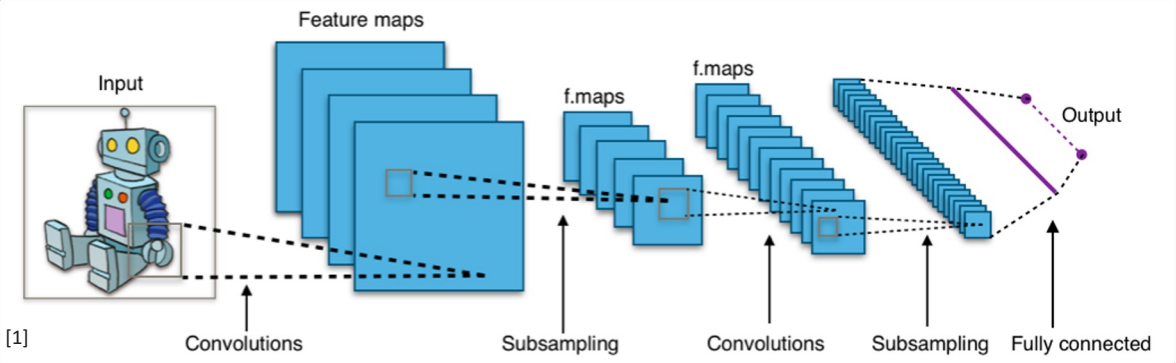
Adapted from Lucas Beyer

The classic landscape:

One architecture per "community"

Computer Vision

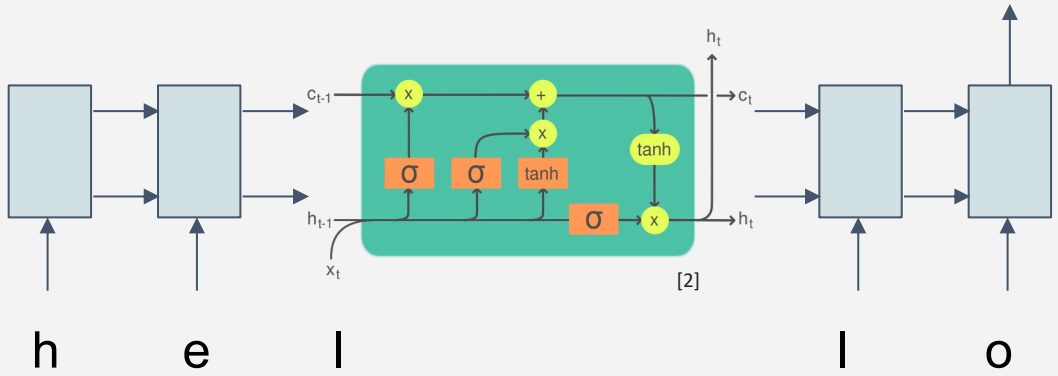
Convolutional NNs (+ResNets)



[1]

Natural Lang. Proc.

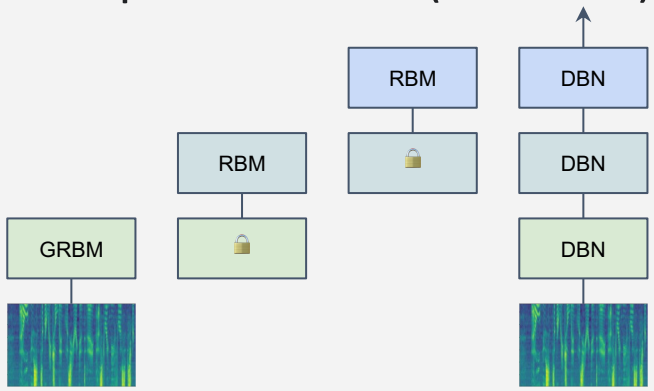
Recurrent NNs (+LSTMs)



h e l l o

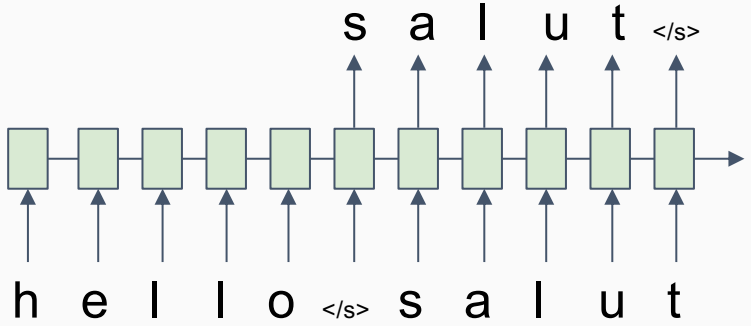
Speech

Deep Belief Nets (+non-DL)



Translation

Seq2Seq



RL

BC/GAIL

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \quad (17)$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta), \quad (18)$$
$$\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}]$$

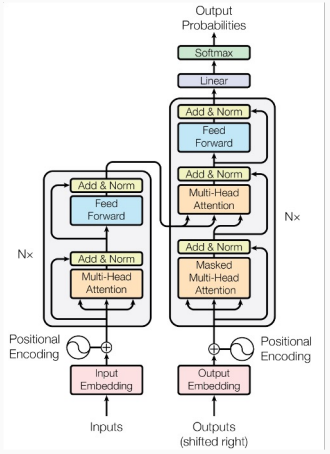
6: **end for**

[1] CNN image CC-BY-SA by Aphex34 for Wikipedia https://commons.wikimedia.org/wiki/File:Typical_cnn.png
[2] RNN image CC-BY-SA by GChe for Wikipedia https://commons.wikimedia.org/wiki/File:The_LSTM_Cell.svg

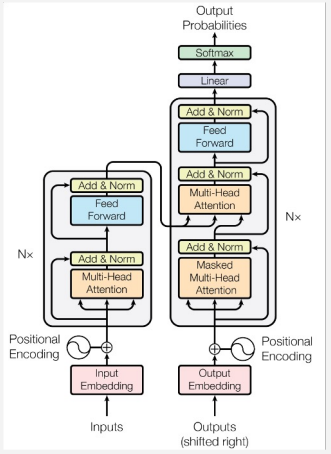
The Transformer's takeover:

One community at a time

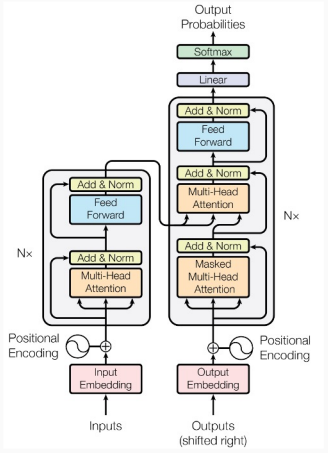
Computer Vision



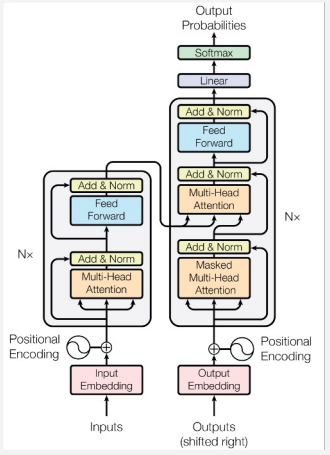
Natural Lang. Proc.



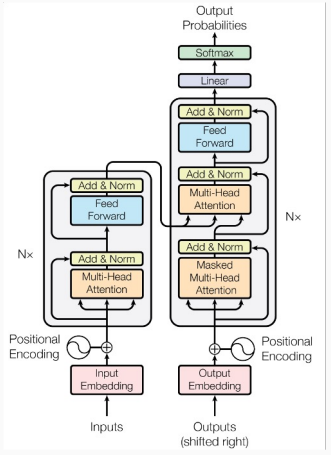
Reinf. Learning



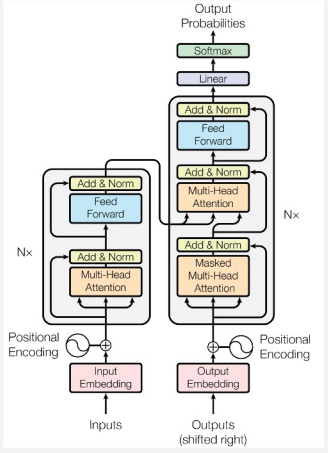
Speech



Translation



Graphs/Science



Transformer image source: "Attention Is All You Need" paper

The origins:

Translation, learned alignment

Neural Machine Translation by Jointly Learning to Align and Translate

2014, Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio

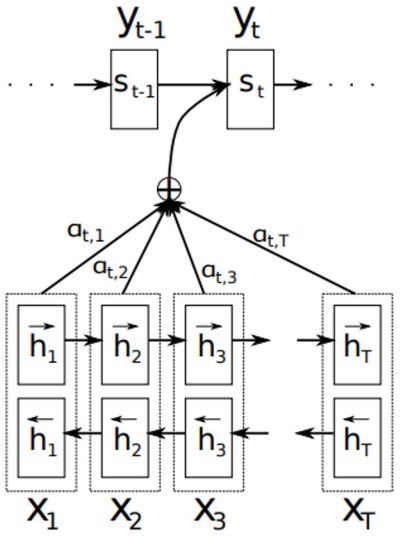
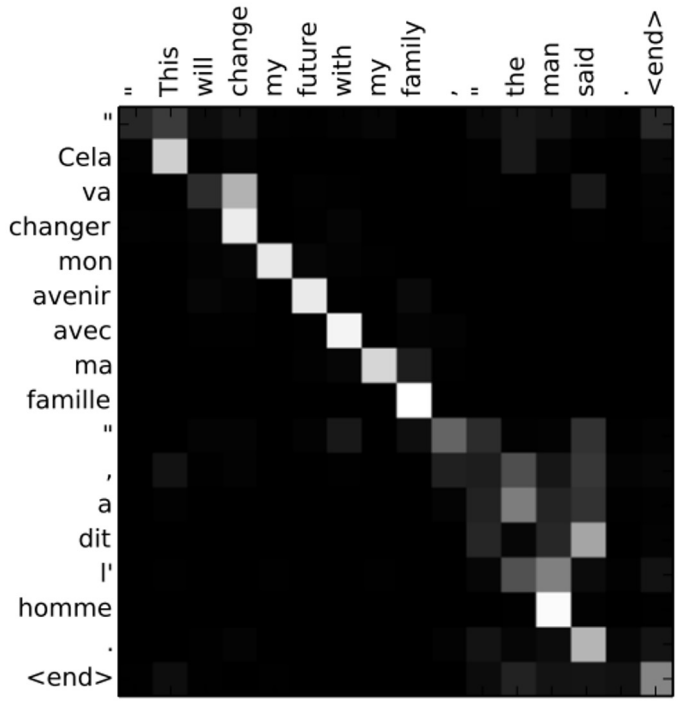


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



(d)

4 / 15

attention 1/3

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed-length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly.

Attention Is All You Need

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

Attention is a function similar to a "soft" kv dictionary lookup:

1. Attention weights $a_{1:N}$ are query-key similarities:

$$\hat{a}_i = q \cdot k_i$$

Normalized via softmax: $a_i = e^{\hat{a}i} / \sum_j e^{\hat{a}j}$

2. Output z is attention-weighted average of values $v_{1:N}$:

$$z = \sum_i \hat{a}_i v_i = \hat{a} \cdot v$$

3. Usually, k and v are derived from the same input x:

$$k = W_k \cdot x$$

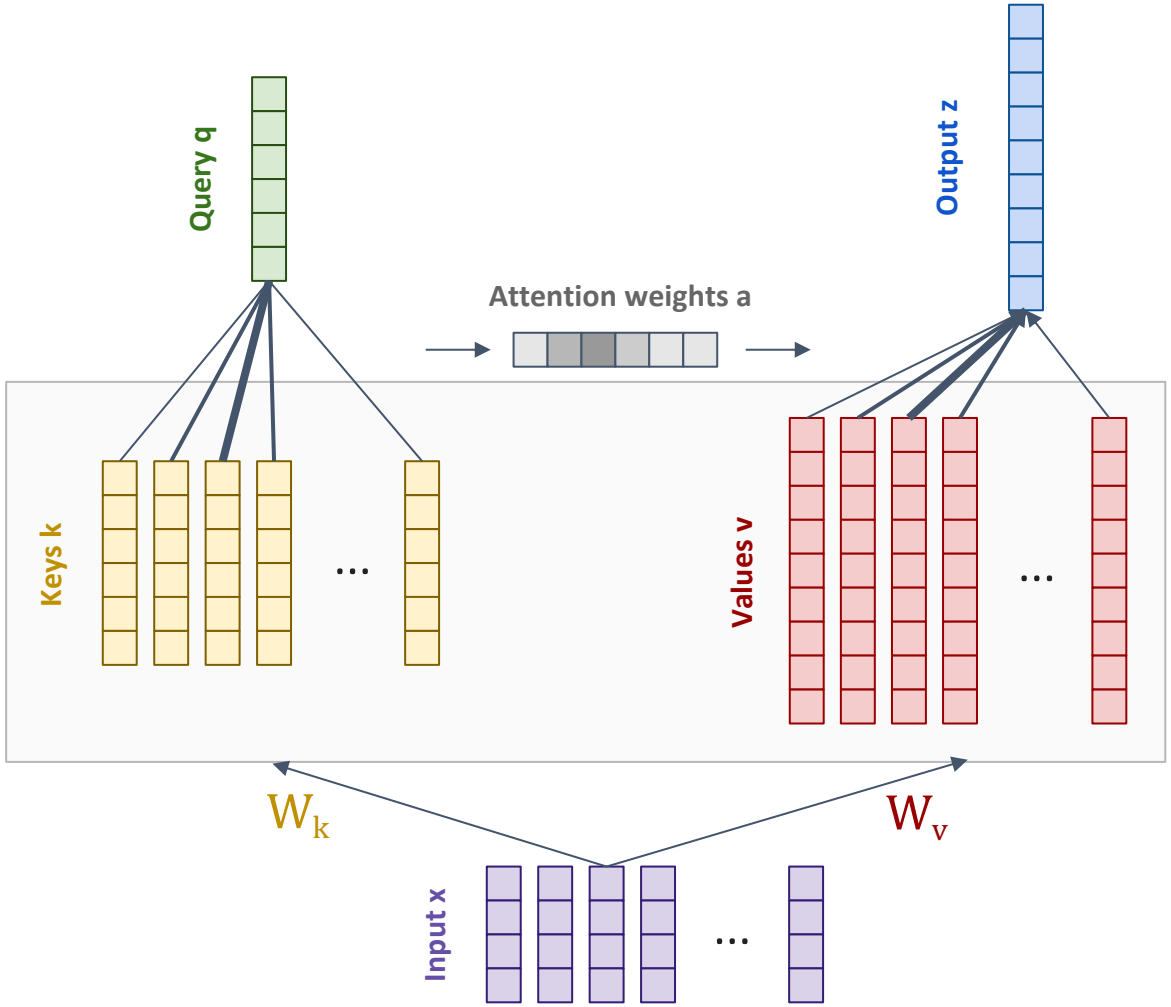
$$v = W_v \cdot x$$

The query q can come from a separate input y:

$$q = W_q \cdot y$$

Or from the same input x! Then we call it "self attention":

$$q = W_q \cdot x$$



Historical side-note: "non-local NNs" in computer vision and "relational NNs" in RL appeared almost at the same time and contain the same core idea!

