CS502: Deep Learning in Biomedicine
Transformers

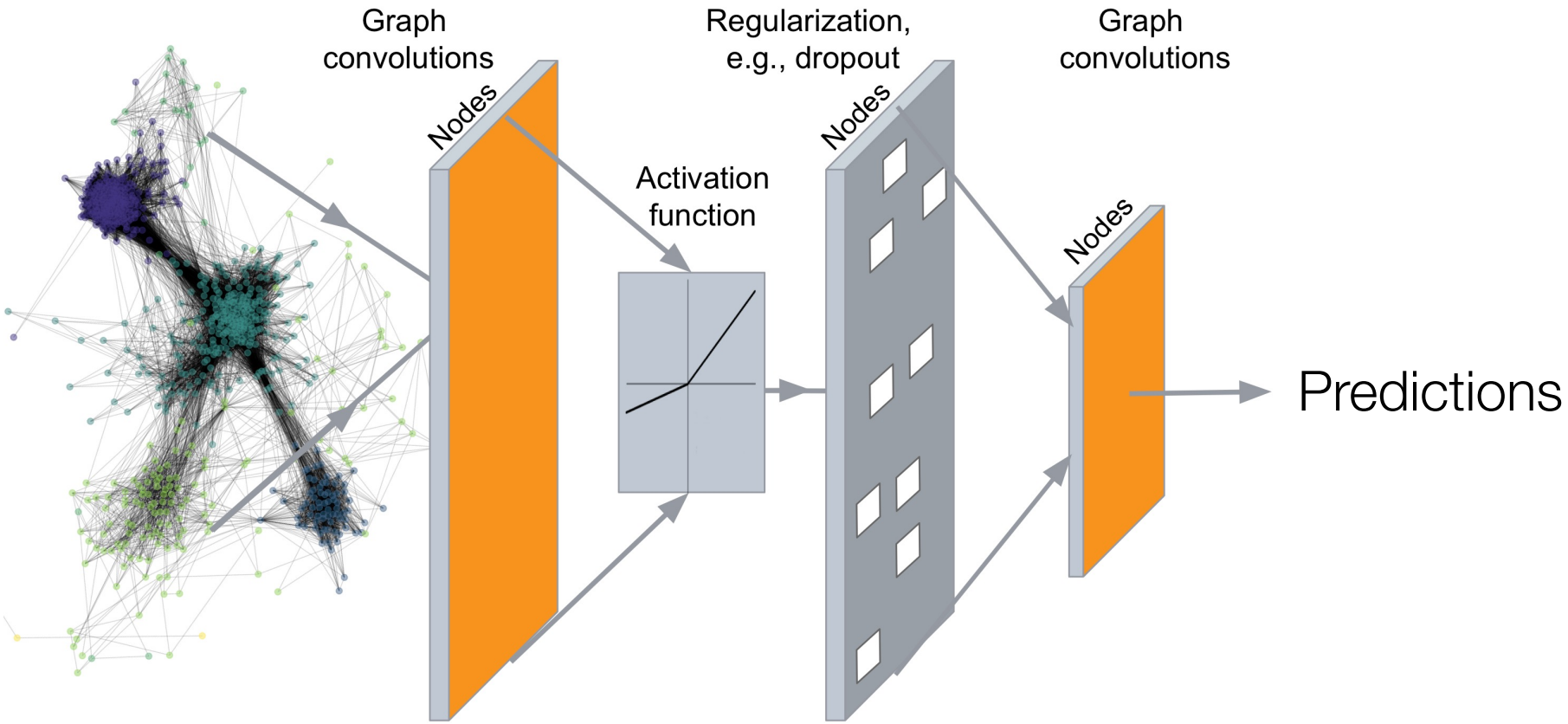
Maria Brbić

Fall 2023

École
polytechnique
fédérale
de Lausanne

Last Week: Recap

- What we covered last time: Graph Convolutional Neural Networks



Sequence Data is Everywhere!

Sentiment classification

"At times poignant, joyful, and terrifying, Shawshank Redemption is an altogether brilliant movie and the debut of an equally brilliant director" → ★★★/★★★

Named-entity recognition

I am flying to New York at 5pm. →

LOC	TIME
New York	5pm

Machine translation

Le vent se lève... il faut tenter de vivre → The wind rises... We must try to live!

Speech recognition

 → Houston, we have a problem.

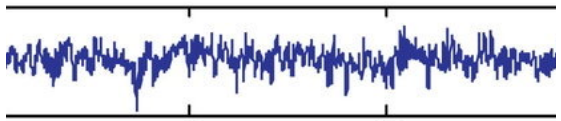
Biological/ biomedical sequences

… GTGCATCTGACTCTGA….

DNA sequence

… VHKTPEEK….

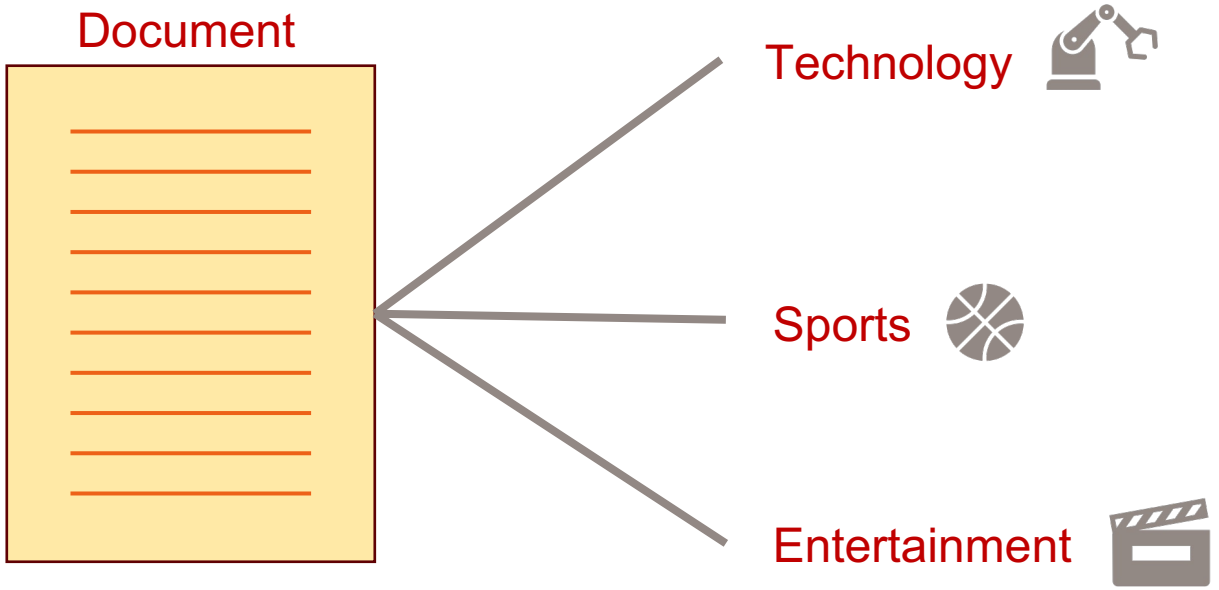
Protein sequence



EEG data

A Sequence Modeling Problem

Take the sequence as an input, and produce a desired output



Challenge: Variable input length

A Sequence Modeling Problem

Can we use feedforward neural networks for this type of data? 🤔

- Idea 1: Count number of appearances of each word
  - One dimension per word in vocabulary

"I will make him an offer he can not refuse."

I	will	make	him	an	offer	he	can	not	refuse	…
1	1	1	1	1	1	1	1	1	1	0

Bag-of-words representation

Problem: does not capture sequence information!

"my cat was chased by a dog" ⟷ *"my dog was chased by a cat"*

same representation

A Sequence Modeling Problem

Can we use feedforward neural networks for this type of data? 🤔

- Idea 2: Fixed sized window

"I will make him an offer he can not refuse."

Given these 2 words — Predict next word

Problem: very limited context while dependencies can be long-term

A Sequence Modeling Problem

Can we use feedforward neural networks for this type of data? 🤔

- Idea 3: Can we just use a larger window size?

"I will make him an offer he can not refuse."

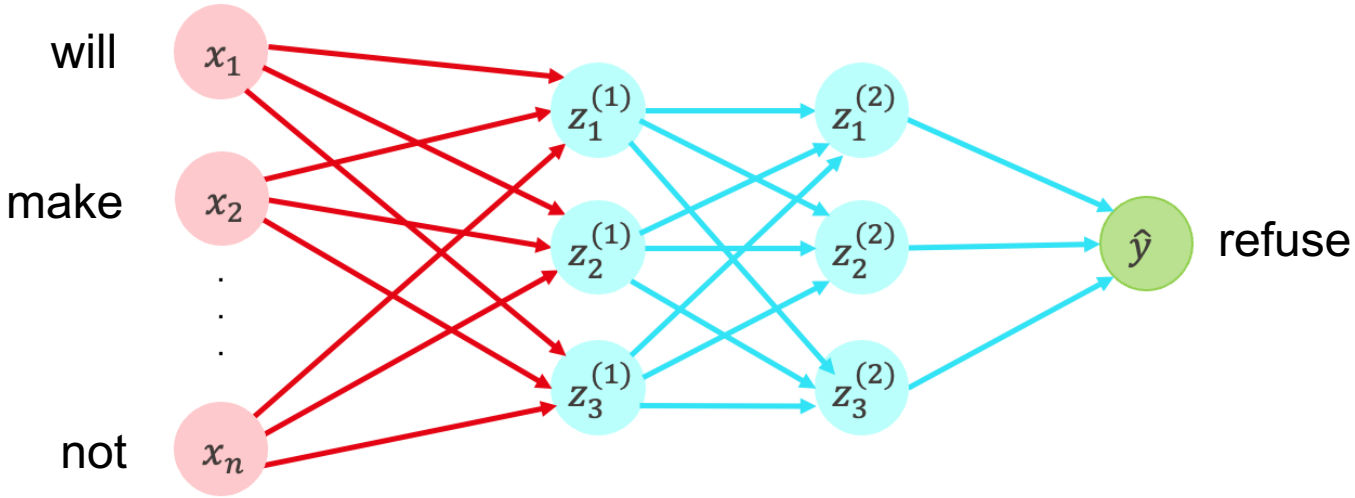
Given these 7 words

Predict next word

Problem: each of these inputs has a separate parameter

A Sequence Modeling Problem

Problem: each of these inputs has a separate parameter



Feedforward networks do not work for sequence modeling!

So, What Do We Need for Sequence Modeling?

We need:

- A way to deal with variable-length input size
- A way to capture long-term dependencies
- Share parameters across the sequence

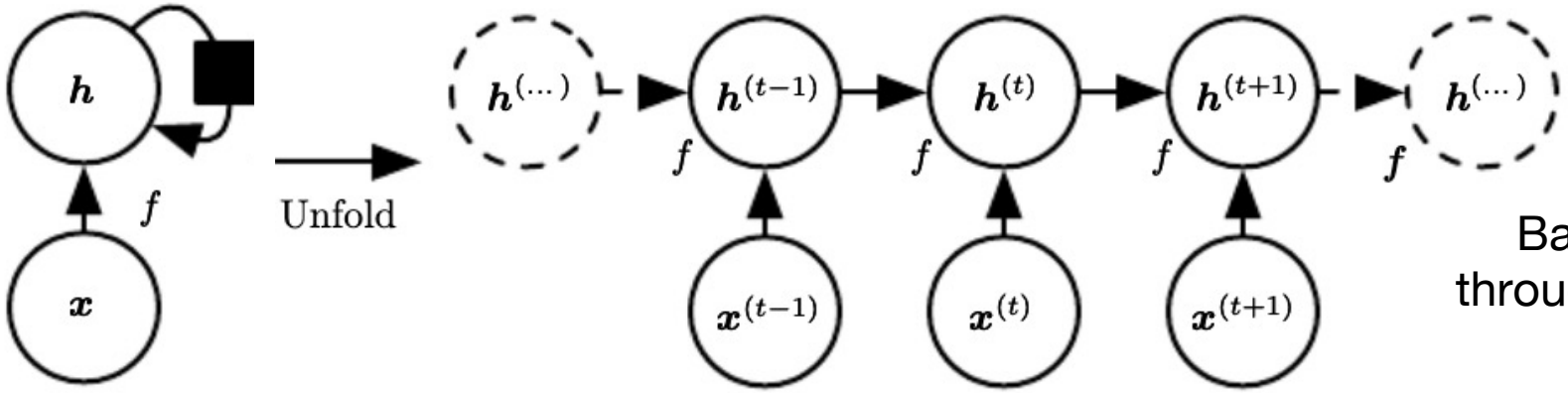
How To Model Sequences with Neural Networks?

- Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986)
  - Key idea: define output as a function of previous inputs at previous stages

$$h_t = f(h_{t-1}, x_t)$$

state at previous time stamp

input at current time stamp



Backpropagation through time to update parameters

Figure from Goodfellow et al. Deep Learning. MIT Press 2016

How To Model Sequences with Neural Networks?

- Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986)
  - Key idea: define output as a function of previous inputs at previous stages

$$h_t = f(h_{t-1}, x_t)$$

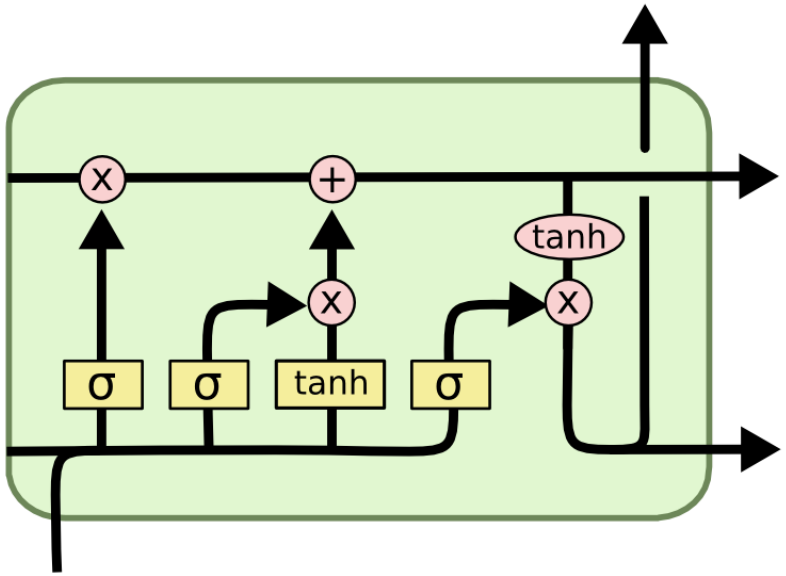
- Example for $t = 3$: $h_3 = f(f(f(h_0, x_1), x_2), x_3)$

- Problem: Vanishing gradient caused by multiplying a lot of small gradients during backpropagation through time
- Result: Vanilla RNN do not work for long sequences

How To Model Sequences with Neural Networks?

- Long Short-Term Memory Networks (LSTMs) (Hochreiter & Schmidhuber, 1997)
  - Solves exploding and vanishing gradient

- Problems:
  - Hard to train: very long gradient paths
  - Transfer learning does not work



LSTM cell. Figure from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

How To Model Sequences with Neural Networks?

- Transformers to the rescue!