Attention Is All You Need

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

But that's not actually it! There are a few more details:

1. We usually use **many queries** $q_{1:M}$, not just one.

   Stacking them leads to the Attention matrix $A_{1:N,1:M}$

   and subsequently to many outputs:

   $$z_{1:M} = \text{Attn}(q_{1:M},x) = [\text{Attn}(q_1,x) \mid \text{Attn}(q_2,x) \mid \ldots \mid \text{Attn}(q_M,x)]$$

2. We usually use "**multi-head**" attention. This means the

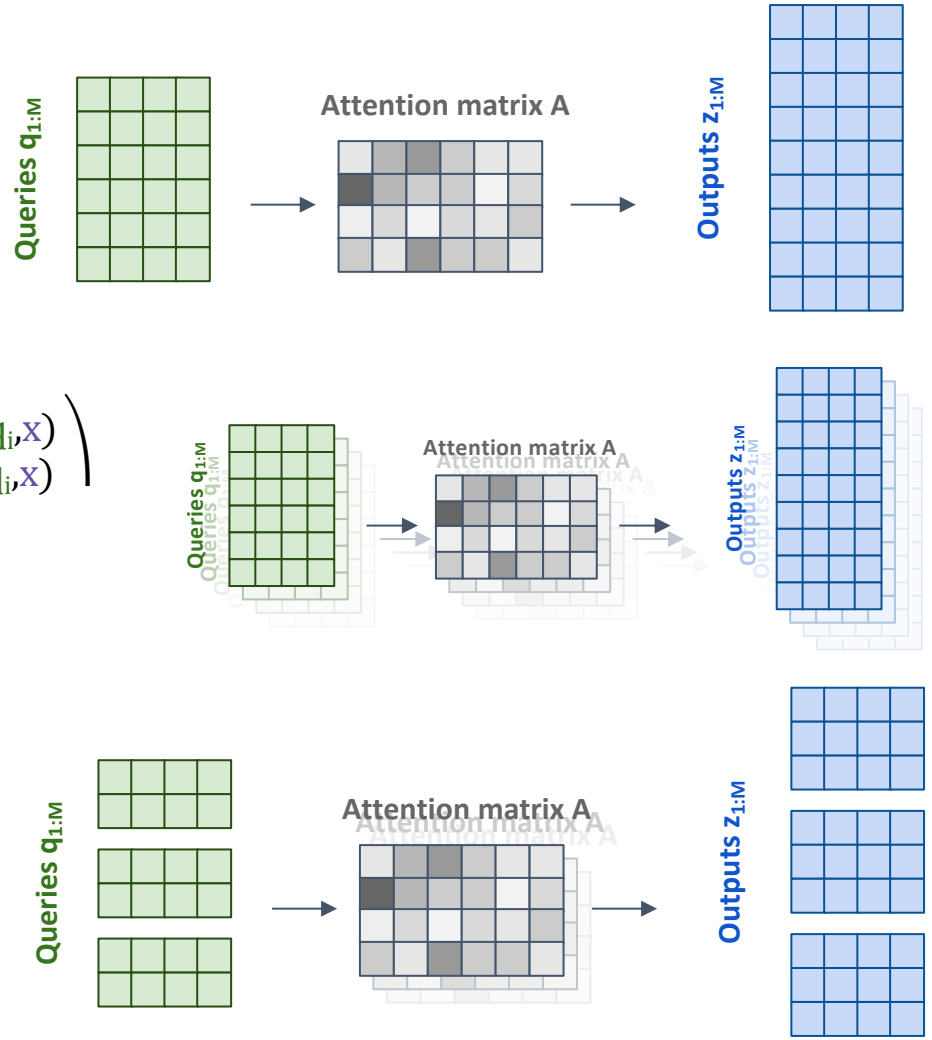
   operation is repeated K times and the results are concatenated along the feature dimension. **W**s differ.

   $$z_i = \begin{pmatrix} \text{Attn}_1(q_i,x) \\ \text{Attn}_2(q_i,x) \\ \ldots. \\ \text{Attn}_K(q_i,x) \end{pmatrix}$$

3. The most commonly seen formulation:

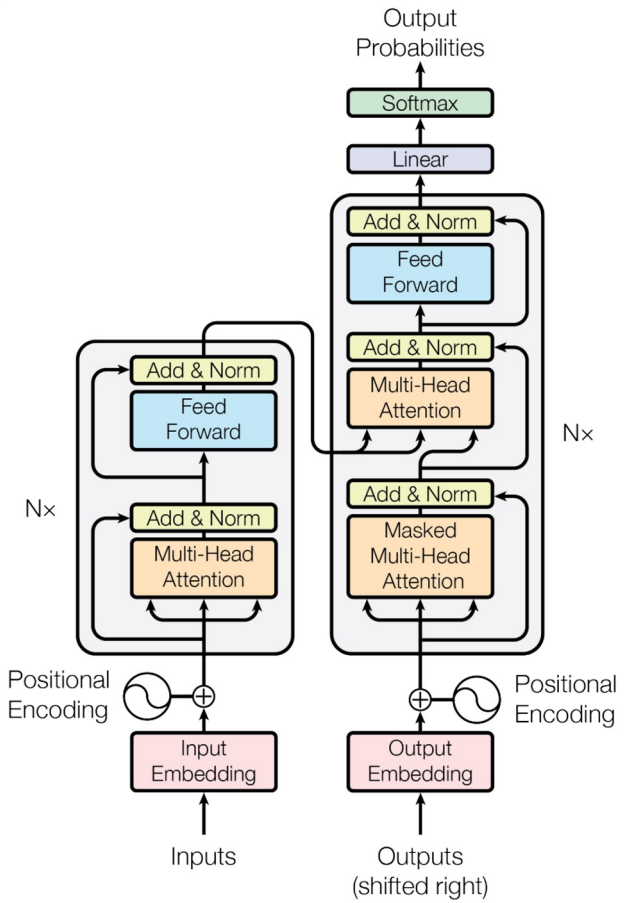
   $$z = \text{softmax}(QK'/\sqrt{d_{key}})V$$

   Note that the complexity is $O(N^2)$



Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

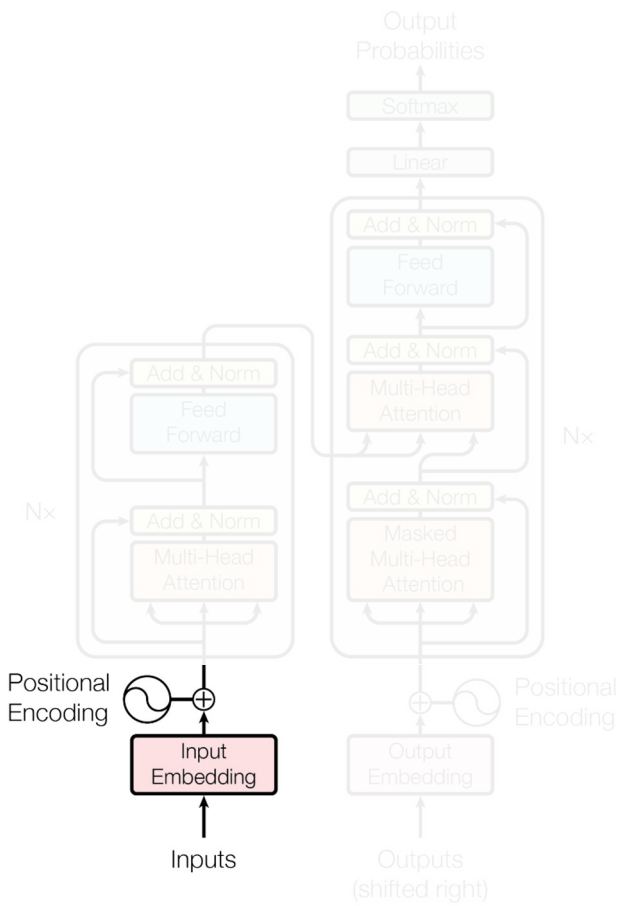


Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Input (Tokenization and) Embedding

Input text is first split into pieces. Can be characters, word, "tokens":

```
"The detective investigated" -> [The_] [detective_] [invest] [igat] [ed_]
```

Tokens are indices into the "vocabulary":

```
[The_] [detective_] [invest] [igat] [ed_] -> [3 721 68 1337 42]
```

Each vocab entry corresponds to a learned d_{model}-dimensional vector.

```
[3 721 68 1337 42] -> [ [0.123, -5.234, ...], [...], [...], [...], [...] ]
```

Positional Encoding

Remember attention is permutation invariant, but language is not!

Need to encode position of each word; just add something.

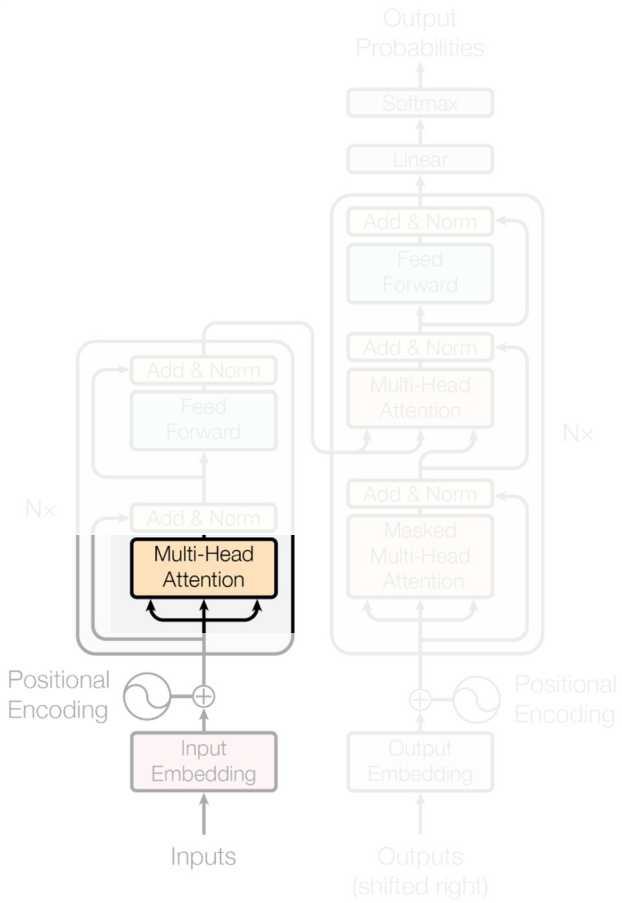
Think `[The_] + 10  [detective_] + 20  [invest] + 30` ... but smarter.

Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - The Transformer architecture

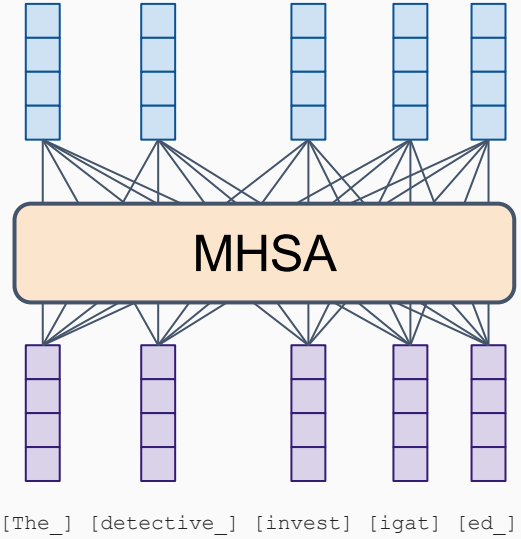
2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Multi-headed Self-Attention

Meaning the input sequence is used to create queries, keys, and values!

Each token can "look around" the whole input, and decide how to update its representation based on what it sees.

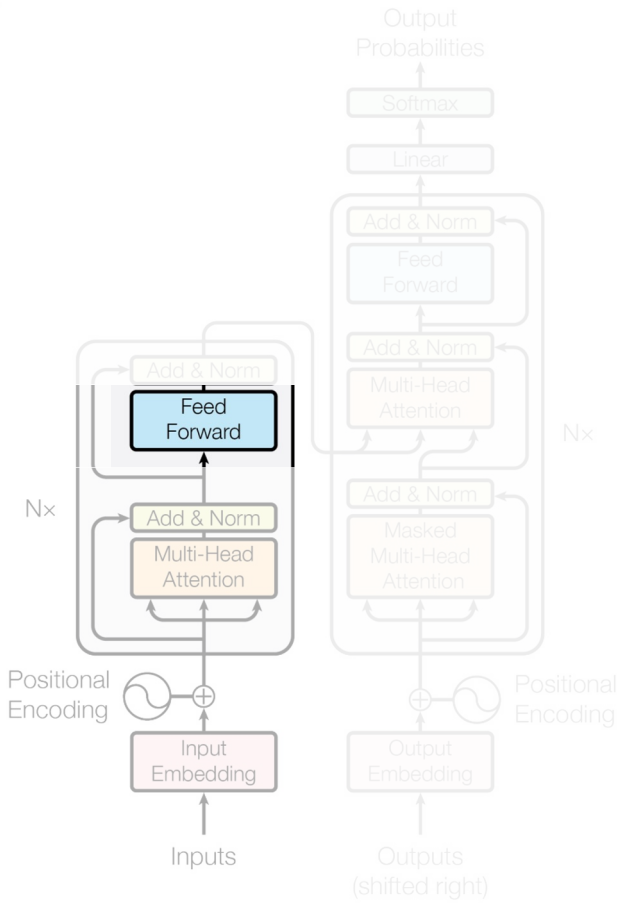


Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Point-wise MLP

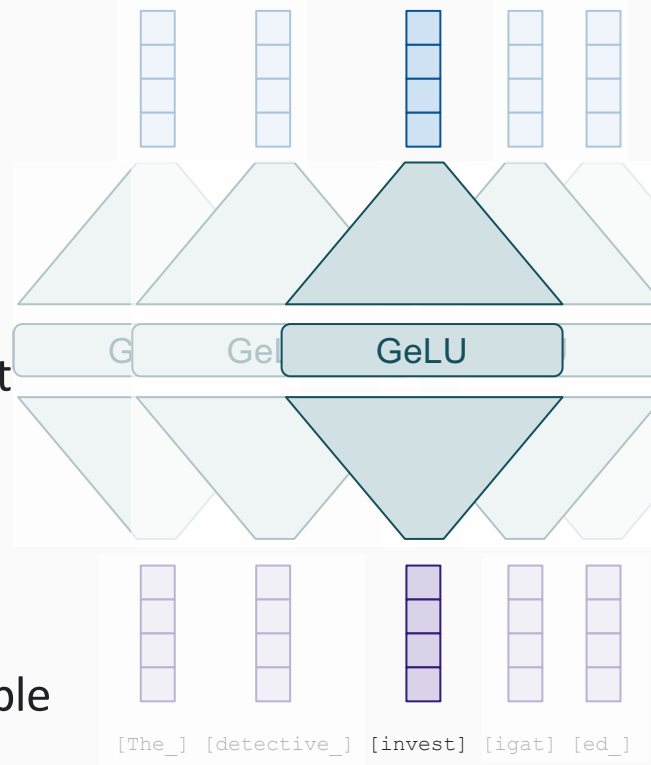
A simple MLP applied to each token individually:

$$z_i = W_2\,\mathrm{GeLU}(W_1 x + b_1) + b_2$$

Think of it as each token pondering for itself about what it has observed previously.
There's some weak evidence this is where "world knowledge" is stored, too.

It contains the bulk of the parameters. When people make giant models and sparse/moe, this is what becomes giant.

Some people like to call it 1x1 convolution.

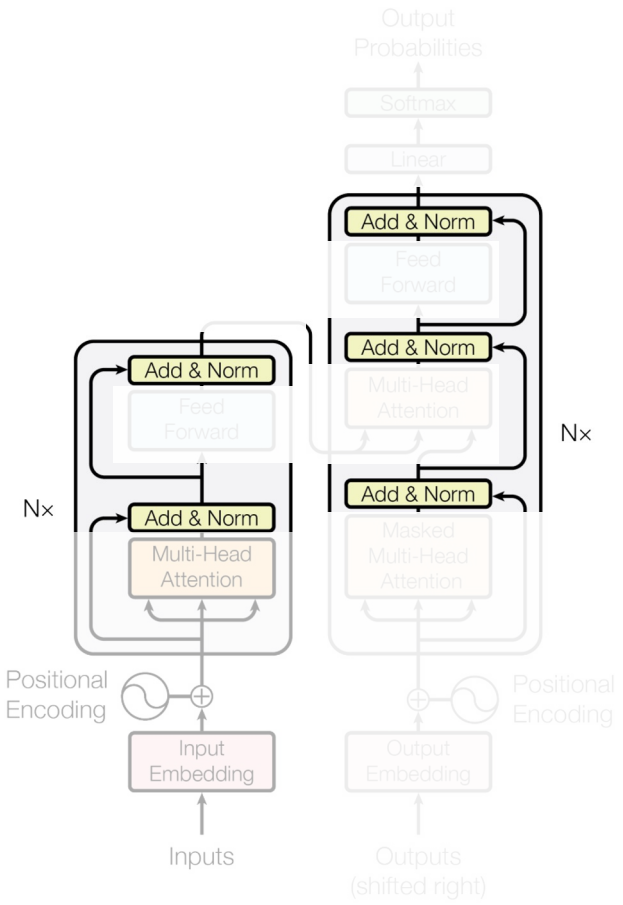


Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Residual connections

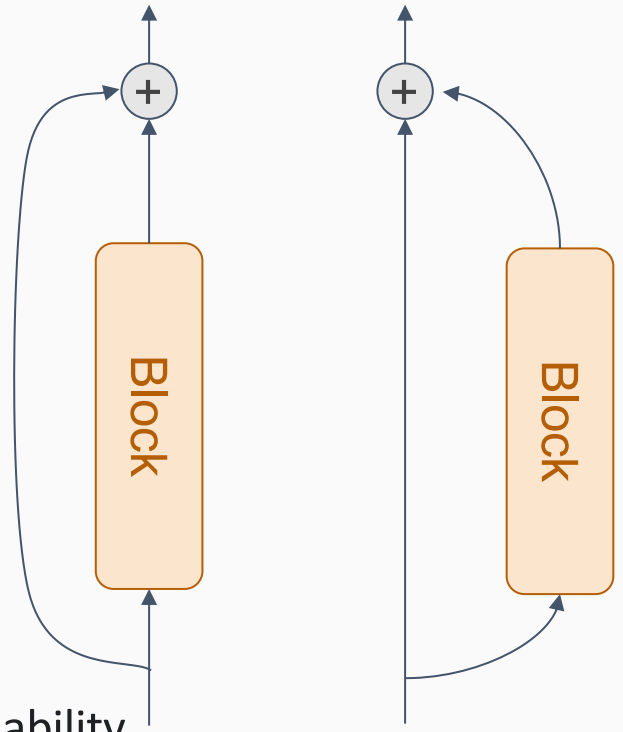
Each module's output has the exact same shape as its input.

Following ResNets, the module computes a "residual" instead of a new value:

$$z_i = \text{Module}(x_i) + x_i$$

This was shown to dramatically improve trainability.

"Skip connection" "Residual block"



LayerNorm

Normalization also dramatically improves trainability.

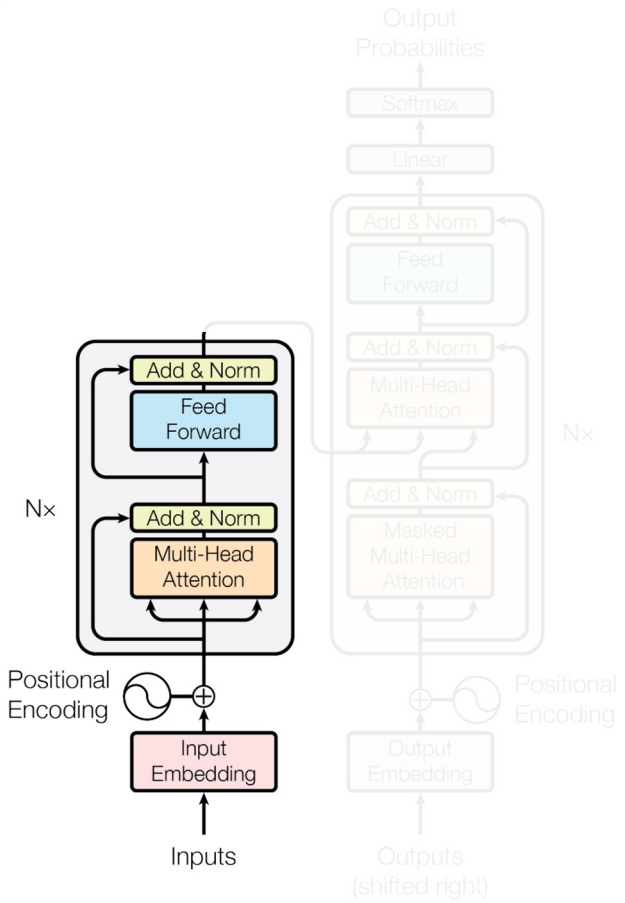
There's **post-norm** (original) and **pre-norm** (modern)

$$z_i = \text{LN}(\text{Module}(x_i) + x_i)$$

$$z_i = \text{Module}(\text{LN}(x_i)) + x_i$$

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Encoding / Encoder

Since input and output shapes are identical, we can stack N such blocks.

Typically, N=6 ("base"), N=12 ("large") or more.

Encoder output is a "heavily processed" (think: "high level, contextualized") version of the input tokens, i.e. a sequence.

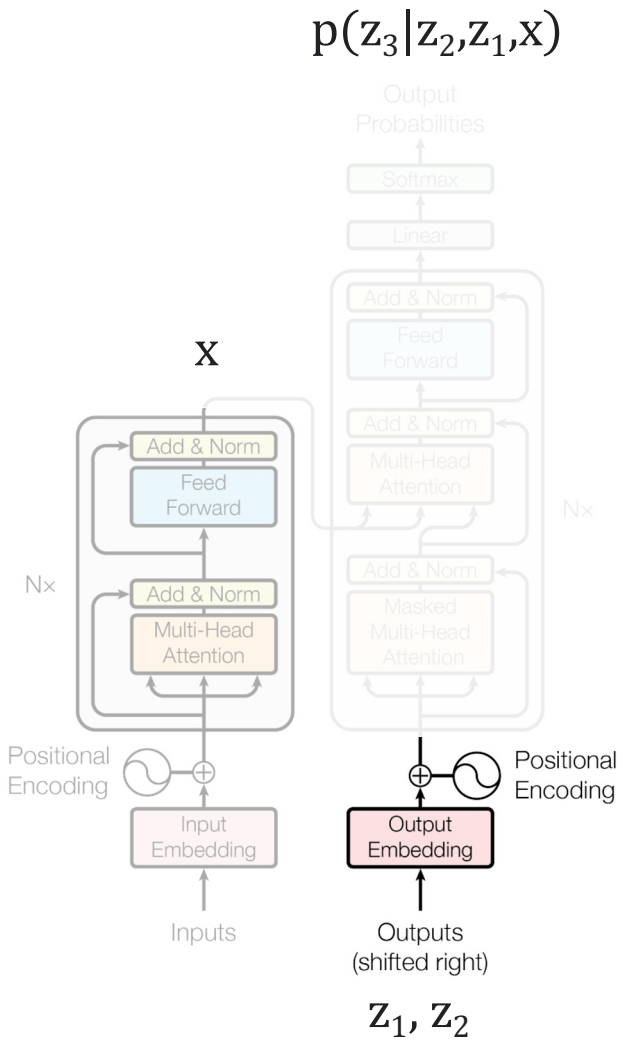
This has nothing to do with the requested output yet (think: translation). That comes with the decoder.

Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Decoding / the Decoder **(alternatively Generating / the Generator)**

What we want to model: $p(z|x)$

for example, in translation: $p(z \mid$ "the detective investigated") **∀z**

Seems impossible at first, but we can *exactly* decompose into tokens:

$$p(z|x) = p(z_1|x)\ p(z_2|z_1,x)\ p(z_3|z_2,z_1,x)\ldots$$

Meaning, we can generate the answer one token at a time.

Each p is a full pass through the model.