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

https://arxiv.org/pdf/1706.03762.pdf

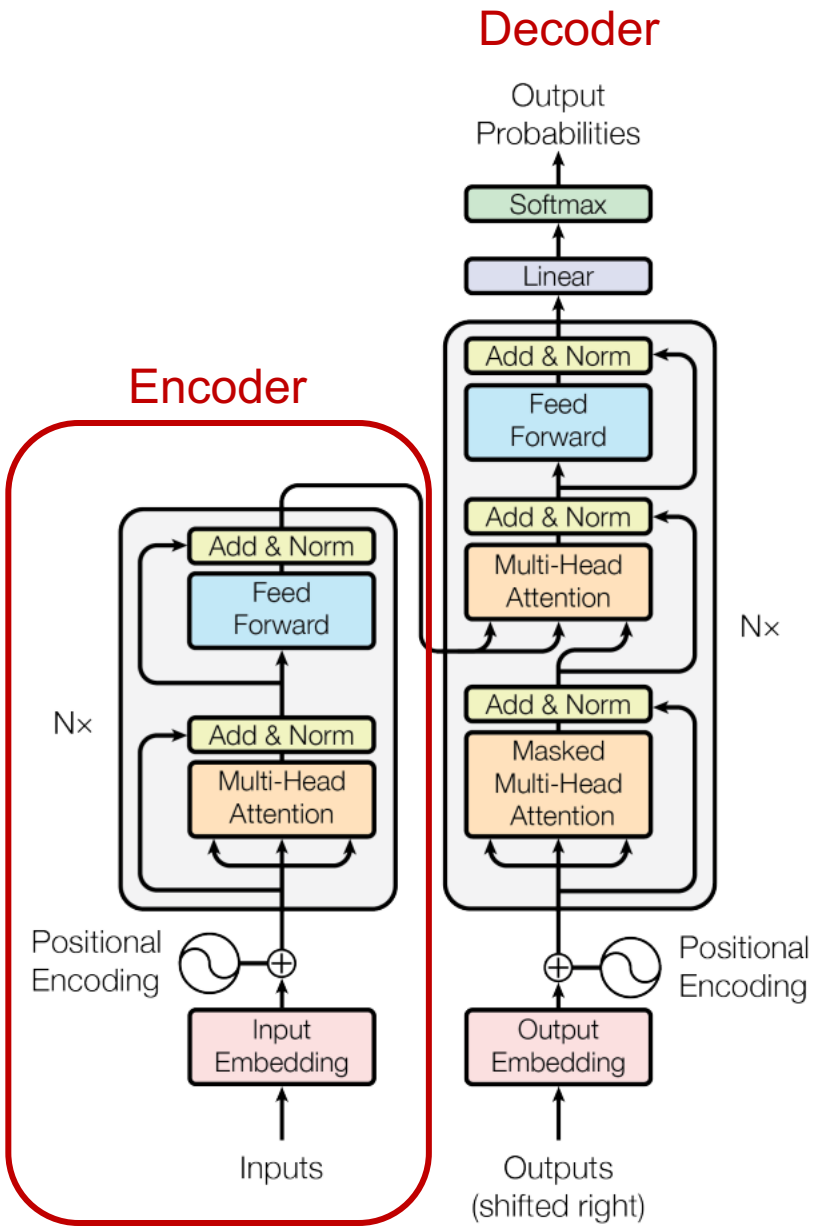
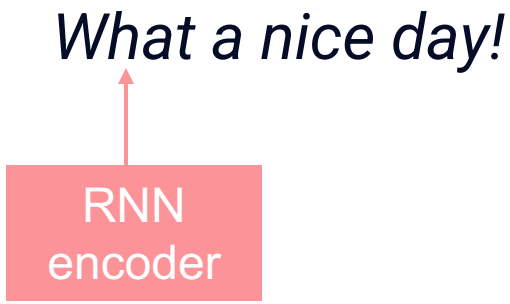


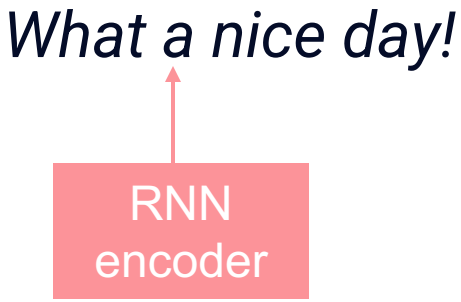
Figure 1: The Transformer - model architecture.

Figure from Vaswani et al. *NeurIPS* 2017

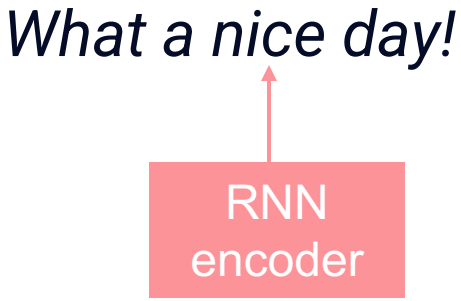
RNNs Vs Transformers: Conceptual Difference



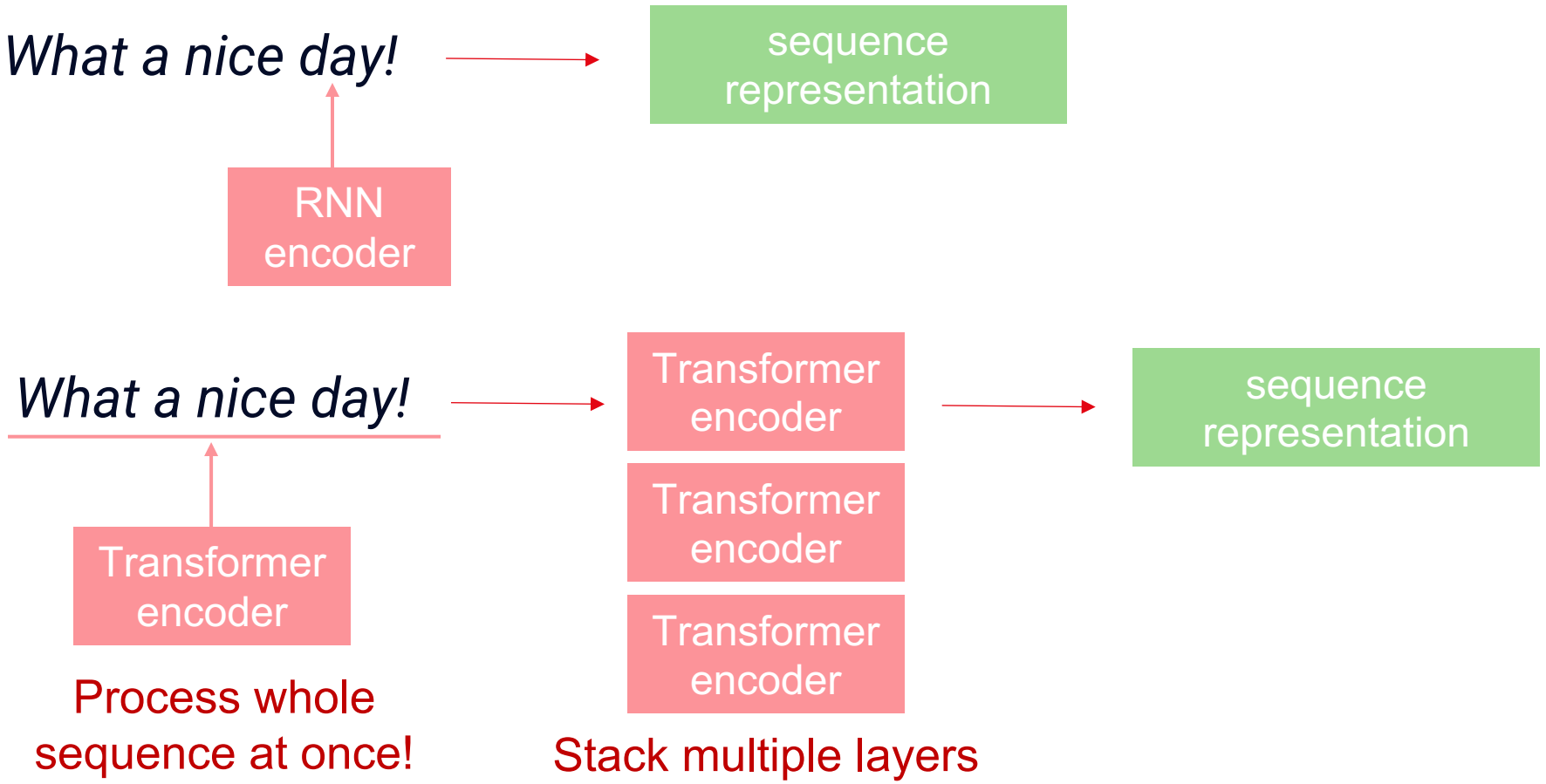
RNNs Vs Transformers: Conceptual Difference



RNNs Vs Transformers: Conceptual Difference



RNNs Vs Transformers: Conceptual Difference



RNNs Vs Transformers: Conceptual Difference

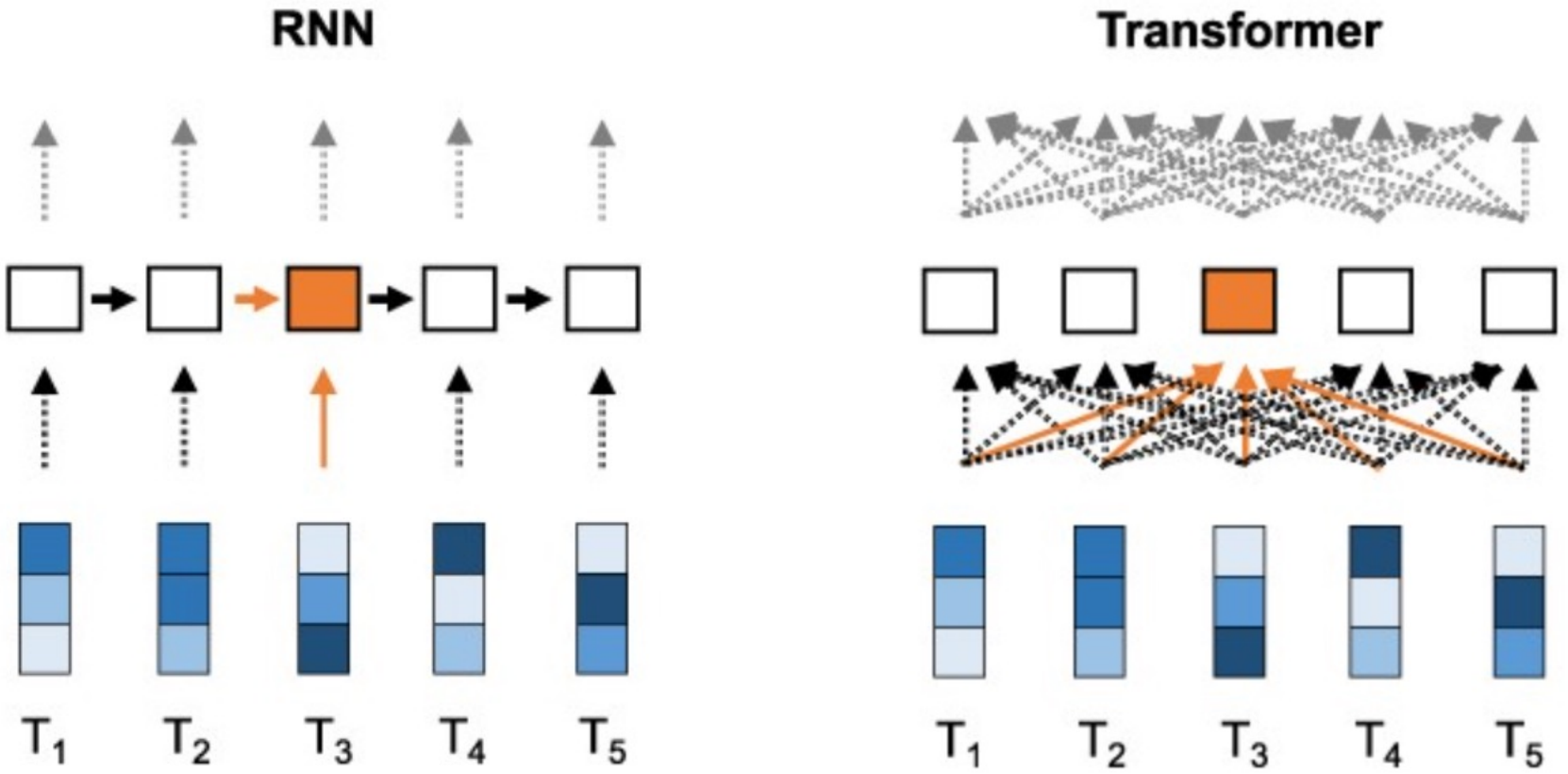
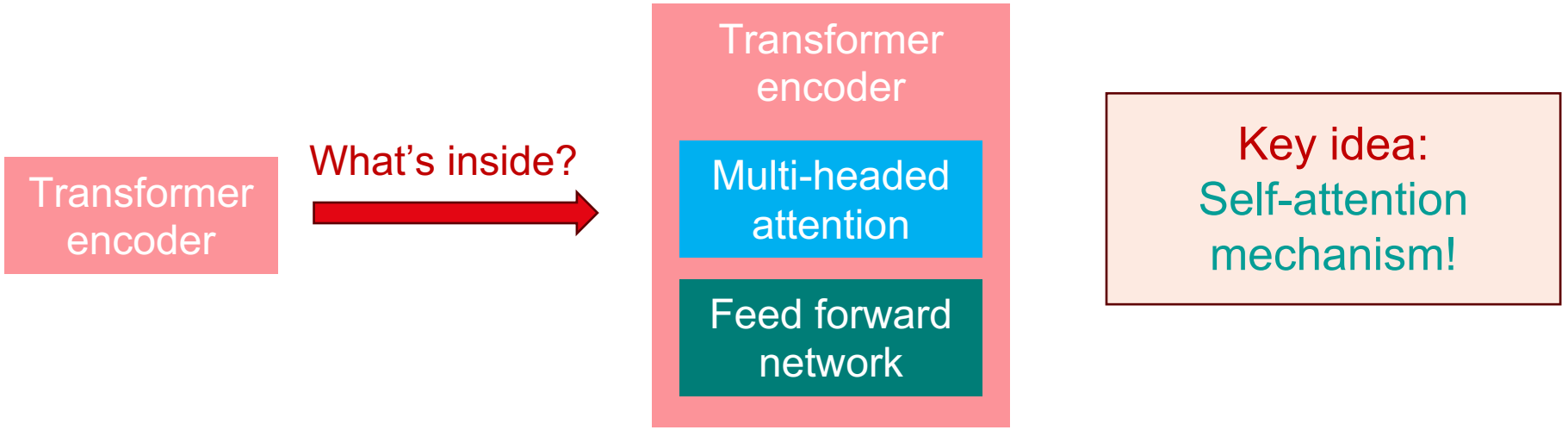


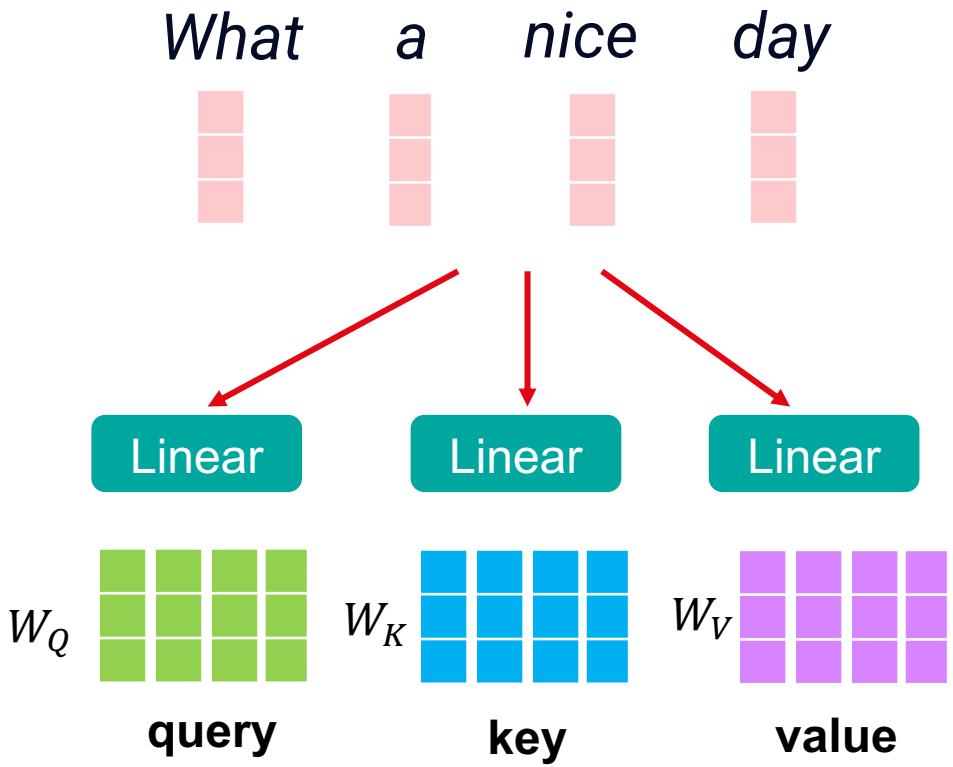
Figure from Ji et al. *Bioinformatics* 2021

Transformer Encoder

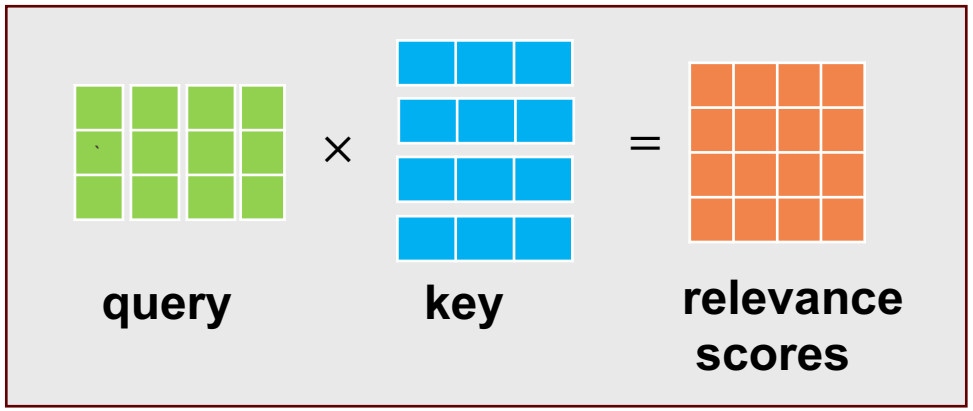


- Self-attention mechanism
  - Allows the model to associate each individual element of the sequence to other elements in the sequence

Transformer Encoder: Self-Attention Mechanism



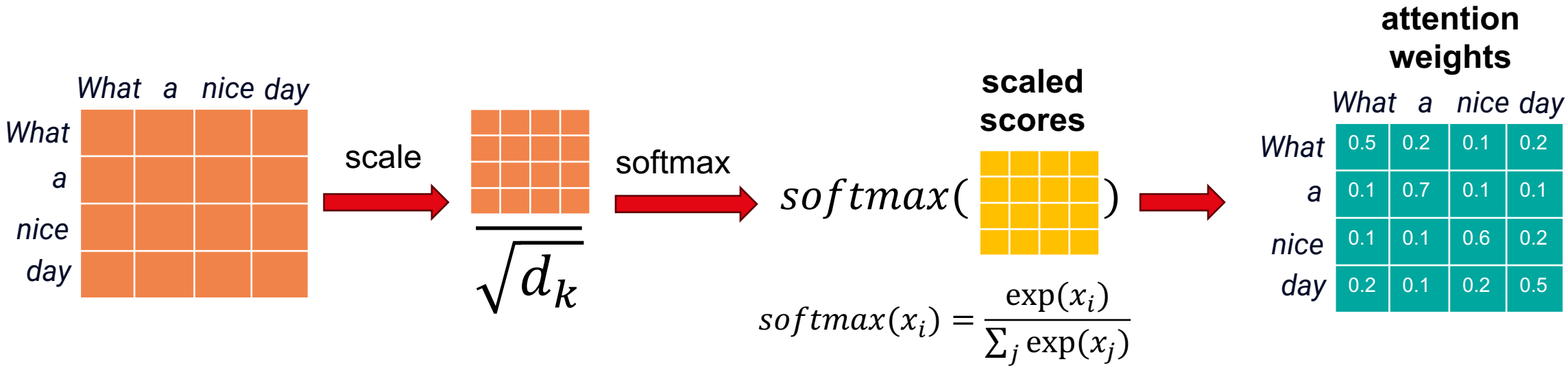
Multiply queries and keys



	What	a	nice	day
What	91	31	10	30
a	31	93	28	32
nice	10	28	89	45
day	30	32	45	94

How relevant word *day* is to word *nice*

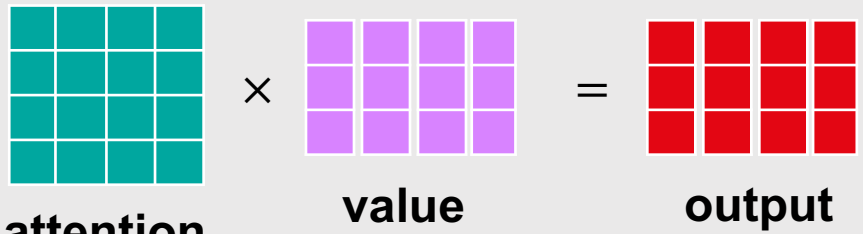
Transformer Encoder: Self-Attention Mechanism



$$\sqrt{d_k}$$

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Multiply attention scores and values



d_k: query and key dimension

Transformer Encoder: Self-Attention Mechanism

- Putting it all together:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention

Q: query matrix
K: key matrix
V: value matrix
d_k: query and key dimension

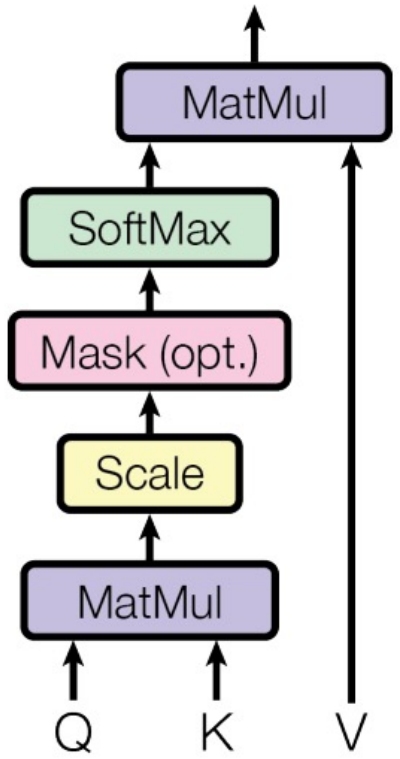
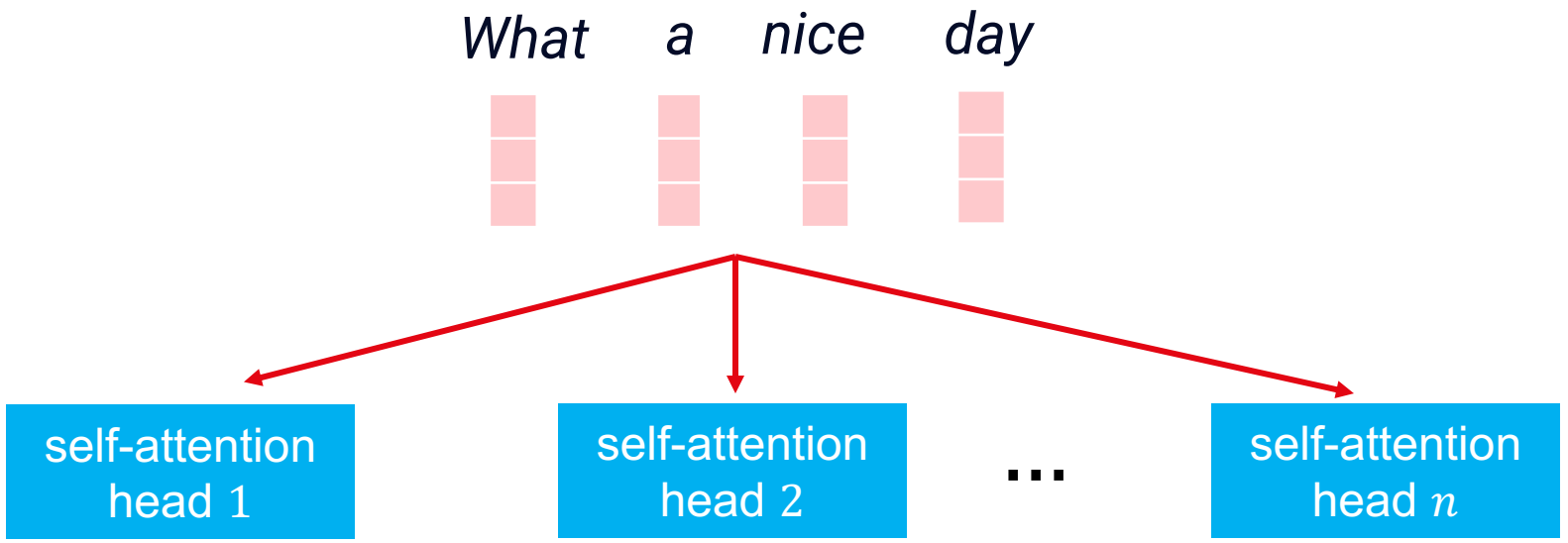


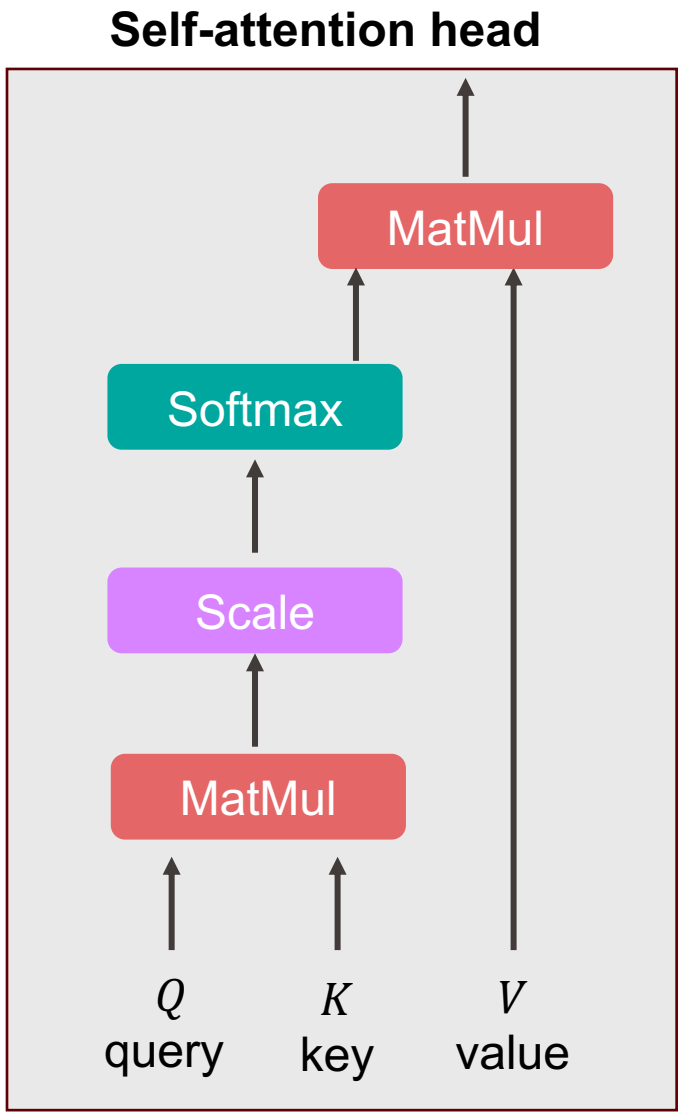
Figure from Vaswani et al. *NeurIPS* 2017

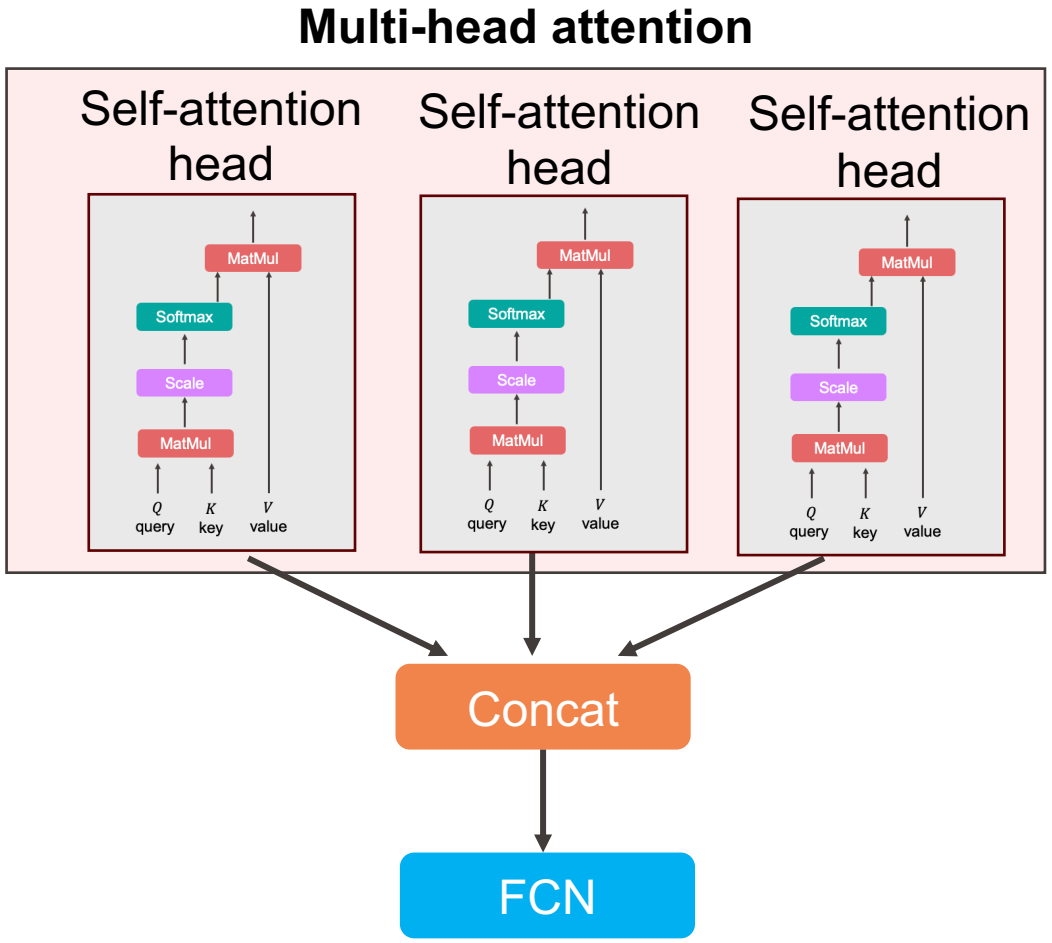
Multi-head Attention



- Apply same self-attention process multiple times individual with different (query, key, value) pairs
- Each self-attention process is called a head
- Each attention head should capture different information