For generating $p(z_3|z_2,z_1,x)$:

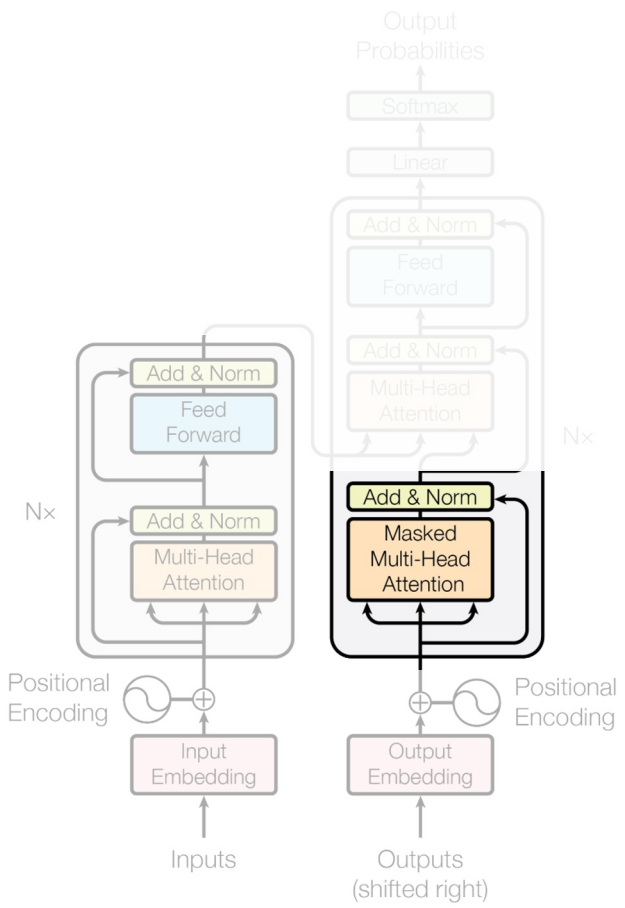
x comes from the encoder,

z_1, z_2 is what we have predicted so far, goes into the decoder.

Once we have $p(z|x)$ we still need to actually sample a sentence such as "le détective a enquêté". Many strategies: greedy, beam-search, ...

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Masked self-attention

This is regular self-attention as in the encoder, to process what's been decoded so far, but with a trick...

If we had to train on one single $p(z_3|z_2,z_1,x)$ at a time: SLOW!

Instead, train on all $p(z_i|z_{1:i},x)$ simultaneously.

How? In the attention weights for z_i, set all entries $i{:}N$ to 0.

This way, each token only sees the already generated ones.

At generation time

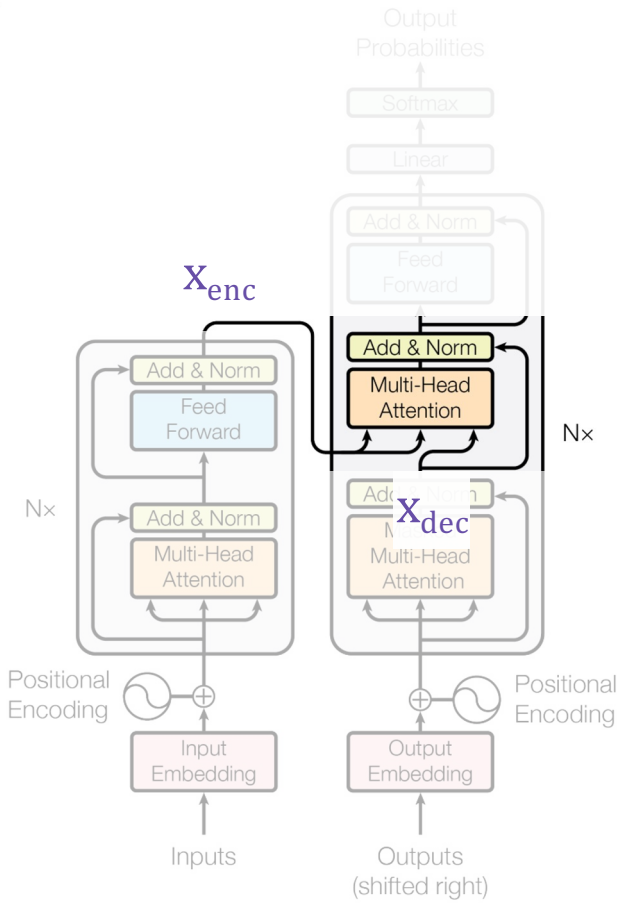
There is no such trick. We need to generate one z_i at a time. This is why autoregressive decoding is extremely slow.

Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



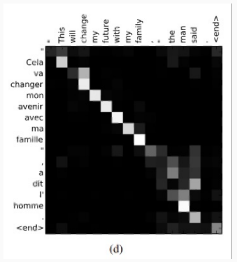
x_{enc}

x_{dec}

"Cross" attention

Each decoded token can "look at" the encoder's output:

$$\text{Attn}(q=W_q x_{dec},\ k=W_k x_{enc},\ v=W_v x_{enc})$$



This is the same as in the 2014 paper.
This is where $|x$ in $p(z_3|z_2,z_1,x)$ comes from.

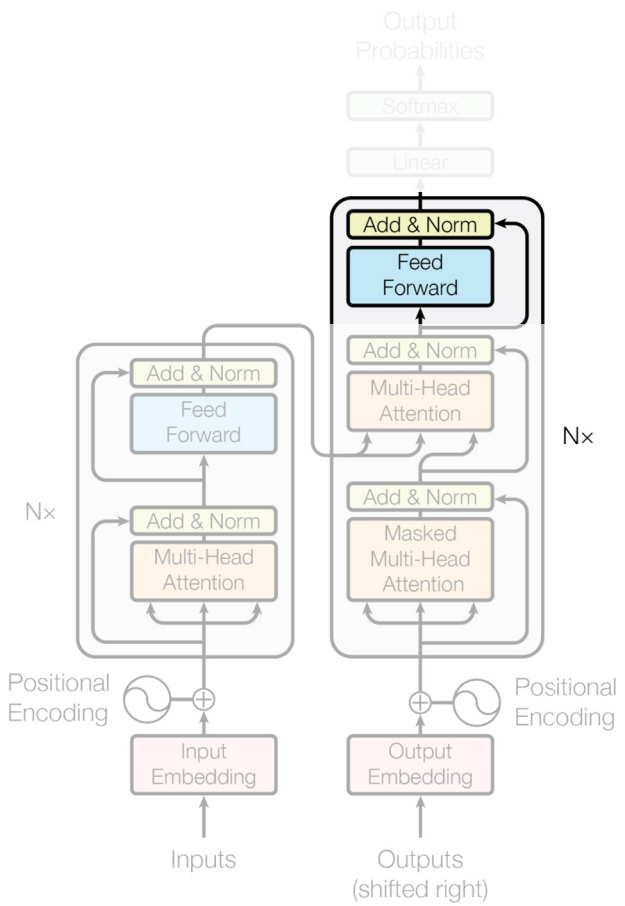
Because self-attention is so widely used, people have started just calling it "attention".
Hence, we now often need to explicitly call this "cross attention".

Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper ; heatmap image source: "Neural Machine Translation by Jointly Learning to Align and Translate" paper

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



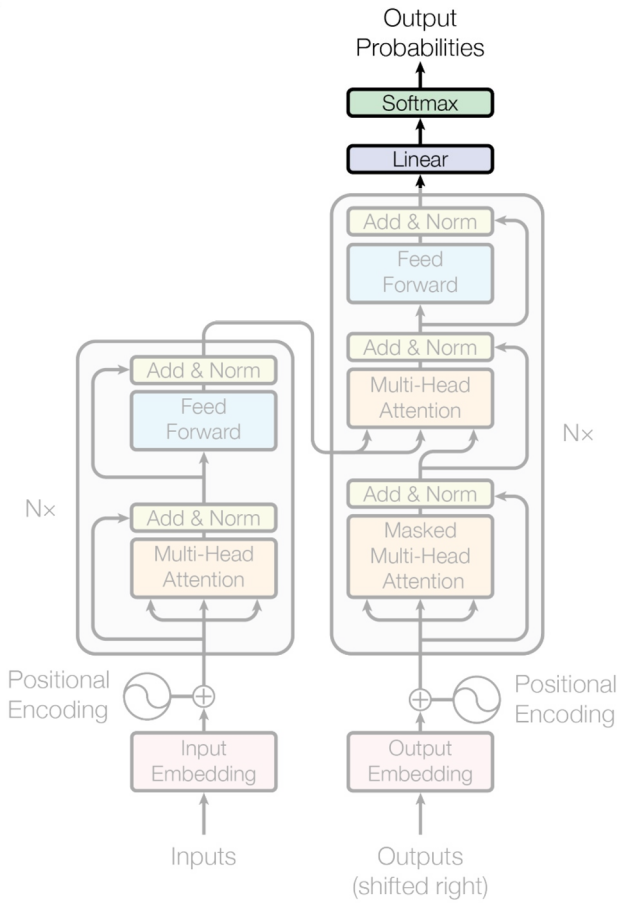
Feedforward and stack layers.

Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - The Transformer architecture

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin



Output layer

Assume we have already generated K tokens, generate the next one.

The decoder was used to gather all information necessary to predict a probability distribution for the next token (K), over the whole vocab.

Simple:

linear projection of token K

SoftMax normalization

Thanks to Basil Mustafa for slide inspiration

Transformer image source: "Attention Is All You Need" paper

Attention Is All You Need - Summary and results

2017, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Animation source: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
Table source: "Attention Is All You Need" paper

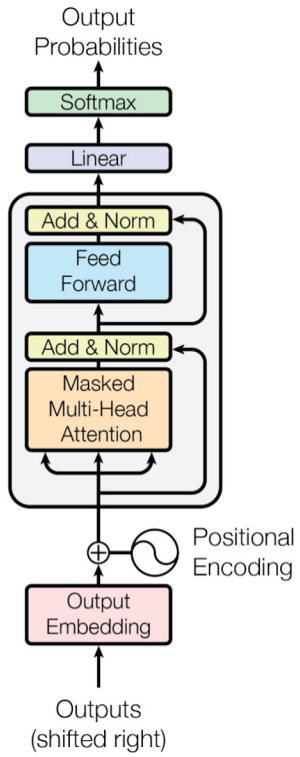
The first (1.5th) big takeover:

Language Modeling / NLP

Decoder-only

GPT

[sat_]

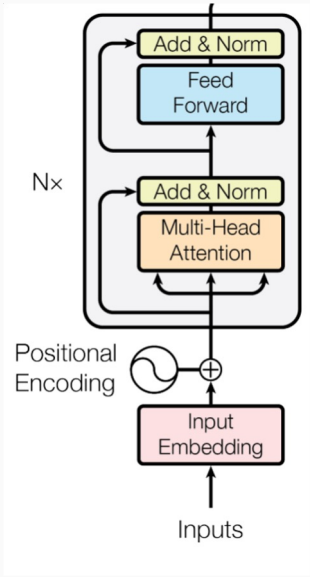


[START] [The_] [cat_]

Encoder-only

BERT

[*] [*] **[sat_]** [*] **[the_]** [*]



[The_] [cat_] **[MASK]** [on_] **[MASK]** [mat_]

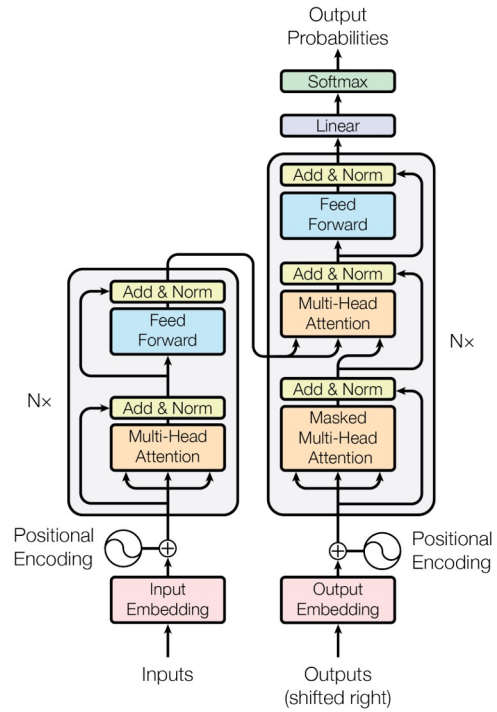
Enc-Dec

T5

Das ist gut.

A storm in Attala caused 6 victims.

This is not toxic.



Translate EN-DE: This is good.

Summarize: state authorities dispatched…

Is this toxic: You look beautiful today!