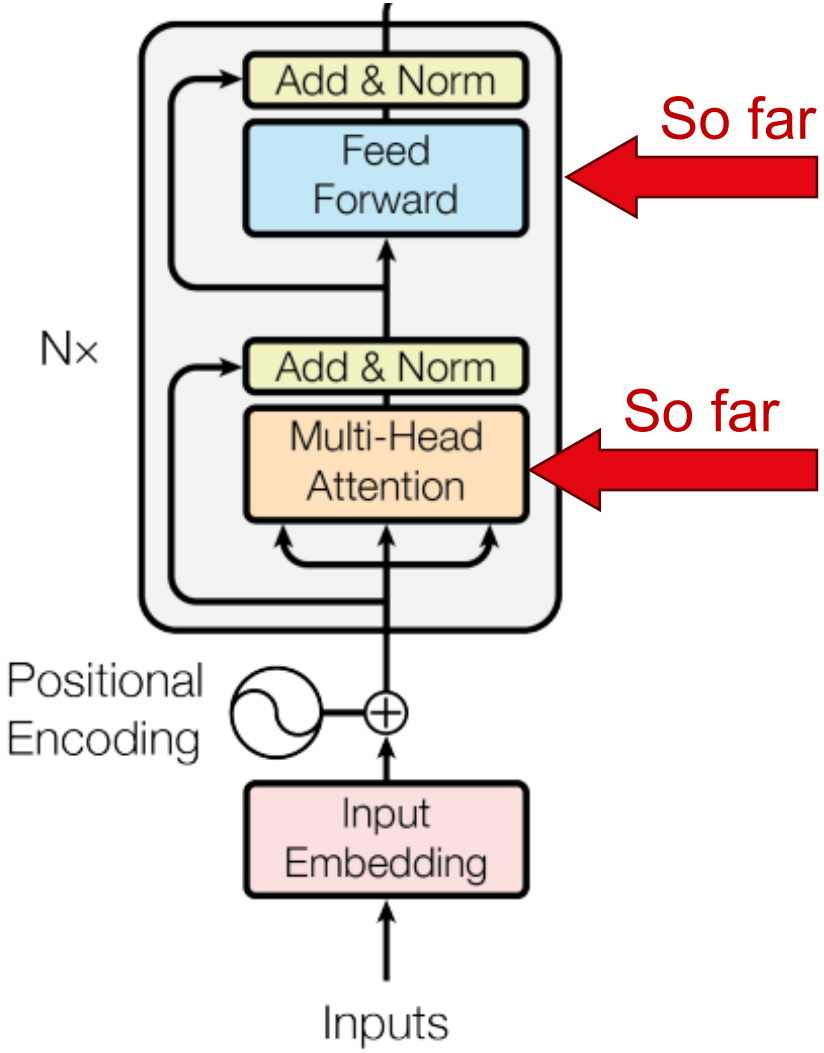
Overview: Multi-head Attention



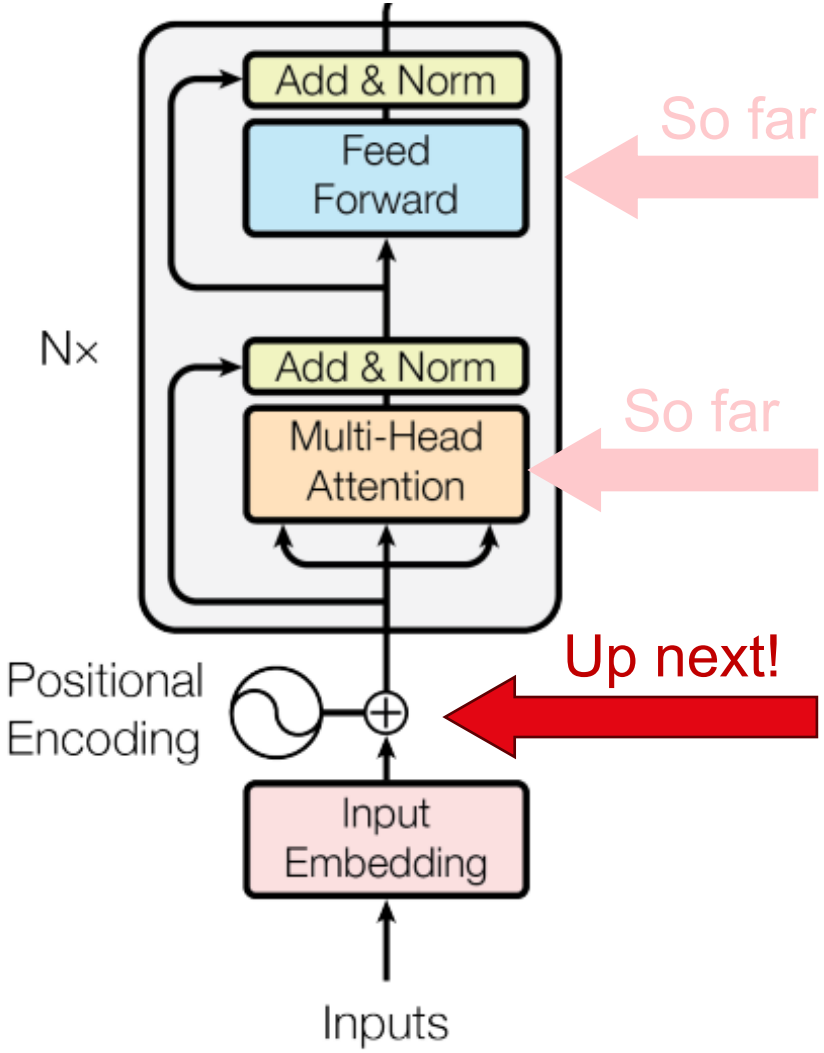


- Concatenate outputs from self-attention heads
- Feed to a feedforward neural network

So Far: Multi-head Attention



Next: Positional Encoding



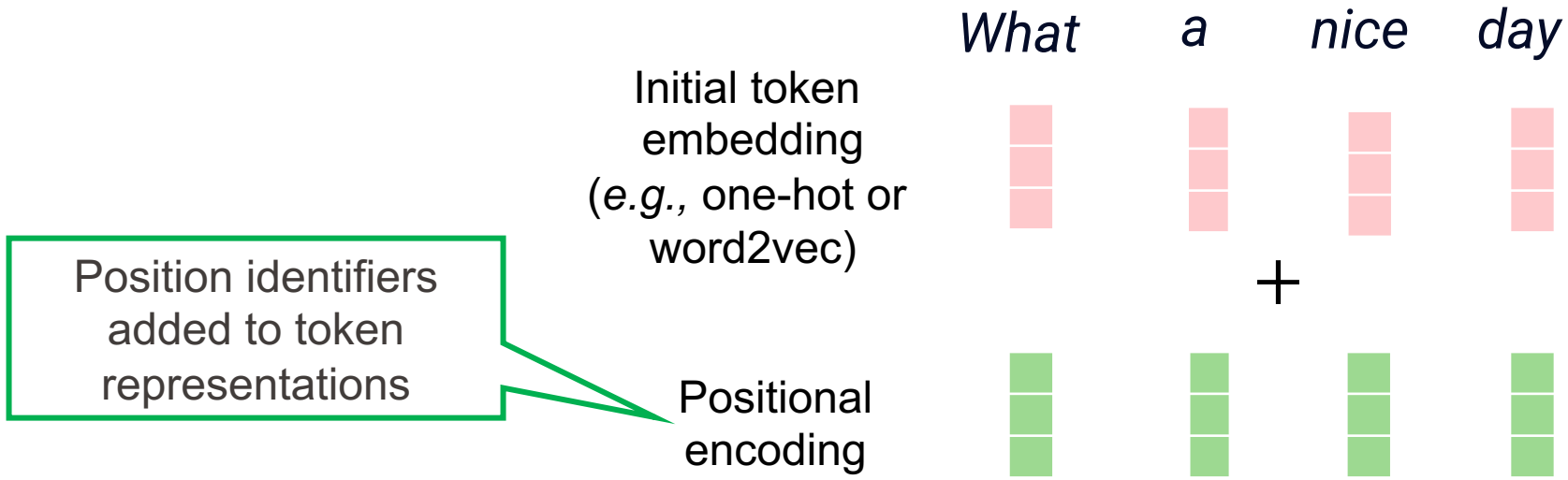
- Attention block does not model order of the sequence → we need to model order explicitly

But, how?

Positional encoding!

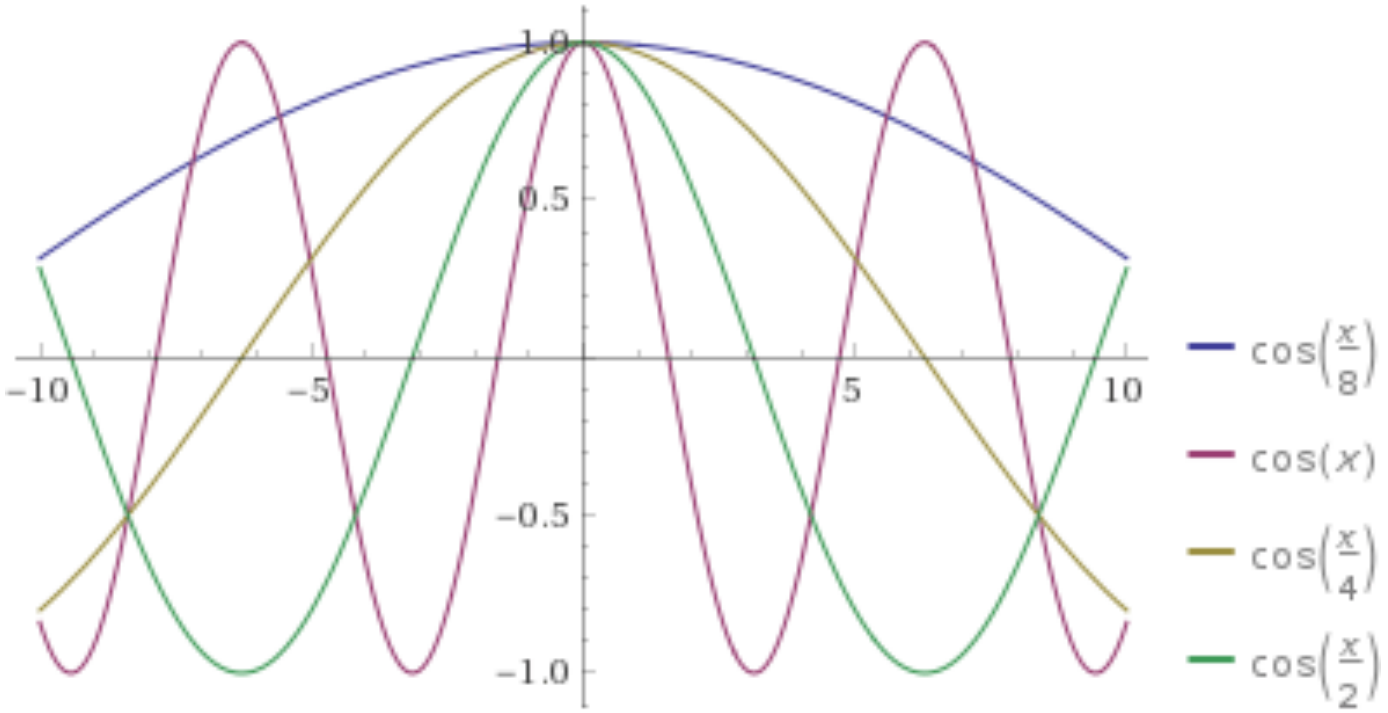
- Enables the model to reason about the position of tokens

Positional Encoding



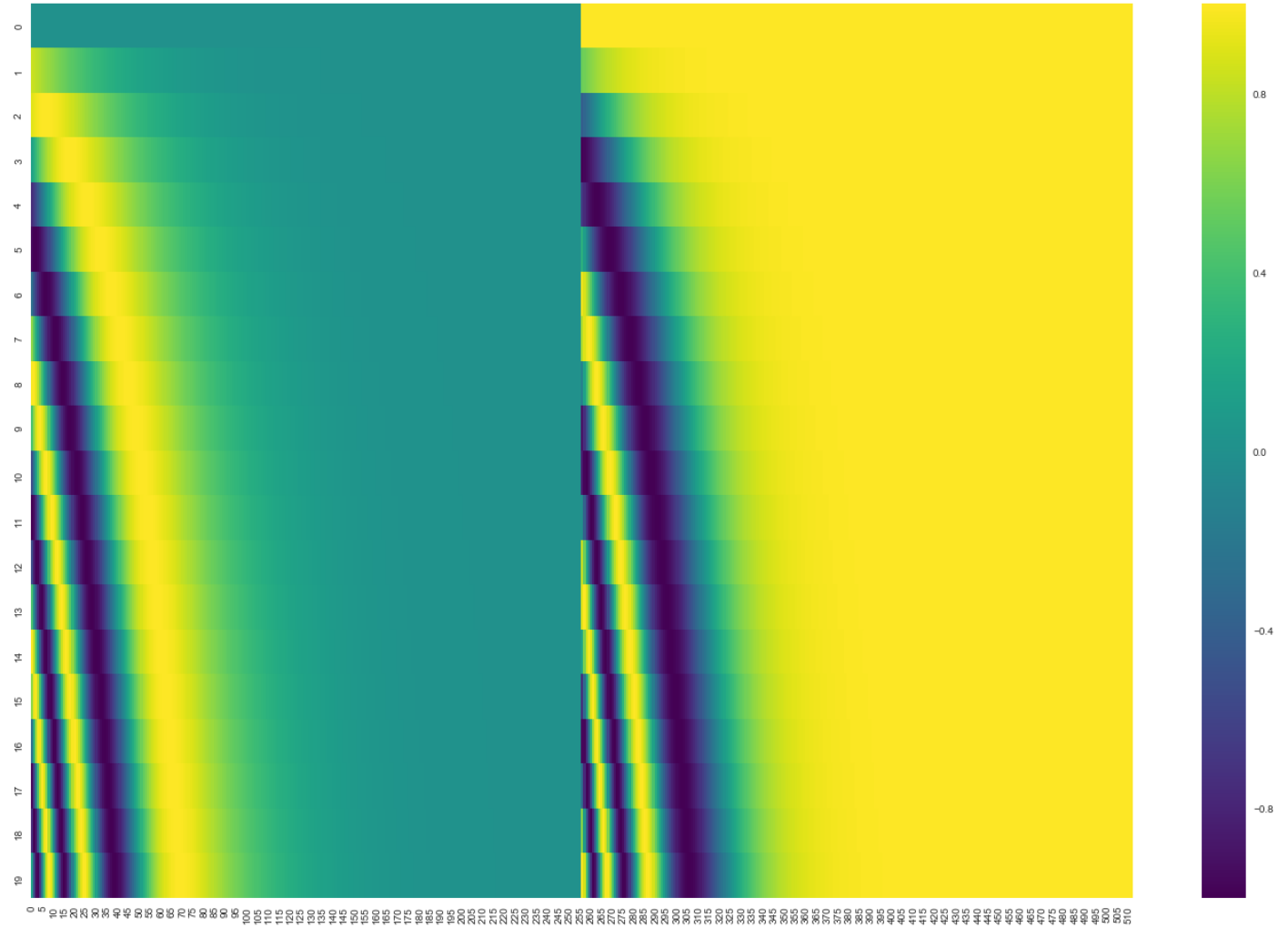
- Properties:
  - Distance between any two time steps should be consistent across sequences with different lengths
  - Shifts should be small so that they not overtake the semantic similarity
  - The values must be bounded and deterministic
  - Needs to generalize to longer sequences

Positional Encoding



- Solution:
  - Use sine and cosine functions with different frequencies to encode positional information!