Transformer image source: "Attention Is All You Need" paper

The second big takeover:

Computer Vision

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

2020, A Dosovitskiy, L Beyer, A Kolesnikov, D Weissenborn, X Zhai, T Unterthiner, M Dehghani, M Minderer, G Heigold, S Gelly, J Uszkoreit, N Houlsby

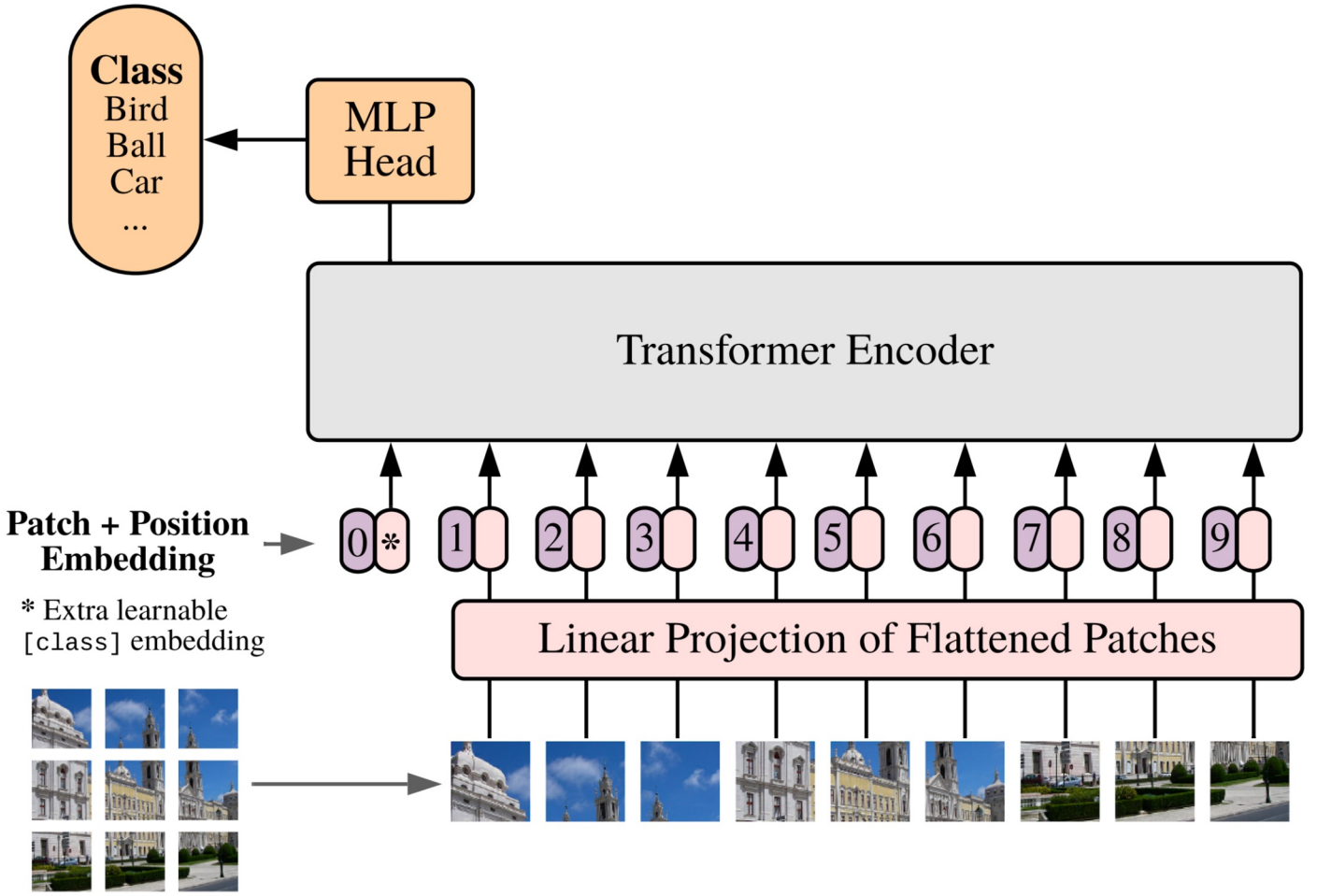
Vision Transformer (ViT)

Many prior works attempted to introduce self-attention at the pixel level.

For 224px², that's 50k sequence length, too much!

Thus, most works restrict attention to local pixel neighborhoods, or as high-level mechanism on top of detections.

The **key breakthrough** in using the full Transformer architecture, standalone, was to **"tokenize" the image** by **cutting it into patches** of 16px², and treating each patch as a token, e.g. embedding it into input space.



Side-note: MLP-Mixer

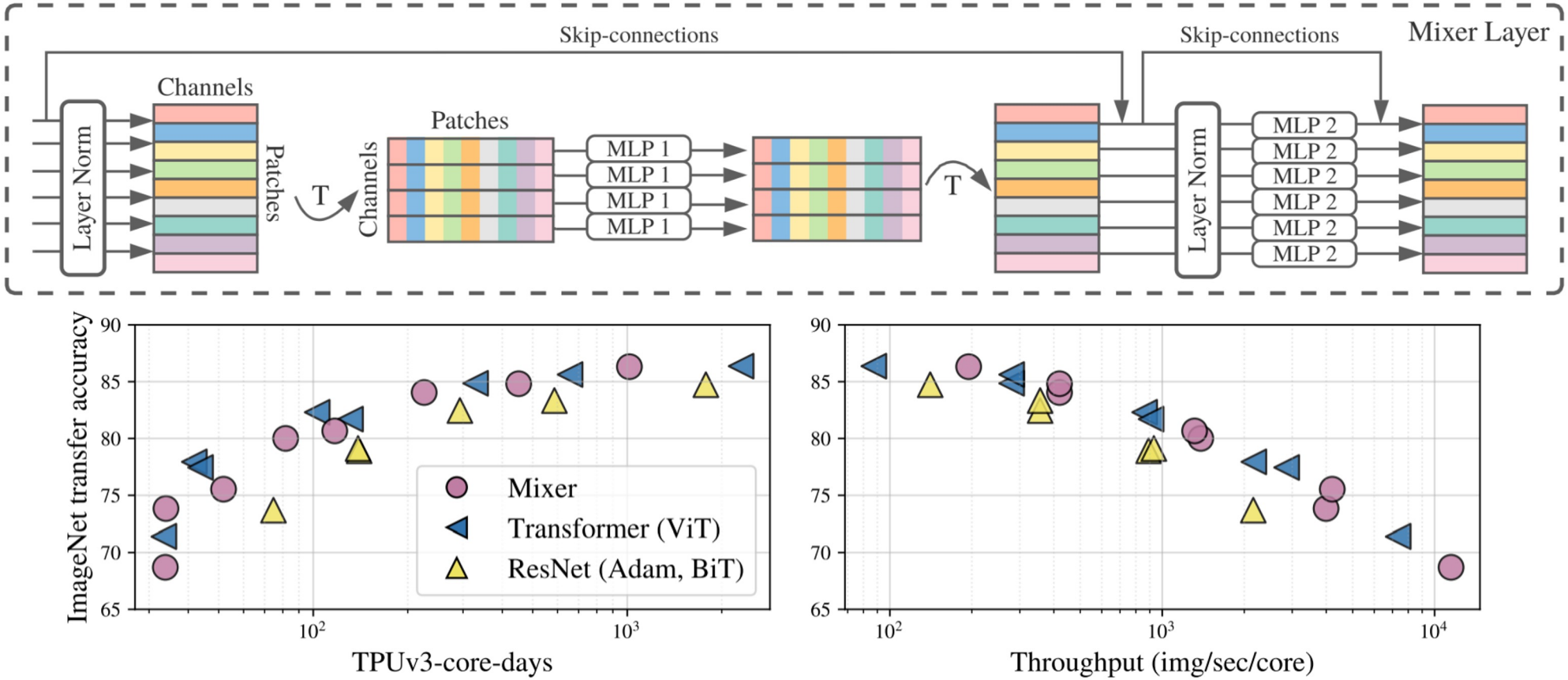
2020, I Tolstikhin, N Houlsby, A Kolesnikov, L Beyer, X Zhai, T Unterthiner, J Yung, A Steiner, D Keysers, J Uszkoreit, M Lucic, A Dosovitskiy

After ViT answered the question "Are convolutions really needed to process images?" with NO...

We wondered if self-attention is really needed?

The role of self-attention is to "mix" information across tokens.

Another simple way to achieve this, is to "transpose" tokens and run that through an MLP:



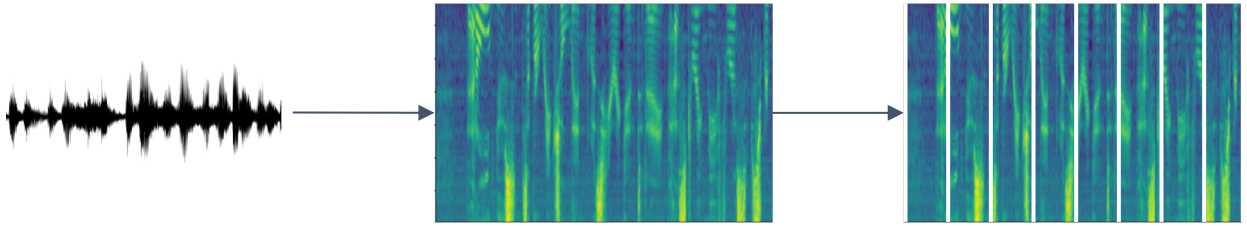
The third big takeover:

Speech

Conformer: Convolution-augmented Transformer for Speech Recognition

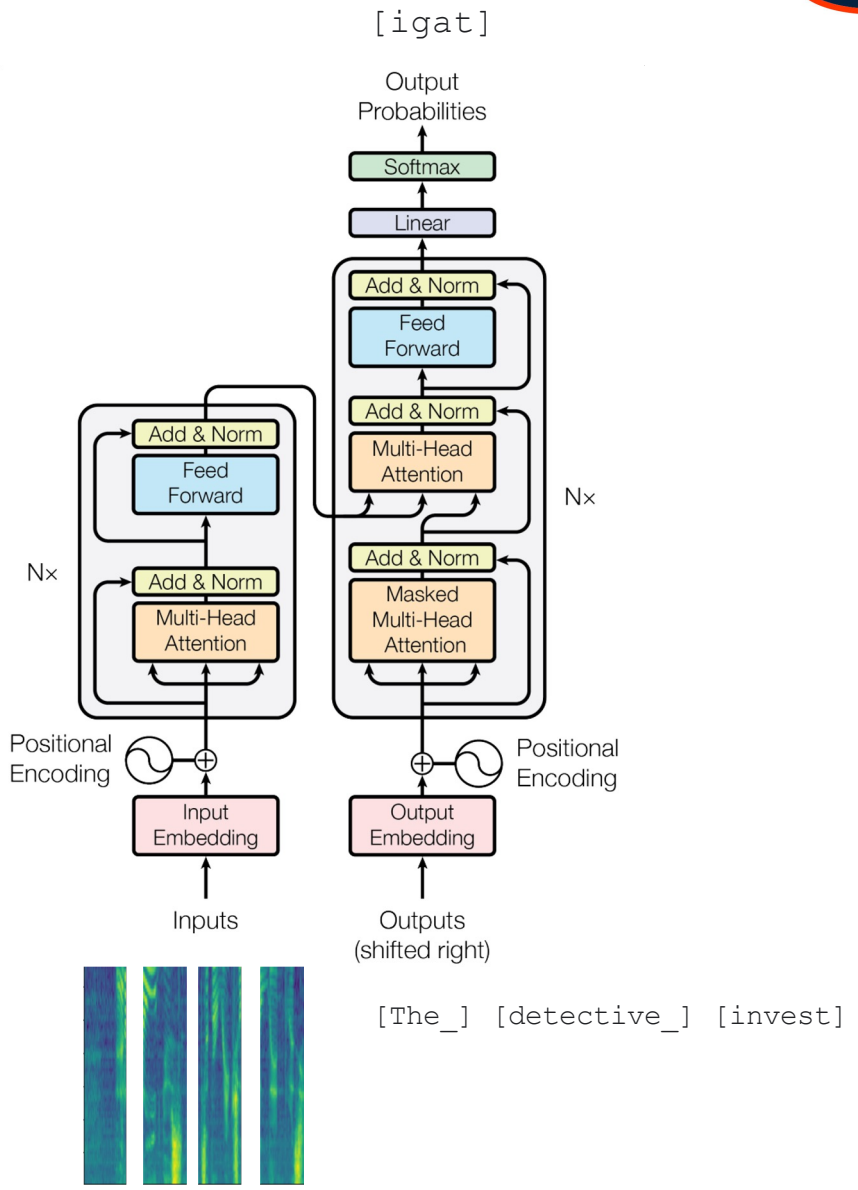
2020, A Gulati, J Qin, C-C Chiu, N Parmar, Y Zhang, J Yu, W Han, S Wang, Z Zhang, Y Wu, R Pang

Largely the same story as in computer vision.
But with spectrograms instead of images.



Add a third type of block using convolutions, and slightly reorder blocks, but overall very transformer-like.

Exists as encoder-decoder variant, or as encoder-only variant with CTC loss.



Transformer image source: "Attention Is All You Need" paper

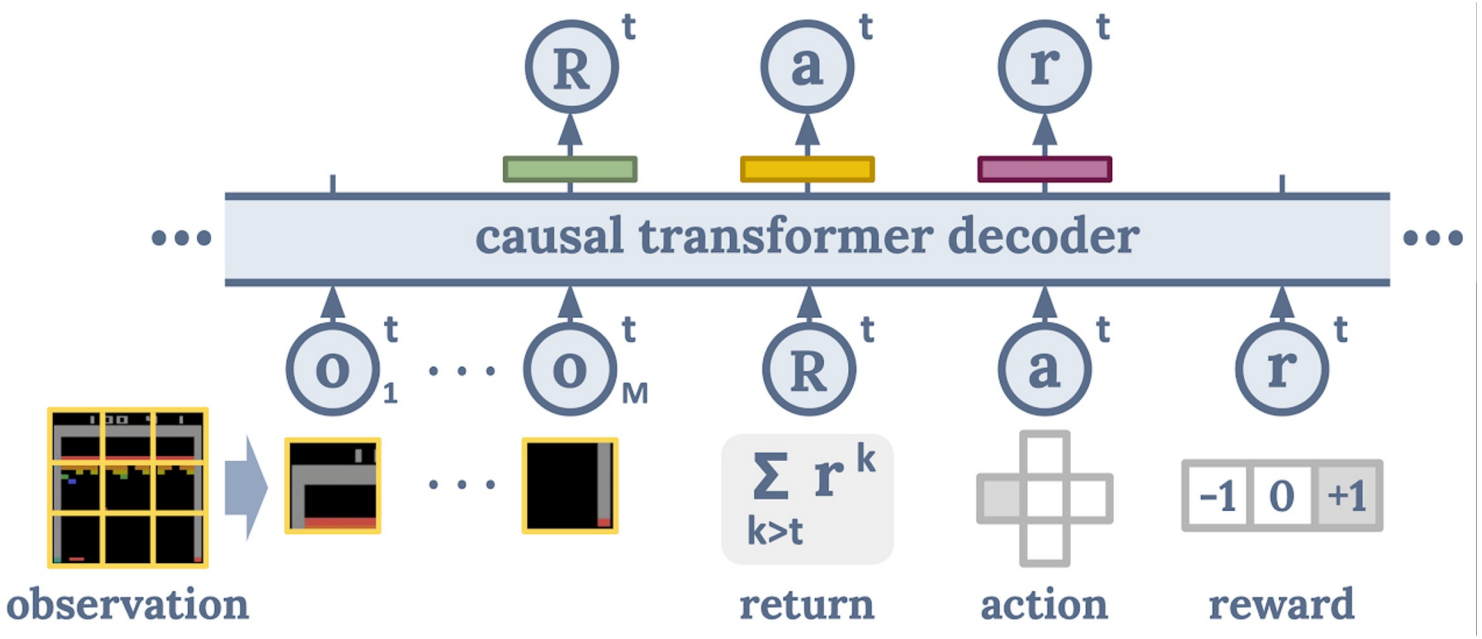
The fourth big takeover:

Reinforcement Learning

Decision Transformer: Reinforcement Learning via Sequence Modeling

2021, L Chen, K Lu, A Rajeswaran, K Lee, A Grover, M Laskin, P Abbeel, A Srinivas, I Mordatch

Cast the (supervised/offline) RL problem into a sequence ("language") modeling task:



Can generate/decode sequences of actions with desired return (eg skill)

The trick is prompting: `"The following is a trajectory of an expert player: [obs] ..."`

Slide credit: Igor Mordatch

The Transformer's

Unification of communities

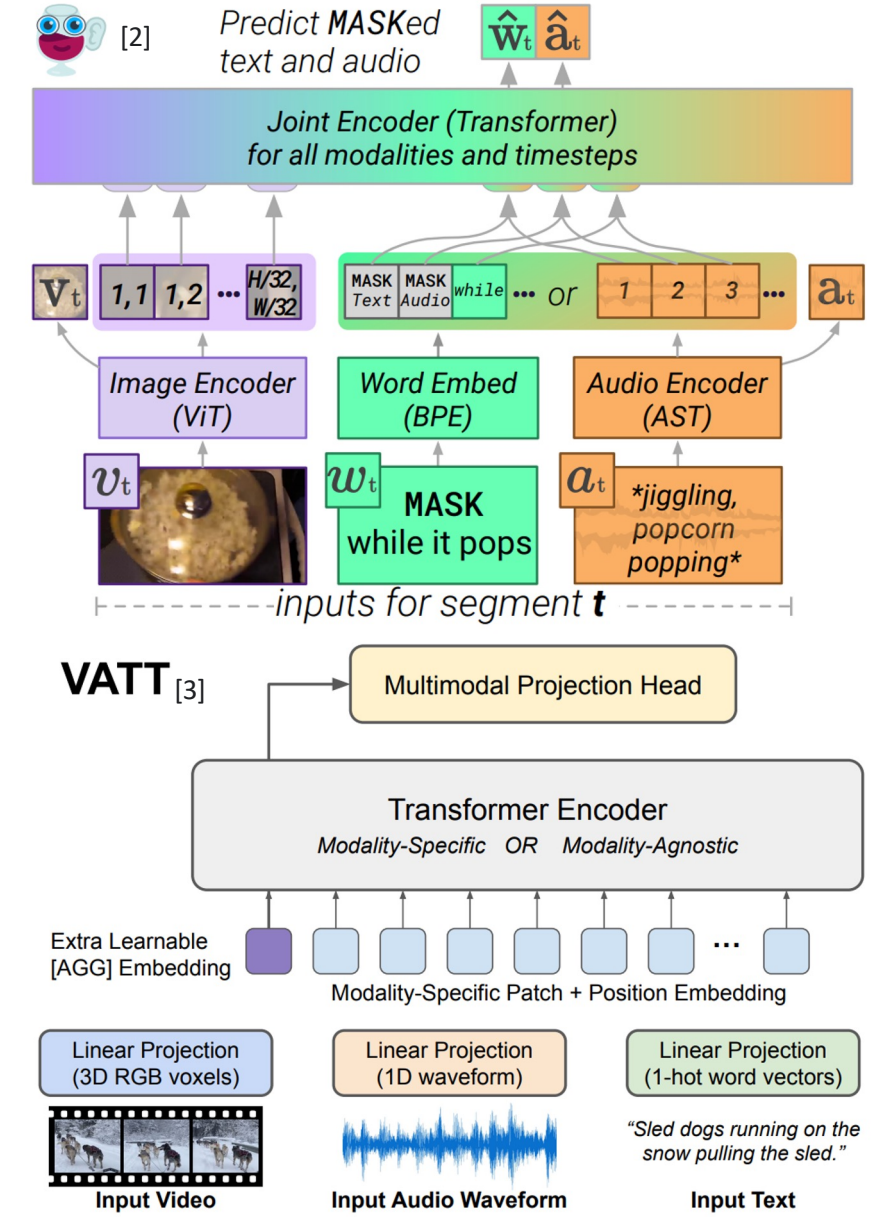
Anything you can tokenize, you can feed to Transformer

ca 2021 and onwards

Tokenize different modalities each in their own way (some kind of "patching"), and send them all jointly into a Transformer...

Seems to just work...

Currently an explosion of works doing this!



[1]

Images from:

[1] LIMoE by B Mustafa, C Riquelme, J Puigcerver, R Jenatton, N Houlsby

[2] MERLOT Reserve by R Zellers, J Lu, X Lu, Y Yu, Y Zhao, M Salehi, A Kusupati, J Hessel, A Farhadi, Y Choi

[3] VATT by H Akbari, L Yuan, R Qian, W-H Chuang, S-F Chang, Y Cui, B Gong

A note on

Efficient Transformers

A note on Efficient Transformers

The self-attention operation complexity is $O(N^2)$ for sequence length N.

We'd like to use large N:

- Whole articles or books
- Full video movies
- High resolution images

Many O(N) approximations to the full self-attention have been proposed in the past two years.

Unfortunately, none provides a clear improvement.
They always trade-off between speed and quality.

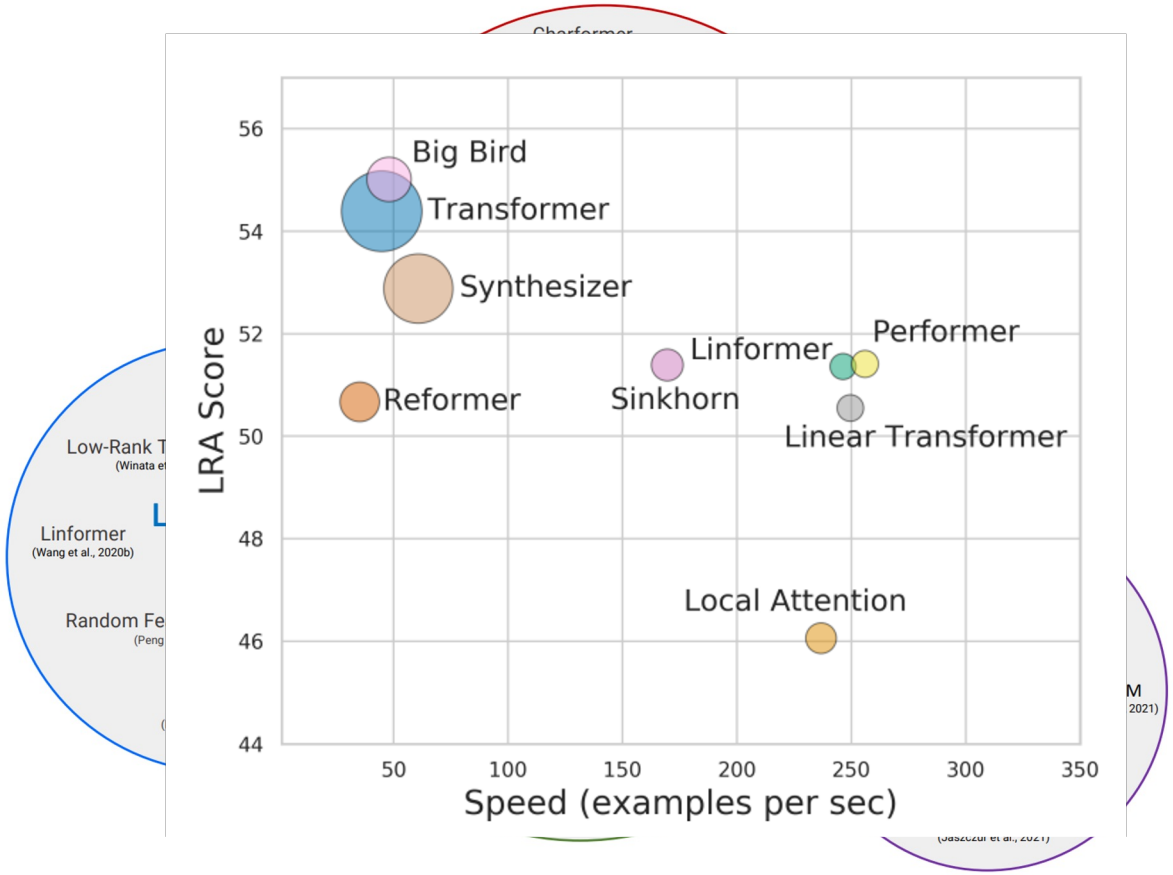


Figure 2: Taxonomy of Efficient Transformer Architectures.

Based on "Efficient Transformers: A Survey" by Yi Tay, Mostafa Dehghani, Dara Bahri, Donald Metzler
and "Long Range Arena: A Benchmark for Efficient Transformers" by Y Tay, M Dehghani, S Abnar, Y Shen, D Bahri, P Pham, J Rao, L Yang, S Ruder, D Metzler

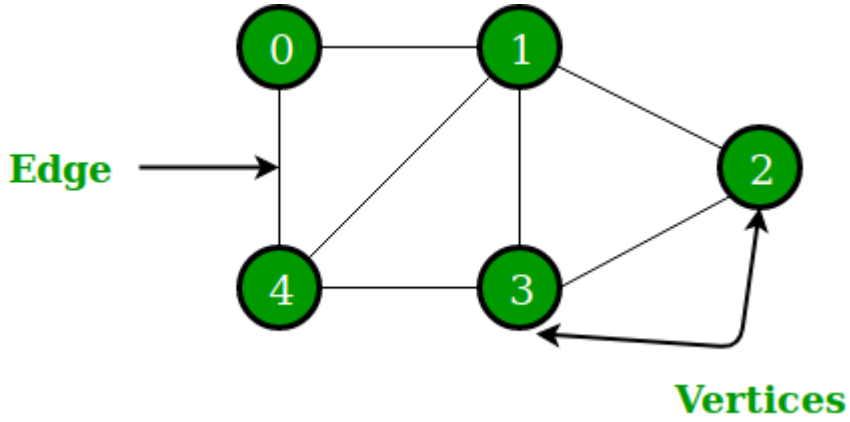
Graph Neural Networks

Slides by Sanjoy Kundu, Ph.D. Student, Oklahoma State University

Graphs: A very useful data structure

a graph data structure (V, E) **consists of**:

1. A **collection of vertices (V)** or nodes
2. A **collection of edges (E)** or paths; Edges can be directed and undirected!