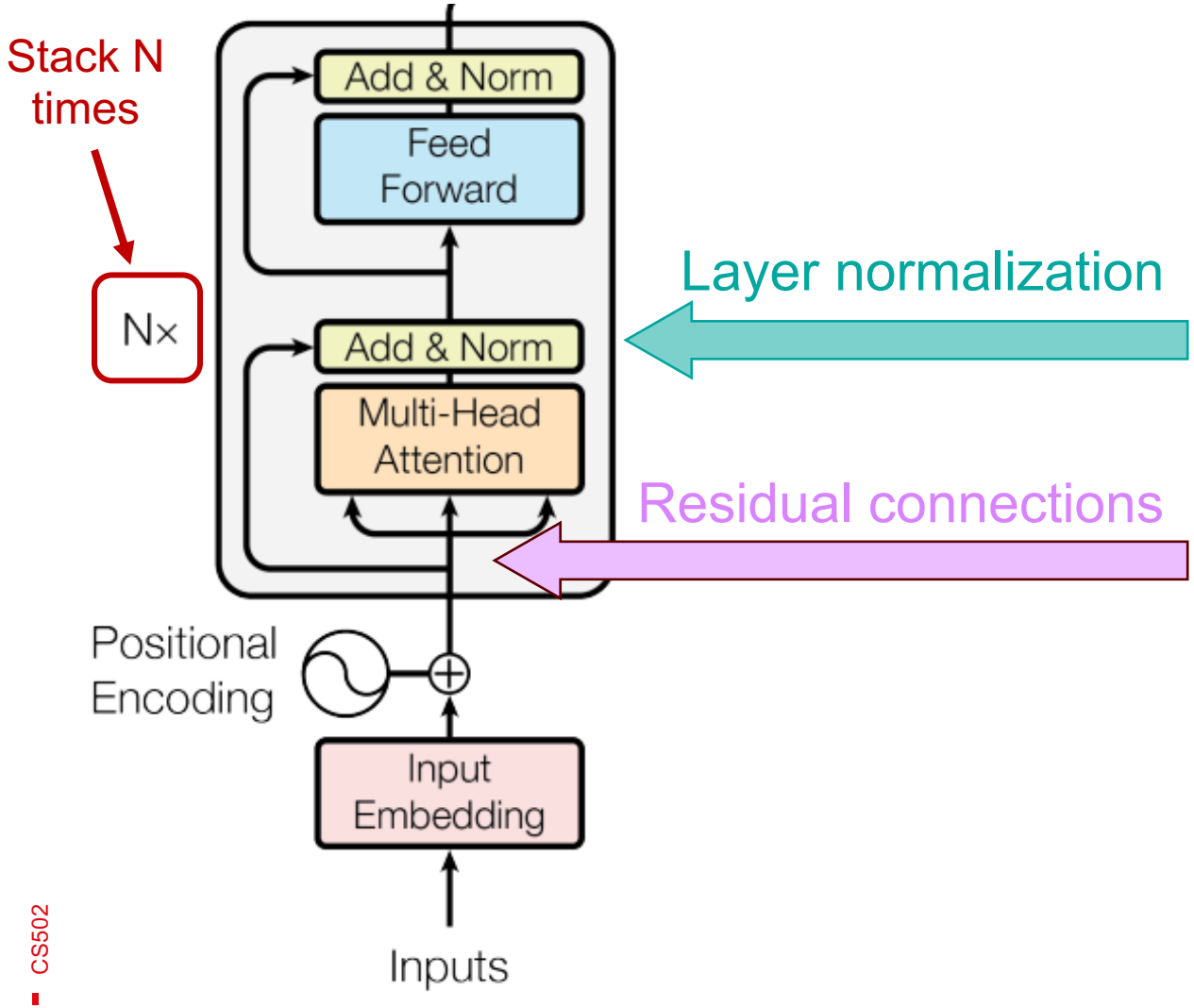
Positional Encoding



Example of positional encoding for 20 words (rows) with an embedding size of 512 (columns). The result appears split in half down the center. That's because the values of the left half are generated by one function (which uses sine), and the right half is generated by another function (which uses cosine). They are then concatenated to form each of the positional encoding vectors.

Figure from https://jalammar.github.io/illustrated-transformer/

Transformer Encoder



- Residual connections
  - Allow the gradients to flow directly through the network
- Layer normalization[1]
  - It enables smoother gradients, faster convergence, and better generalization accuracy

[1] https://pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

https://arxiv.org/pdf/1706.03762.pdf

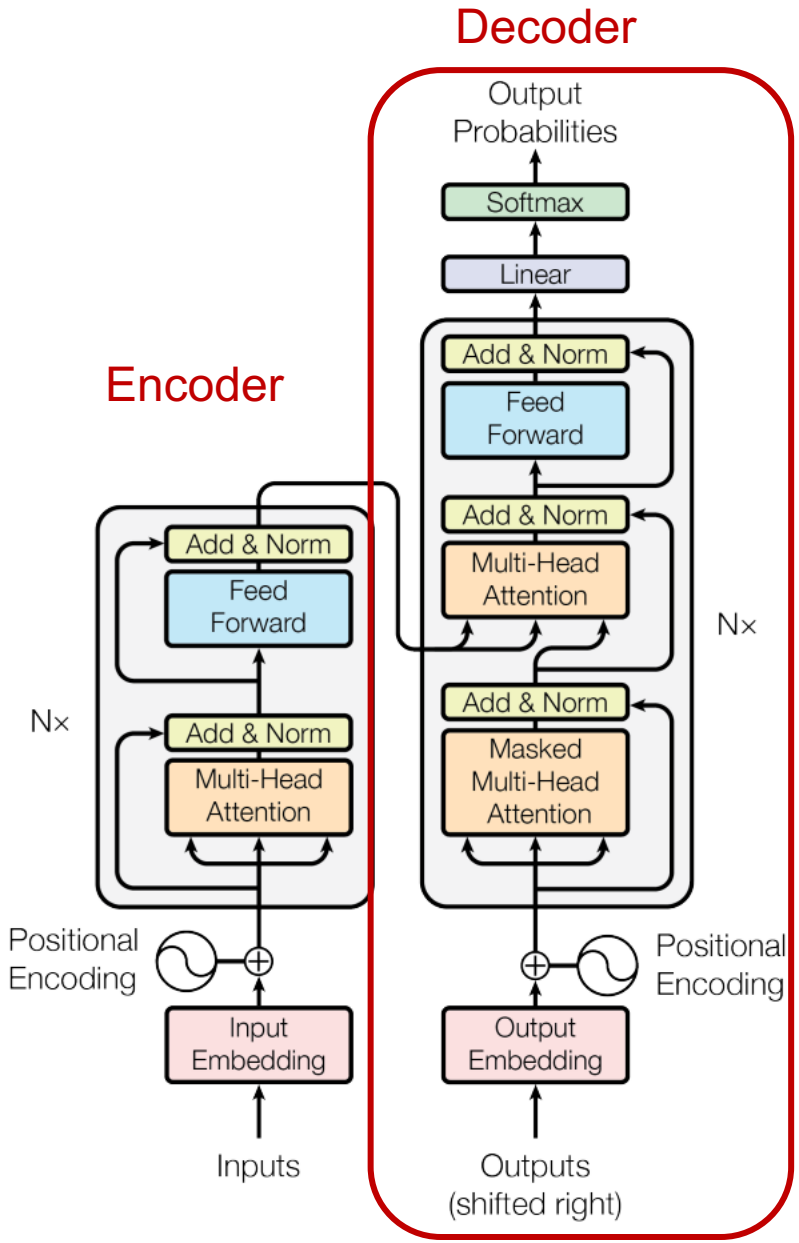


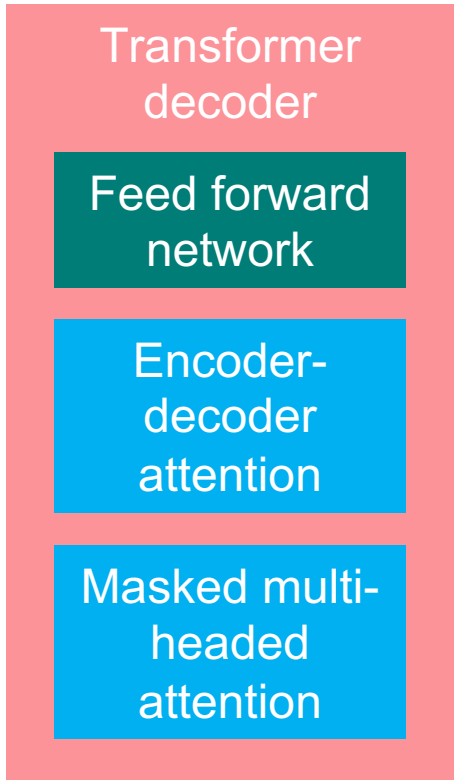
Figure 1: The Transformer - model architecture.

Figure from Vaswani et al. *NeurIPS* 2017

Transformer Decoder

Transformer decoder

What's inside?

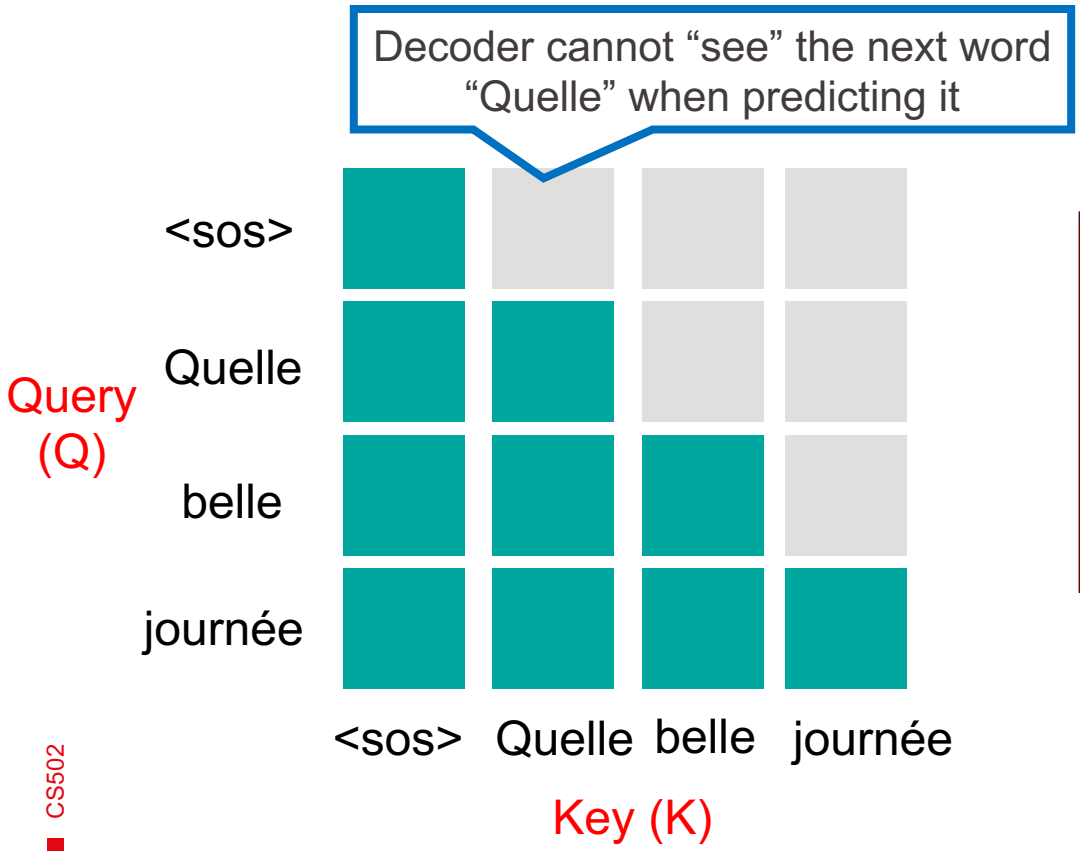


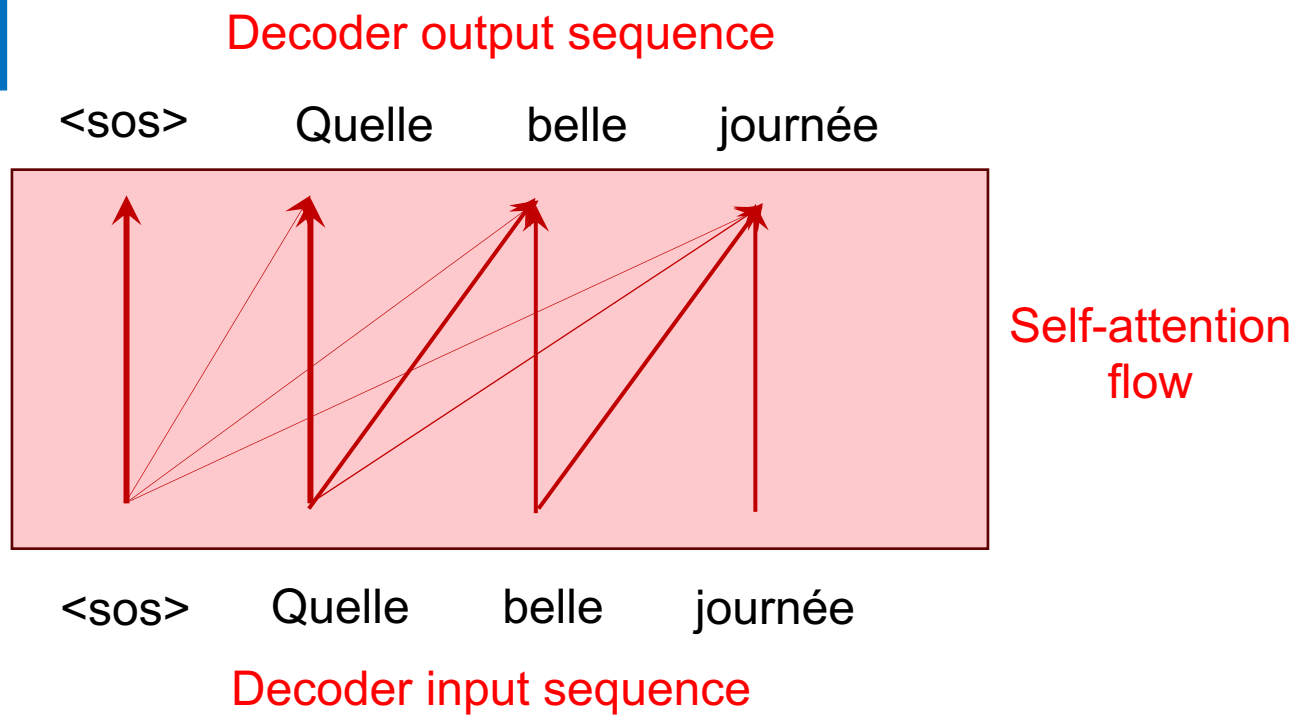
Key Ideas:
Masked self-attention
Encoder-decoder attention

- Masked attention
  - Need to prevent the decoder from cheating by stopping information flow from the "future"
- Encoder-decoder attention
  - Allow the decoder to leverage representation learned by the encoder in generating predictions

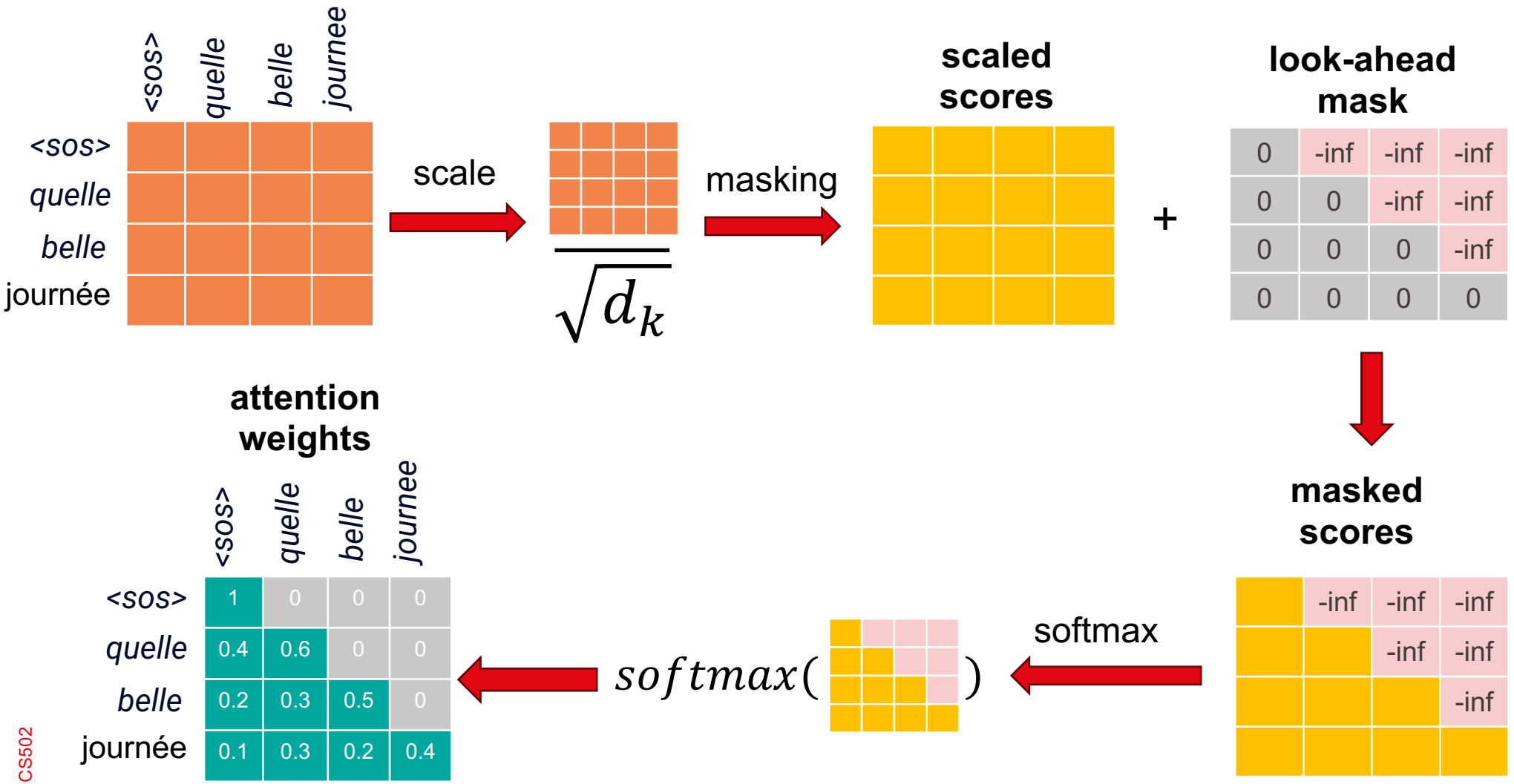
Transformer Decoder: Masked Self-Attention

- Each position in the decoder can only attend to previous positions otherwise next-token prediction would be trivial
- Different from encoder self-attention!





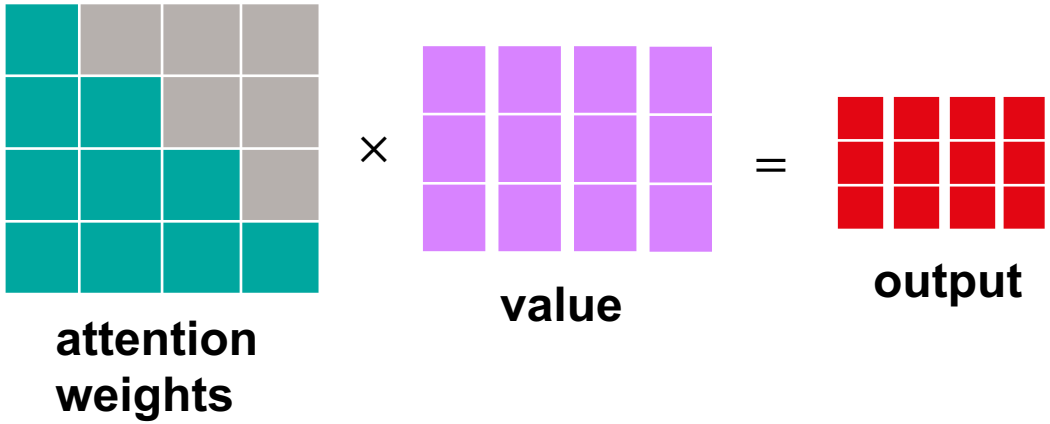
How Does Masking Work in Practice?



$$softmax\left(\frac{}{\sqrt{d_k}}\right)$$

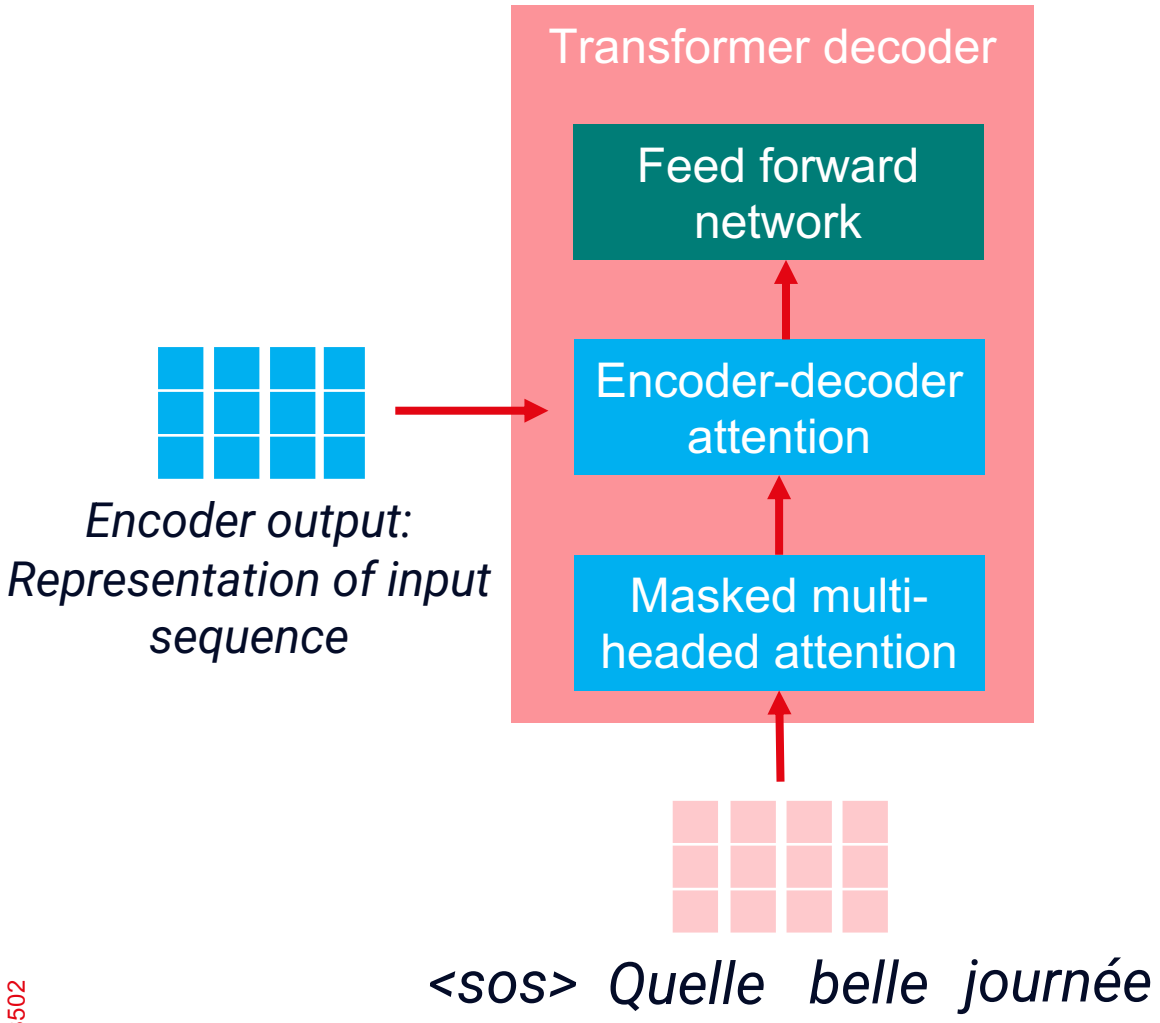
Transformer Decoder: Masked Self-Attention

Multiply attention scores and values



The N_{th} element in the output is only a weighted sum of the previous N values

Encoder-Decoder Attention

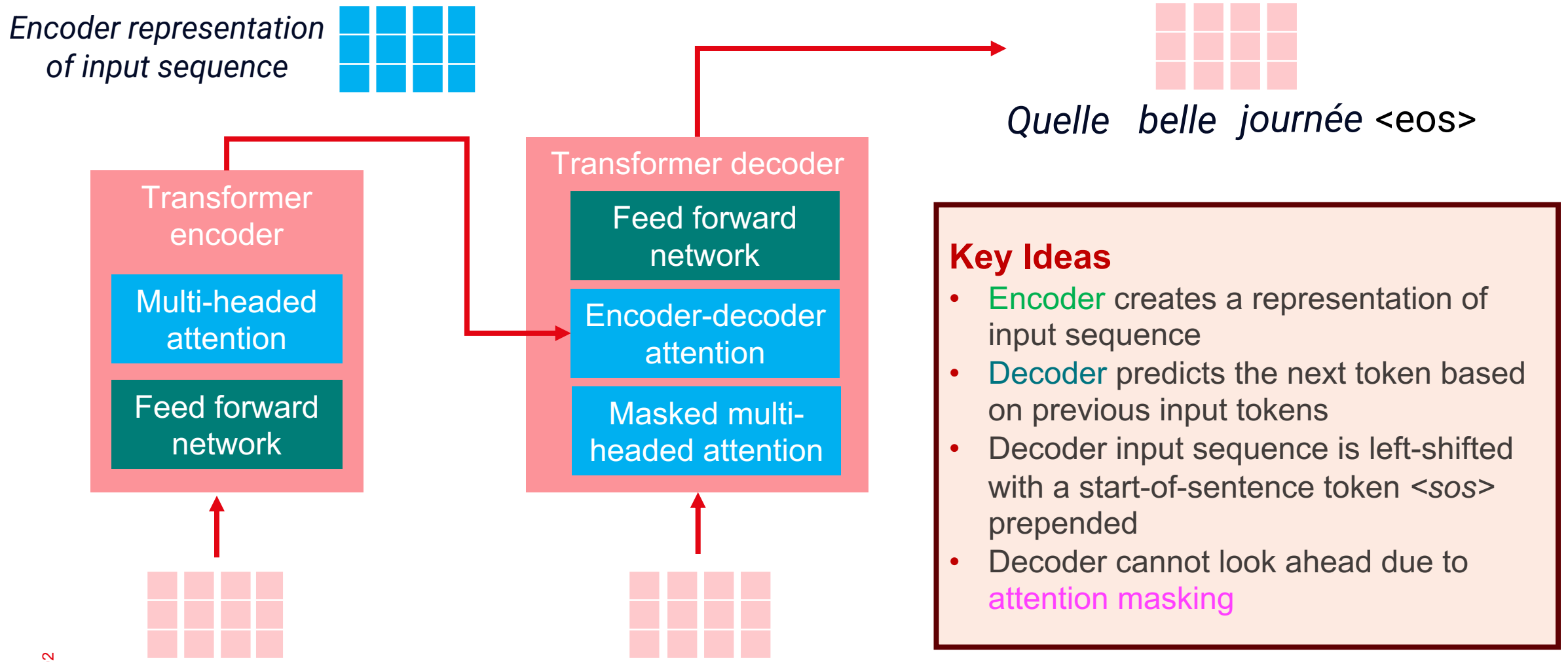


attention weights

	What	*a*	*nice*	*day*
<sos>	0.5	0.1	0.2	0.2
Quelle	0.1	0.7	0.1	0.1
belle	0.1	0.1	0.6	0.2
journée	0.3	0.1	0.1	0.5

- Each position in the decoder can associate with any element in the encoder embedding
- Similar to self-attention

Transformer: Putting It All Together

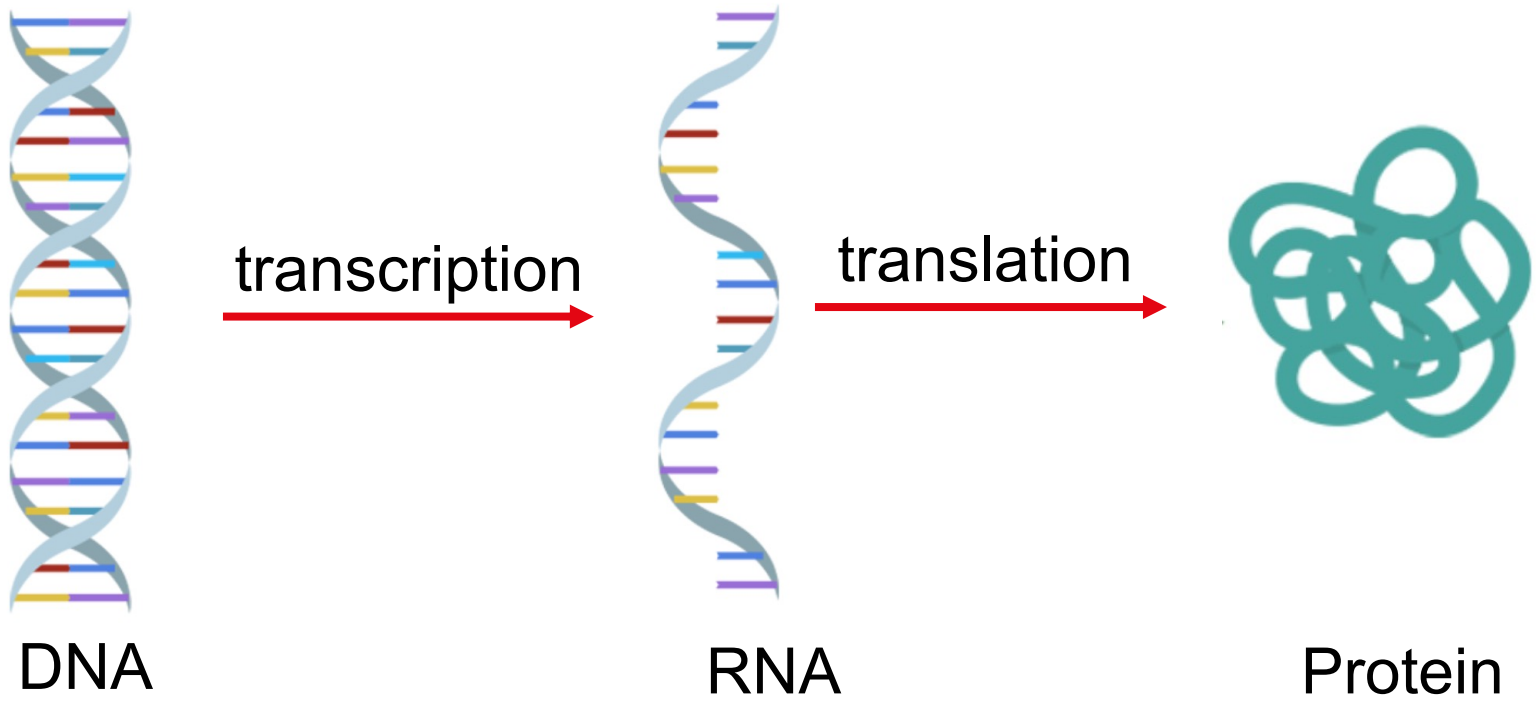


Key Ideas

- Encoder creates a representation of input sequence
- Decoder predicts the next token based on previous input tokens
- Decoder input sequence is left-shifted with a start-of-sentence token <*sos*> prepended
- Decoder cannot look ahead due to attention masking

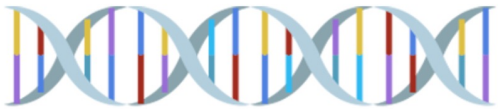
Biomedical Applications: Biological Sequence Modeling

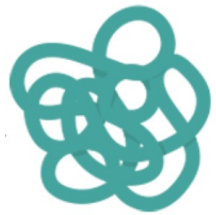
Information Flow in Molecular Biology



- DNA gets transcribed to RNA
- RNA gets translated to protein

Information Flow in Molecular Biology





Sequences!