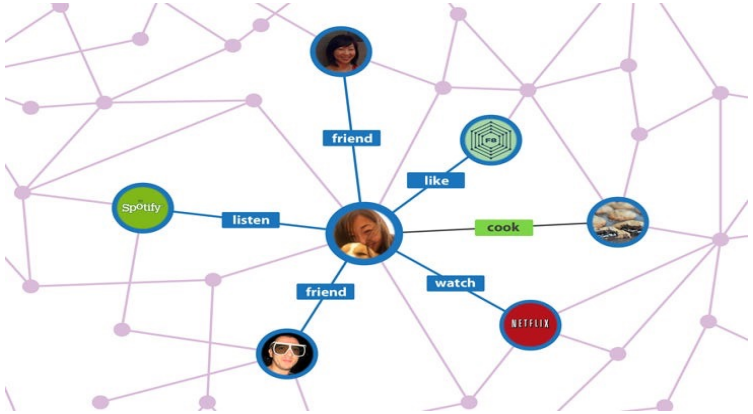
V = {0 , 1, 2, 3,4}
E = {(0, 1), (0, 4), (1, 2),
(1,3), (3,4), (1,4), (2,3)}
G = { V, E }

Illustration is taken from https://www.geeksforgeeks.org/

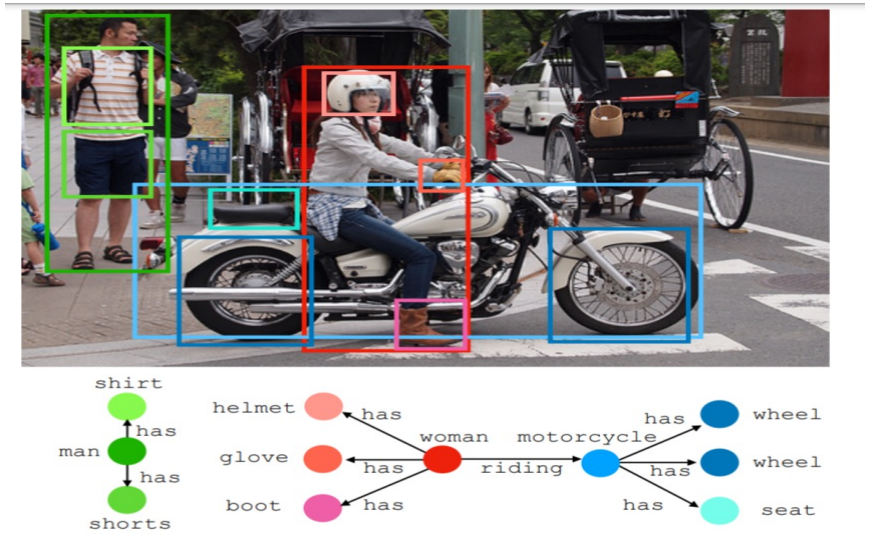
A real-World example of Graph

- The Internet is a vast, virtual graph
- Every vertex is an individual webpage, and every edge means that there is a hyperlink between two pages
- For an example Wikipedia or Facebook, have lots of hyperlinks. The pages are nodes and the links are edges

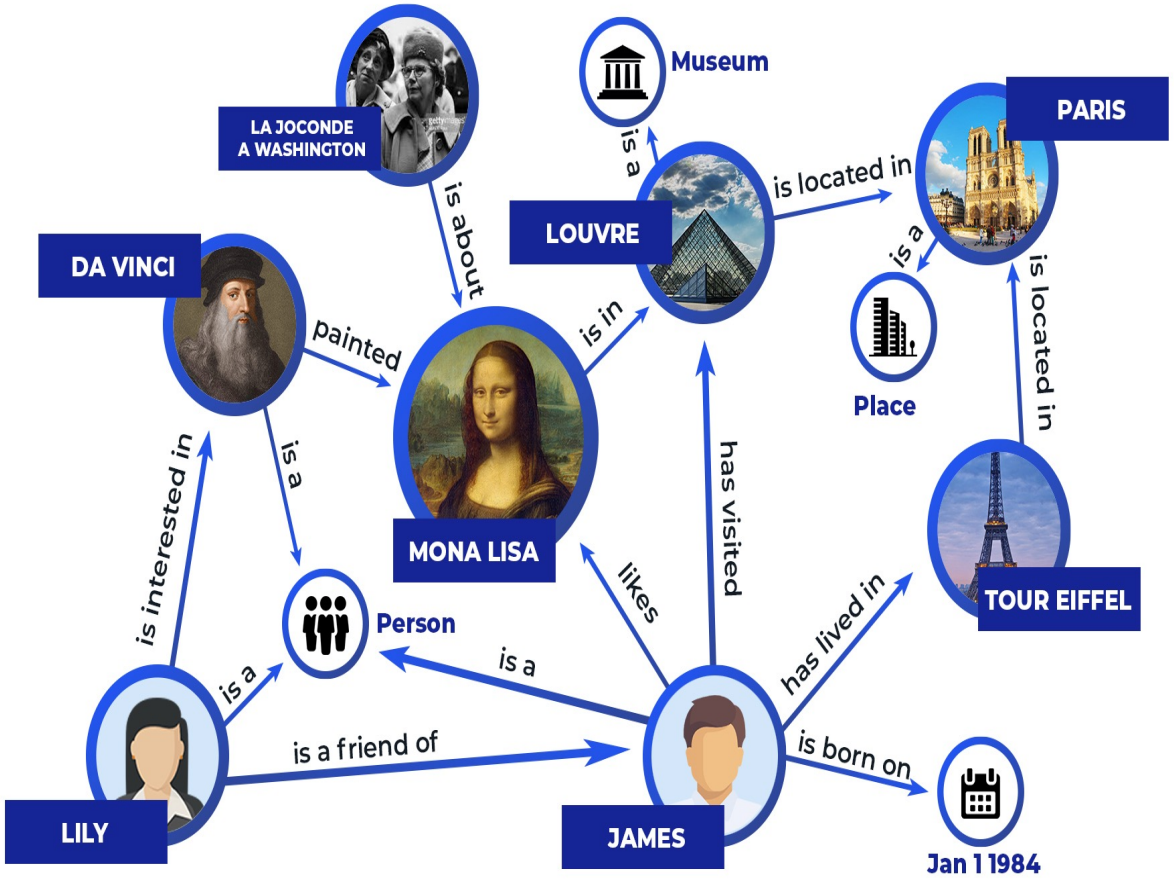
Other examples



Social Graphs



Scene Graphs



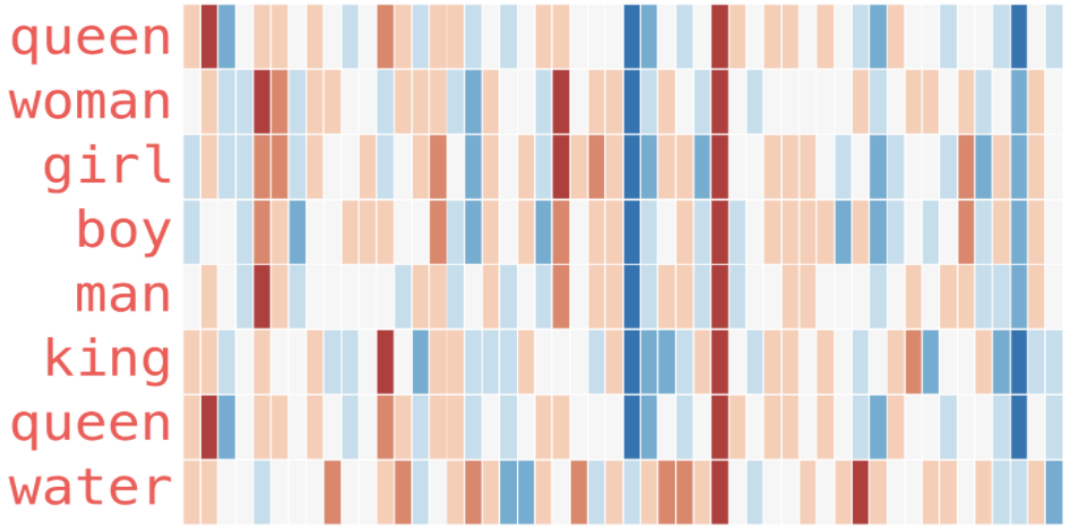
Knowledge Graph

Nodes and Edges: Embeddings

- Nodes and edges can be represented in some vector format, like word2vec
- BERT, GPT2, GPT3
- For visual graphs we can use visual features as embeddings of nodes

[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , 0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]

GloVe embedding for the work ‘king’



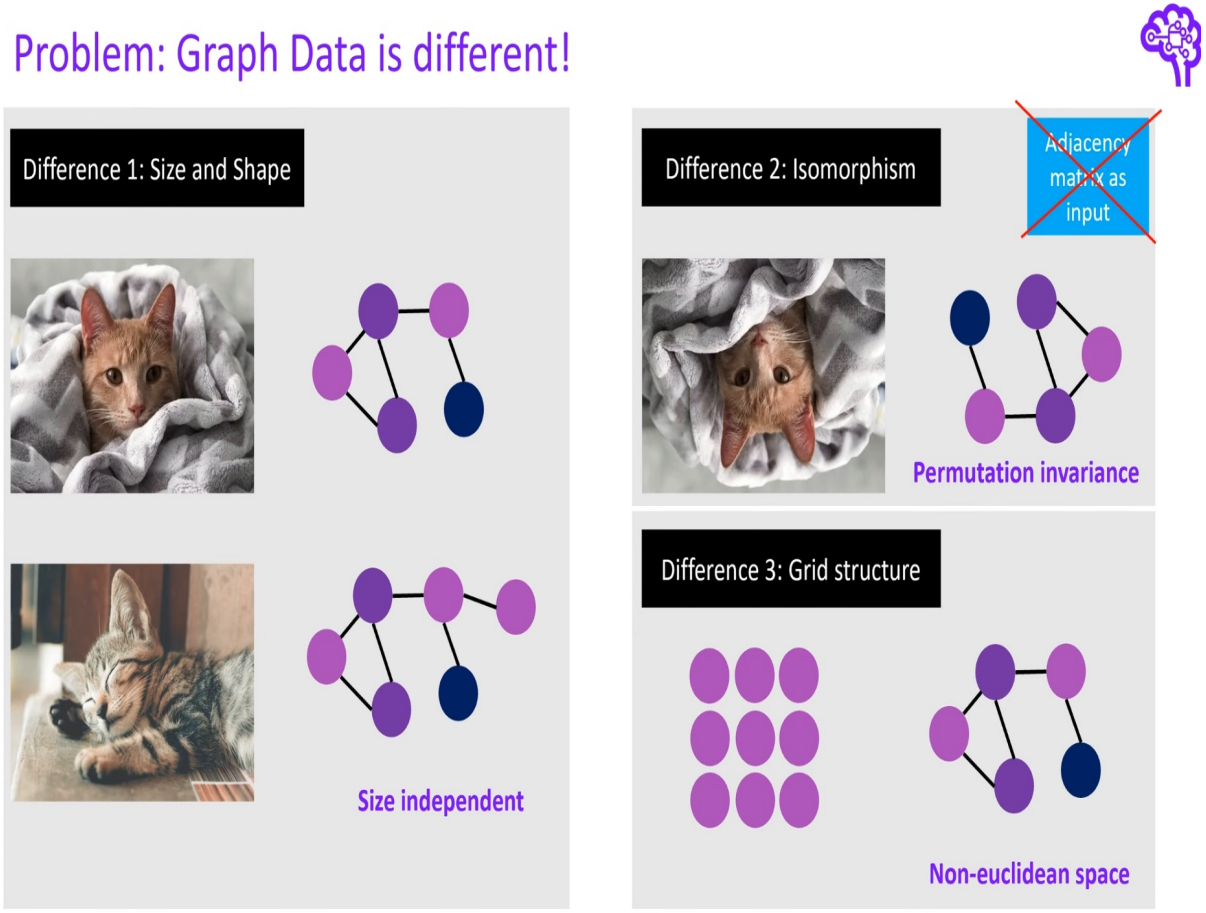
Visualization of word2vec,

The illustration is taken from https://jalammar.github.io/illustrated-word2vec/

Graphs Neural Network

Why do we need GNN?

- core assumption that instances are independent of each other.
- For graph data, each node is related to others by links of various types
- Conventional Machine Learning and Deep Learning tools are specialized in simple data types
- Images with the same structure and size, which we can think of as fixed-size grid graphs
- Text and speech are sequences, so we can think of them as line graphs
- But there are more complex graphs, without a fixed form, with a variable size of unordered nodes, where nodes can have different amounts of neighbors



Graphs Neural Network

- GNNs are deep learning methods designed to perform inference on graph data
- GNNs are neural networks that can be directly applied to graphs, and provide an easy way to do node-level, edge-level, and graph-level prediction tasks.
- GNNs can do what Convolutional Neural Networks (CNNs) failed to do

due to arbitrary size of graphs, unfixed ordering of nodes etc.

Challenges of graph Neural network

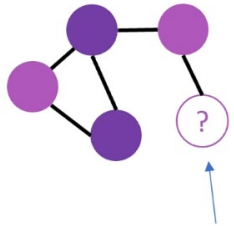
- Dynamic nature – Since GNNs are dynamic graphs, and it can be a challenge to deal with graphs with dynamic structures.
- Scalability – Applying embedding methods in social networks or recommendation systems can be computationally complex for all graph embedding algorithms, including GNNs.

What can we do with GNN?

- Graph Prediction
- Node Prediction
- Edge/link Prediction
- Graph visualization
- Graph clustering

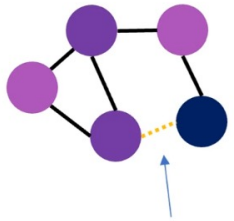
Examples for Machine Learning Problems with Graph Data

Node-level predictions



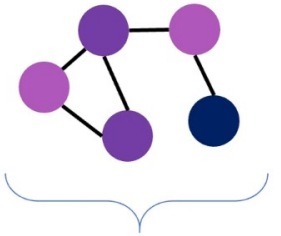
Does this person smoke?
(unlabeled node)

Edge-level predictions (Link prediction)



Next Netflix video?

Graph-level predictions



Is this molecule a suitable drug?

Basic Structure of GNN

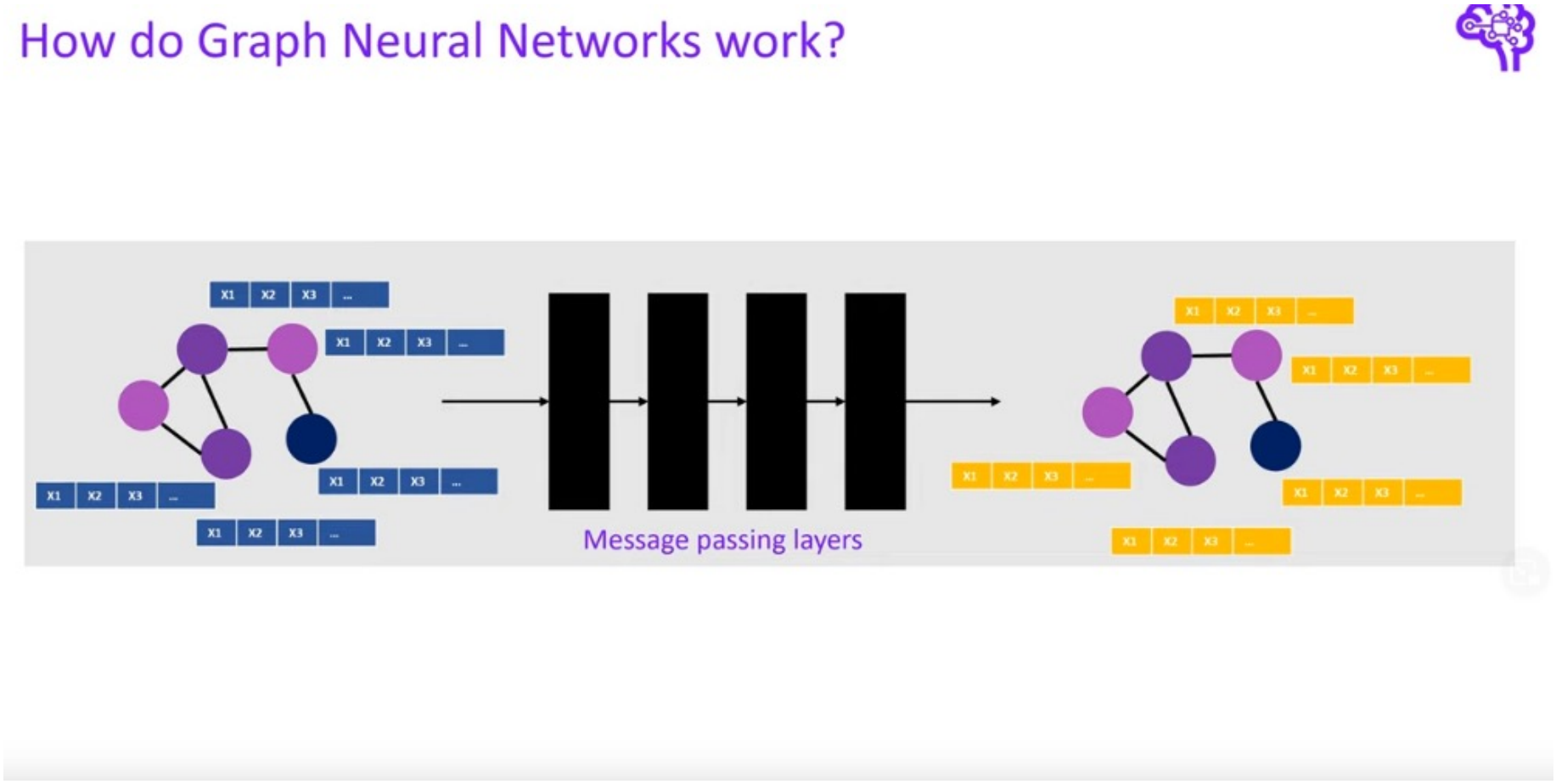
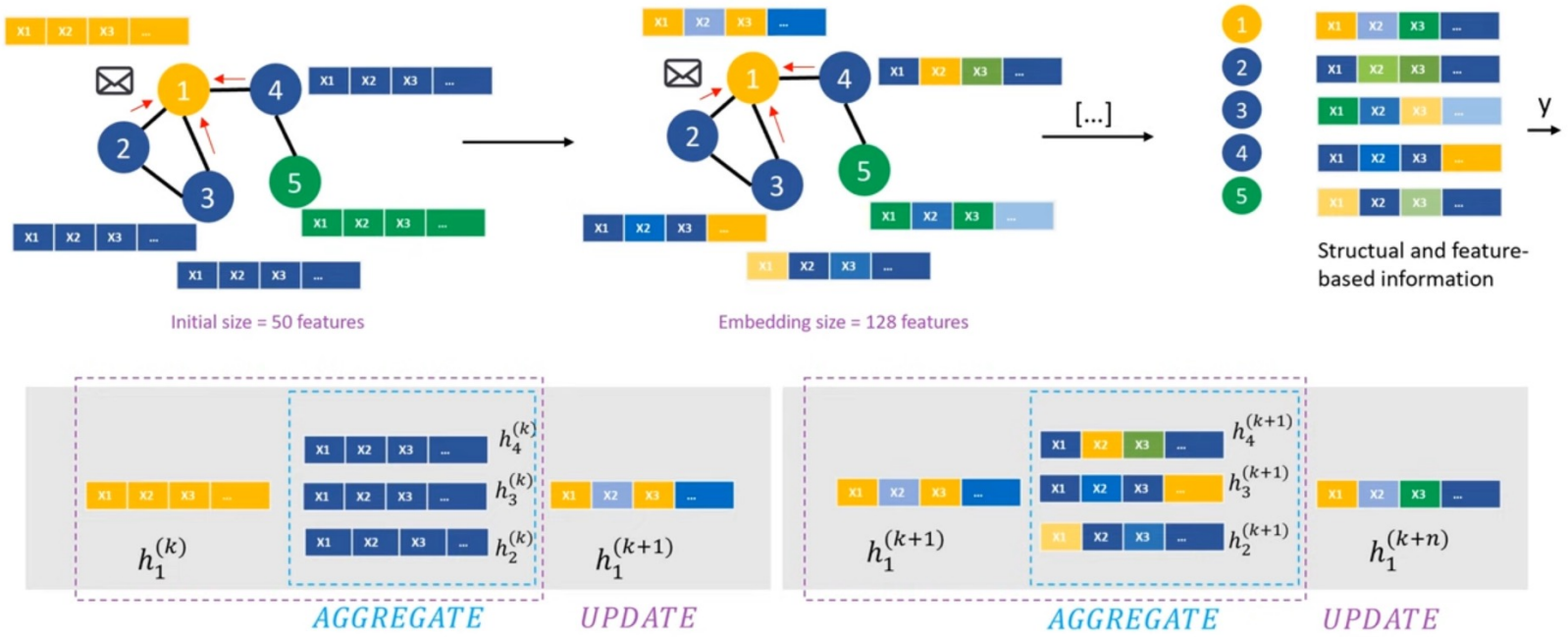


Illustration copied from DeepFindr youtube channel
Video link: https://www.youtube.com/watch?v=ABCGCf8cJOE&t=315s

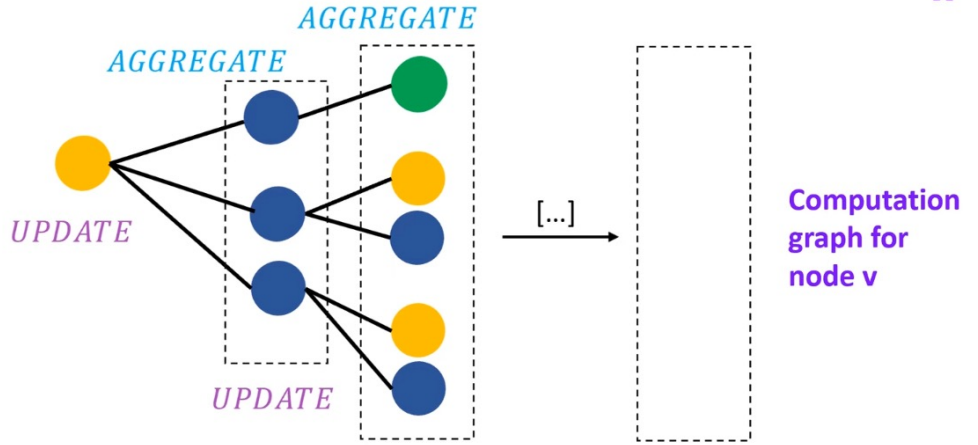
Message Passing in GCN

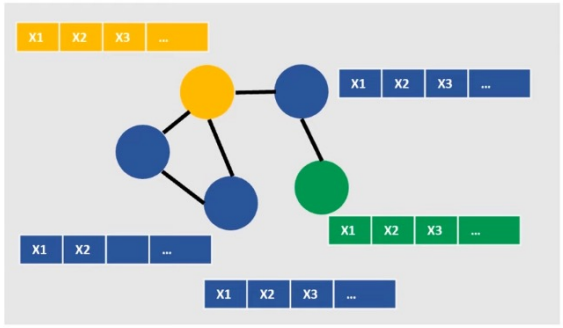
What is happening in the Message Passing Layers?

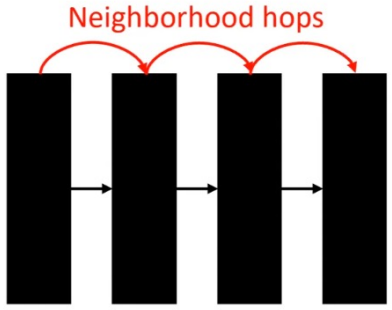


Number of Layers: Aggregate how many neighborhood hops?

Computation Graph Representation



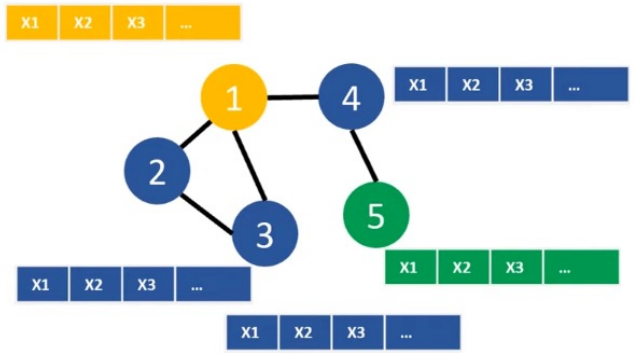




The number of MP-layers is a hyperparameter

Message Passing Update and Aggregation Functions

$$h_u^{(k+1)} = UPDATE^{(k)}\left(h_u^{(k)}, AGGREGATE^{(k)}(\{h_v^{(k)}, \forall\, v \in \mathcal{N}(u)\})\right)$$



Message passing in simple GNN and GCN

GNN variants

Graph Convolutional Networks, Kipf and Welling [2016]

$$\mathbf{h}_v^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)$$

Self-loop

Sum of normalized neighbor embeddings

- **Simple neighborhood aggregation:**

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

CNN vs GNN

Convolutional neural network (CNN) layer with 3x3 filter:

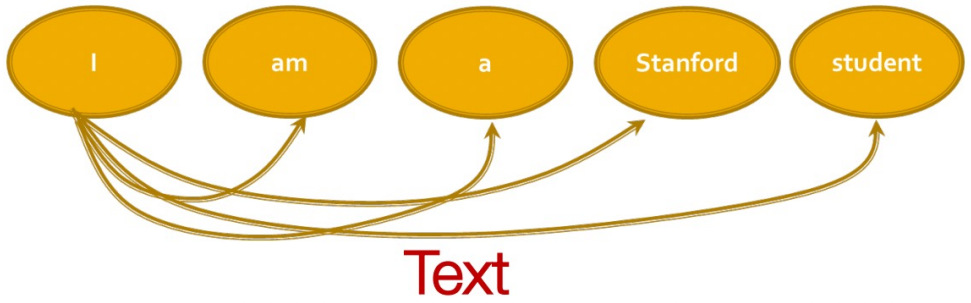
- **CNN can be seen as a special GNN with fixed neighbor size and ordering.**
- **CNN is not permutation equivariant.**
  - **Switching the order of pixels will leads to different outputs.**

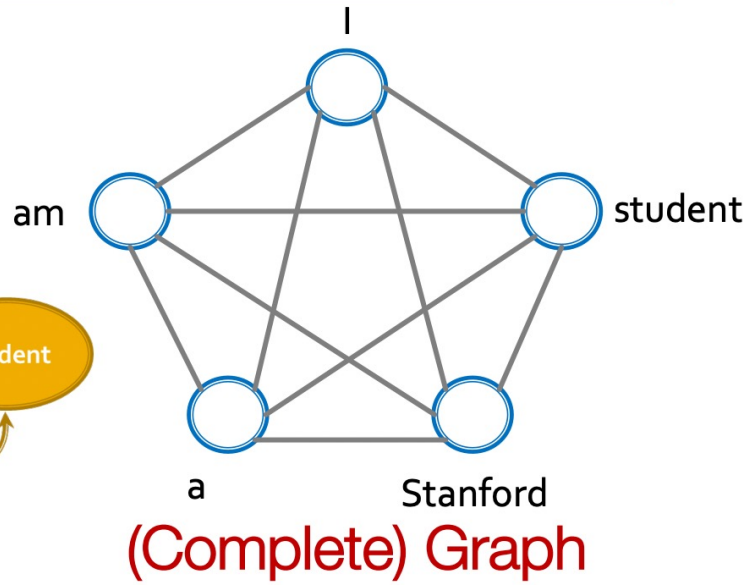
Key difference: We can learn different W_l^u for different "neighbor" u for pixel v on the image. The reason is we can pick an order for the 9 neighbors using **relative position** to the center pixel: {(-1,-1). (-1,0), (-1, 1), …, (1, 1)}

CNN vs Transformers

Transformer layer can be seen as a special GNN that runs on a fully-connected "word" graph!

Since each word attends to **all the other words**, **the computation graph** of a transformer layer is identical to that of a GNN on the **fully-connected "word" graph**.





Recurrent Neural Networks

Key Ideas

- Recurrent neural networks (RNN) are a class of neural networks that are helpful in modeling sequence data.
- produces predictive results in sequential data that other algorithms can’t.
- When do you need it:
  - “Whenever there is a sequence of data and that temporal dynamics that connects the data is more important than the spatial content of each individual frame.” – Lex Fridman (MIT)

Reminder of the lecture is from Stanford. Credits to Drs. Fei Fei Li, Ranjay Krishna and Danfei Xu