...CACGTAGACTGAGGACTCCTCTTC...

transcription

...GUGCAUCUGACUCCUGAGGAGAAG...

translation

... V H L T P E E K ...

Biological Sequence Analysis

- Biological sequence analysis is one of the fundamental applications of computational methods in molecular biology
  - DNA, RNA and protein sequence analysis
- Traditional techniques:
  - K-mers (Koonin and Galperin, 2003)
  - CNNs (e.g., Zhou and Troyanskaya, 2015, Kelley et al., 2016)
  - RNNs and LSTMs (e.g., Jurtz et al., 2017)
- Nowadays:
  - **Transformers**

An Overview of Transformer Related Works

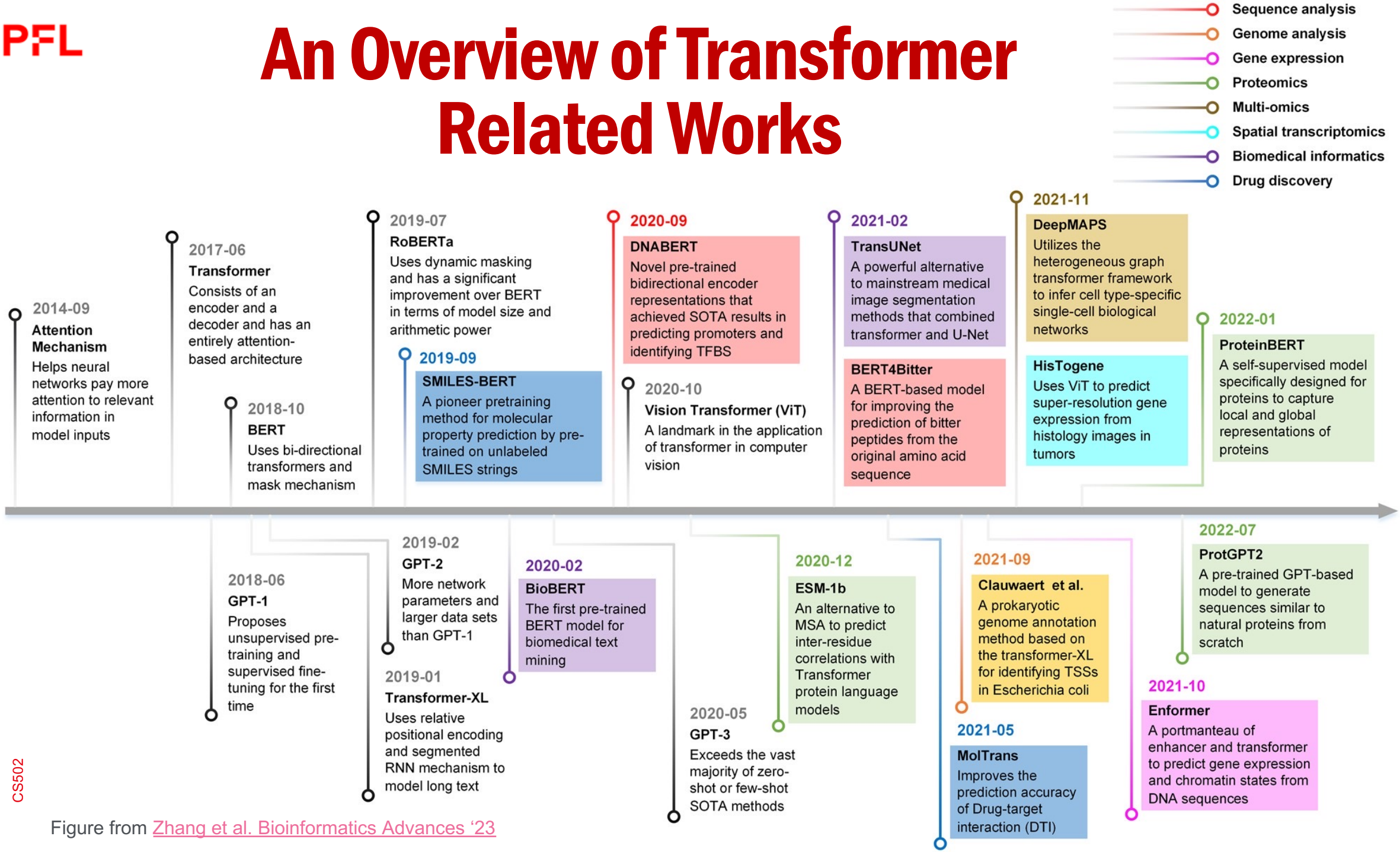
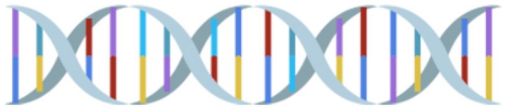


Figure from Zhang et al. Bioinformatics Advances '23

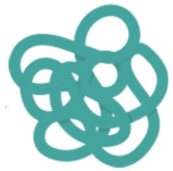
Transformer-Based Models for Biological Sequences: Examples

- **DNA sequences**
  - DNABert (Ji et al., 2021)
  - Enformer (Avsec et al., 2021)
  - HyenaDNA (Nguyen et al., 2023)
  - DNAGPT (Zhang et al., 2023)



- **RNA sequences:**
  - GeneFormer (Theodoris et al., 2023)
  - scGPT (Cui et al., 2023)
  - scFoundation (Hao et al., 2023)

- **Protein sequences:**
  - ESM1b (Rives et al., 2021)
  - ESM2 (Lin et al., 2023)
  - ProGen (Madani et al., 2023)



DNABERT

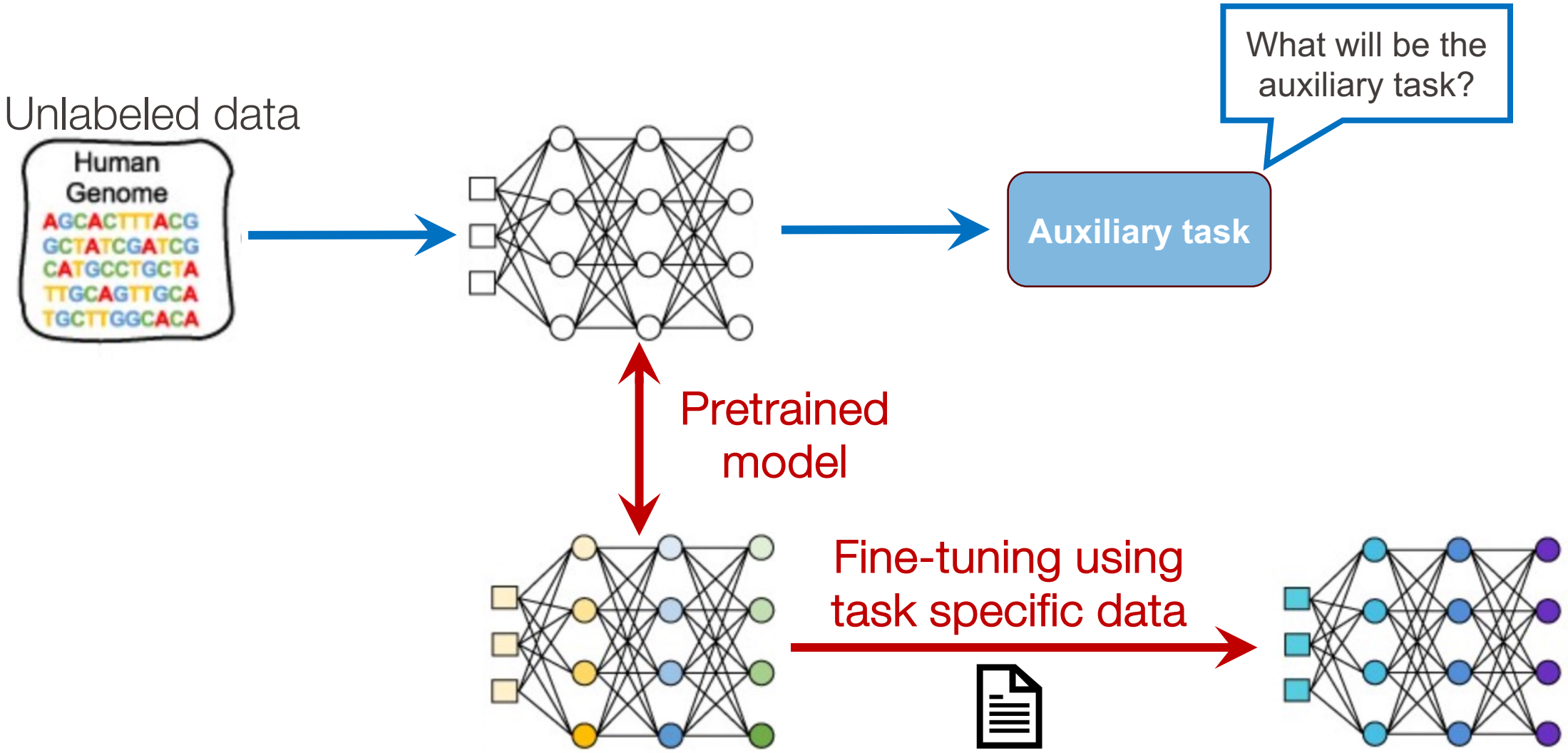
Ji, Zhou, Liu, Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics* 2021

DNABERT: Pretrain & Fine-tune

- Pre-trained bidirectional encoder representation for understanding of genomic DNA sequences
- Based on masked language model BERT (Devlin et al., 2018)
- Encoder-only transformer

- **Approach:**
  - Pretrain the model using large amount of available data on the auxiliary task → self-supervised learning
  - Fine-tune the model on task-specific data

DNABERT: Pretrain & Fine-tune



DNABERT: Pretraining

Masked token prediction!

- But, let's first construct tokens!
  - In NLP, tokens are words what could be tokens in DNA sequences?

N-grams!

AGCACTGCTATCATGCTTGCAG

tokenize

AGC GCA CAC ACT … CTT TTG TGC GCA CAG

DNABERT: Pretraining

Special token representing entire sequence

Special token representing sequence separator

CLS AGC GCA CAC ACT … CTT TTG TGC GCA CAG SEP

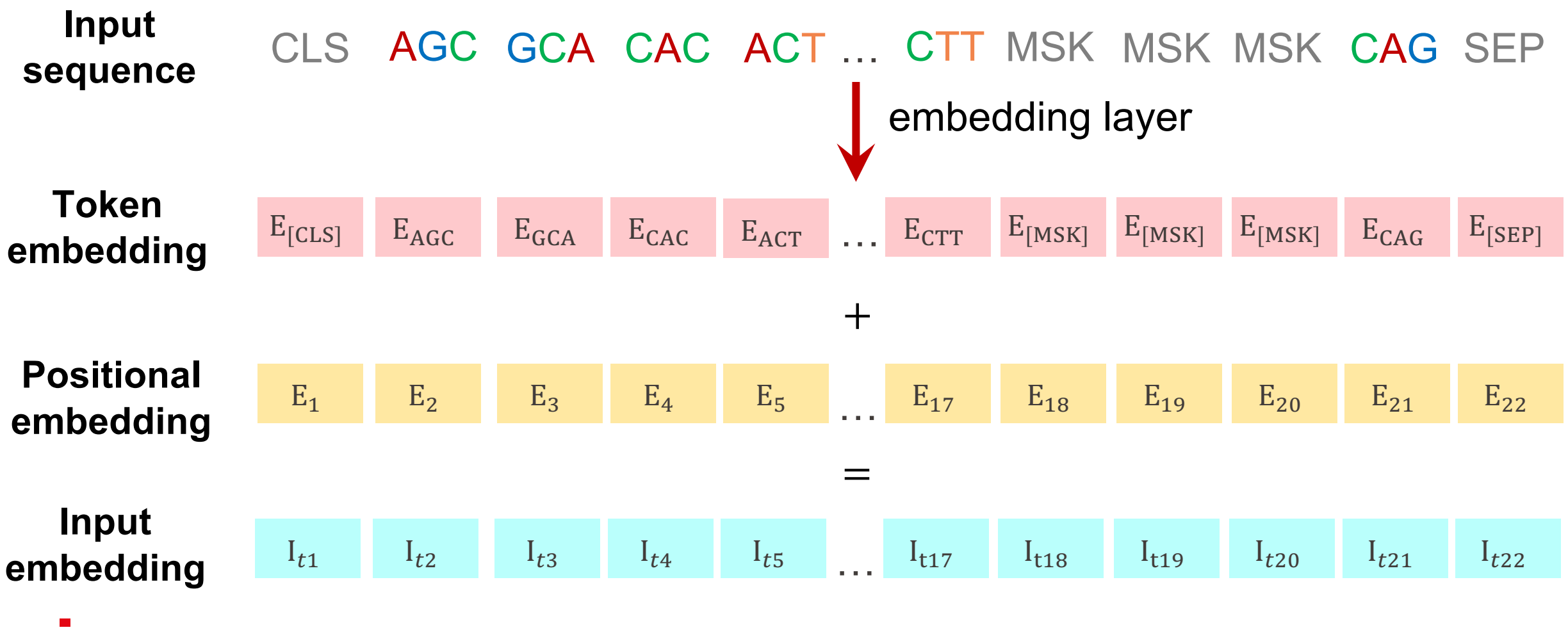
mask tokens
(only during pretraining)

CLS AGC GCA CAC ACT … CTT MSK MSK MSK CAG SEP

Special token representing masked token

- Randomly mask tokens
  - DNABERT masks 15% tokens
  - In DNA, we need to mask contiguous k-length spans of k-mers

DNABERT: Pretraining

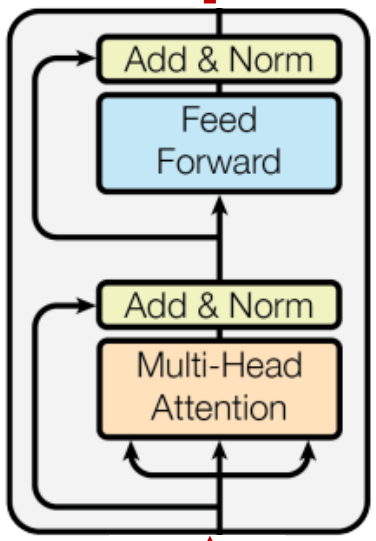


DNABERT: Pretraining

Last hidden state

O_{t1} O_{t2} O_{t3} O_{t4} O_{t5} … O_{t17} O_{t18} O_{t19} O_{t20} O_{t21} O_{t22}

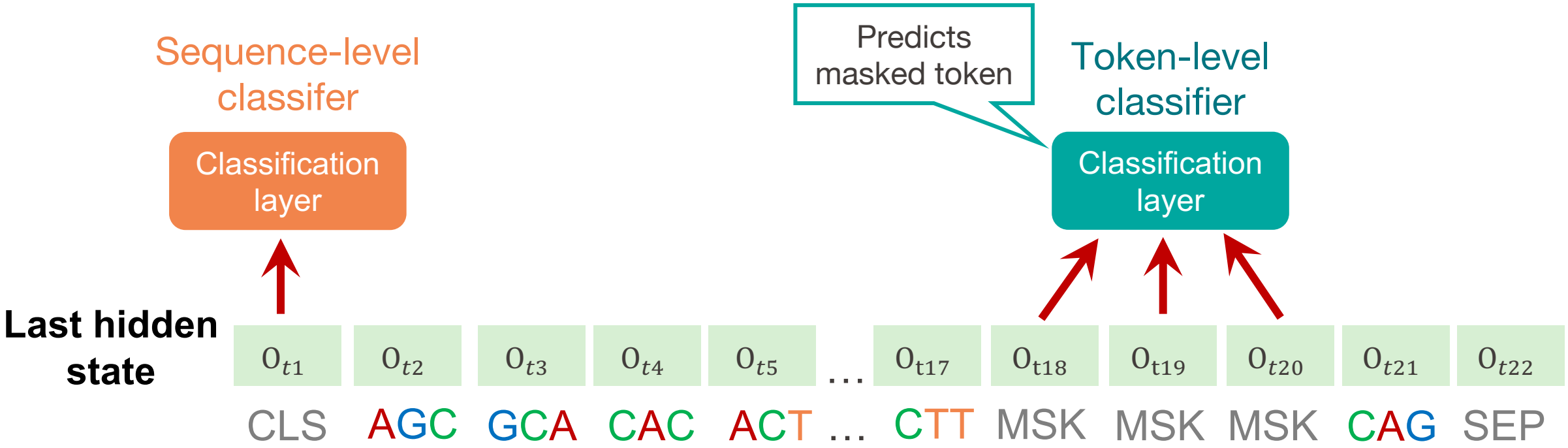
12 ×



Input embedding

I_{t1} I_{t2} I_{t3} I_{t4} I_{t5} … I_{t17} I_{t18} I_{t19} I_{t20} I_{t21} I_{t22}

DNABERT: Pretraining



- Sequence-level classifier could be added but not used in DNABERT
  - Given a pair of sequences, predicts whether one follows after the other one

DNABERT: Pretraining

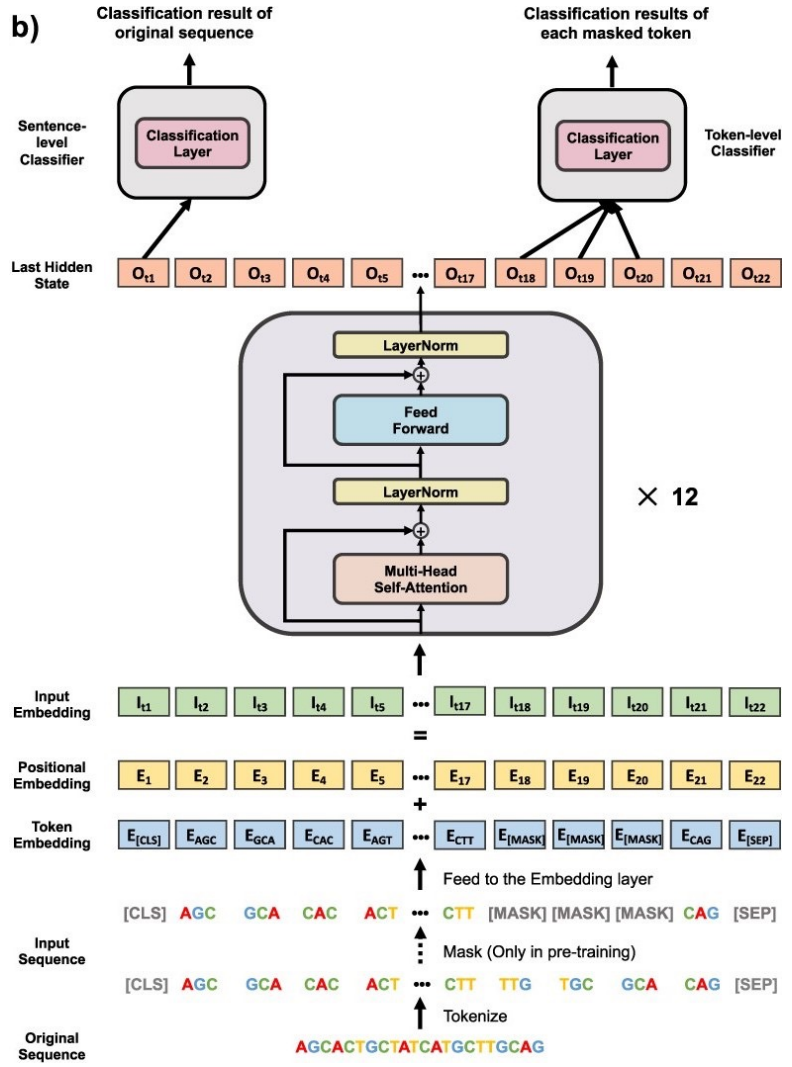
- Data: reference human genome
- For each sequence, randomly mask regions of k contiguous tokens that constitute 15% of the sequence
- DNABERT predicts the masked sequences based on the remainder tokens
- Cross-entropy loss function: $\mathcal{L} = \sum_{i=0}^{N} -y_i' \log(y_i)$

  y_i': ground truth

  y_i: predicted probability for a class i
- Different values of k for k-mers:
  - Experiments with $k = 3 \ldots 6$

DNABERT: Architecture Recap

- DNABERT architecture is composed of multiple sequential blocks:
  - Input embedding (including positional embedding)
  - Transformer encoder layers (x12)
  - Classifier (one fully connected layer)



DNABERT: Fine-tuning Parameters optimization

- During DNABERT pretraining:
  - Optimization of all layers
- During DNABERT fine-tuning:
  - Only optimization of the final layer, the classifier (fully connected layer)
  - “Freezing” of the other layers (keep their parameters fixed)

DNABERT: Fine-tuning Applications

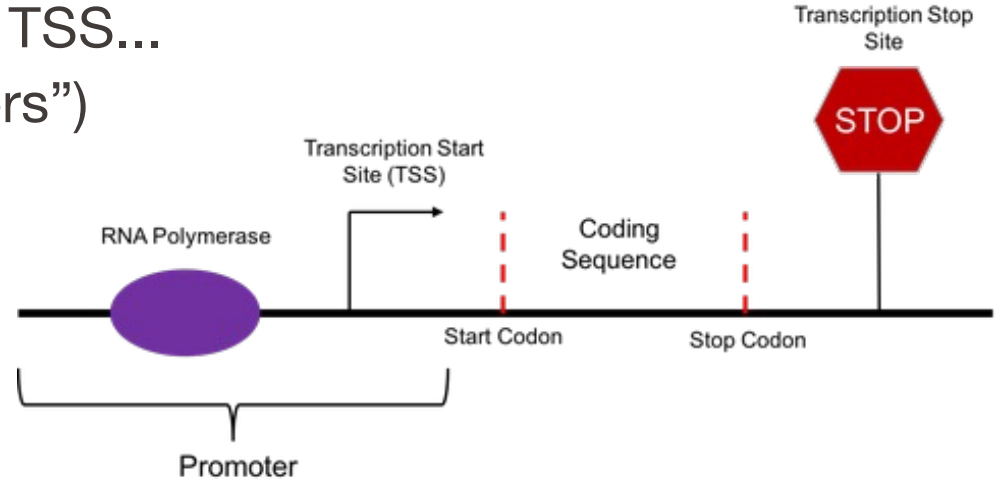
- DNABERT is able to solve different types of tasks:
  - **Token-level tasks**
    - Example: seen during pretraining
  - **Sequence-level tasks**
    - Example: classifying a sequence as positive/negative (detection)

DNABERT: Fine-tuning Applications

- DNABERT is fine-tuned on three tasks (sequence-level):
  1. Prediction of promoters
  2. Prediction of transcription factor binding sites
  3. Prediction of splice sites

DNABERT: Fine-tuning 1 – Promoters (biology)

- **Promoter:** DNA sequence indicating where the **transcription** should start
  - Found before the sequence to transcript (coding sequence)
  - Indicates where the “cell machinery” (polymerase) will initiate the transcription
- **Promoter examples**
  - RNA polymerase binding site, TATA box, TSS...
  - Length: 100-1000 DNA bases (“characters”)
- **Predicting promoters**
  - Challenging bioinformatics problem !

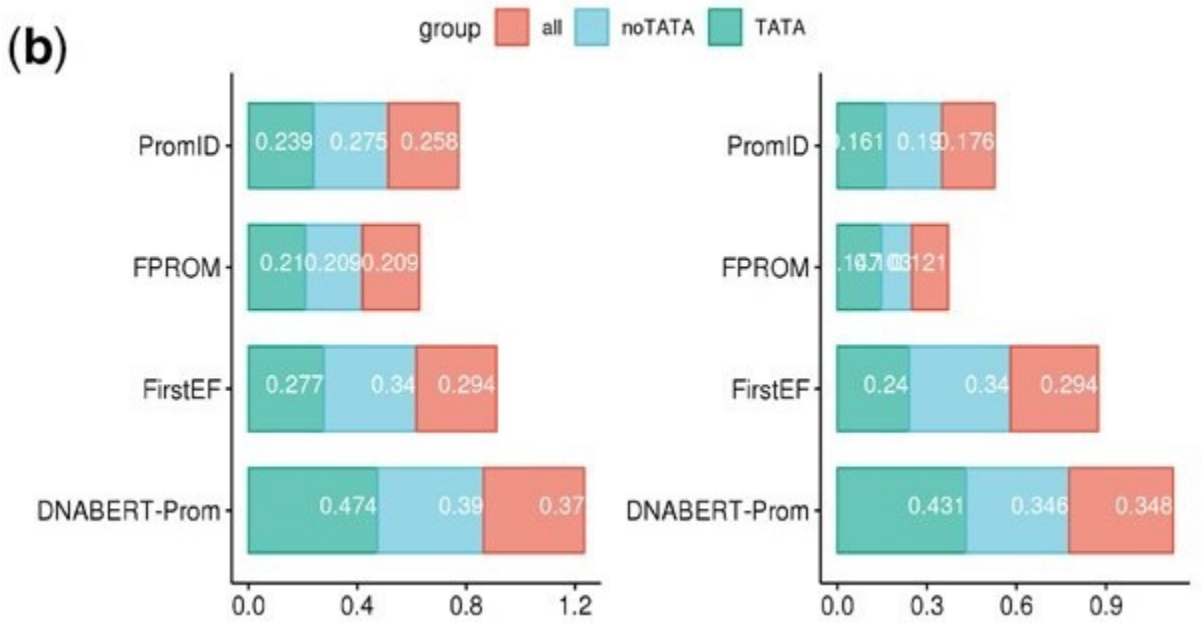


DNABERT: Fine-tuning 1 – Predicting promoters

- Binary classification of sequences (detection)
- Specific data used to fine-tune DNABERT
  - Positive samples: promoter sequences (from Promoter Database)
  - Negative samples: random sequences outside promoter regions
    - Random sequences are not enough
    - Use of random sequences containing similar motifs (TATA)
  - Selecting “difficult” negative sequences helps DNABERT to learn less obvious features to discriminate positive/negative samples

DNABERT: Fine-tuning
1 – Promoter prediction results

- DNABERT significantly outperforms other models in identifying promoter regions

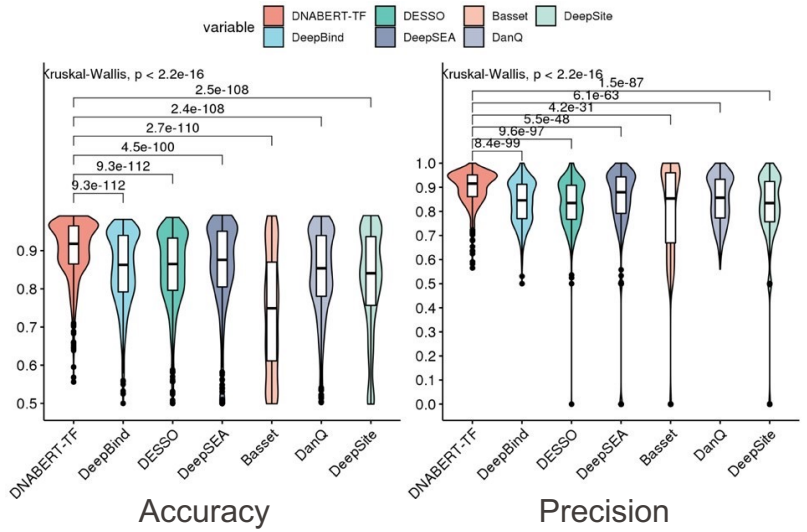


F1-score

MCC
(Matthews correlation coefficient)

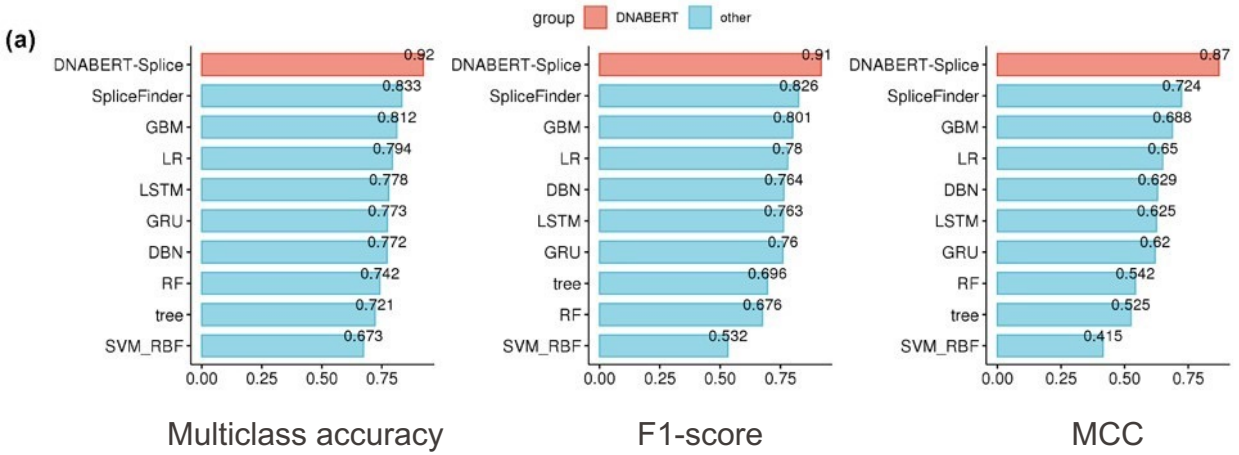
DNABERT: Fine-tuning Others finetuned models

- **Same idea as promoter prediction**
  - Sequence-level tasks
  - Binary classification (positive/negative) to detect specific sites
  - Different specific data for transcription factor binding sites & splice sites
  - **DNABERT is also outperforming other existing models !**



Accuracy Precision

2. Prediction of transcription factor binding sites

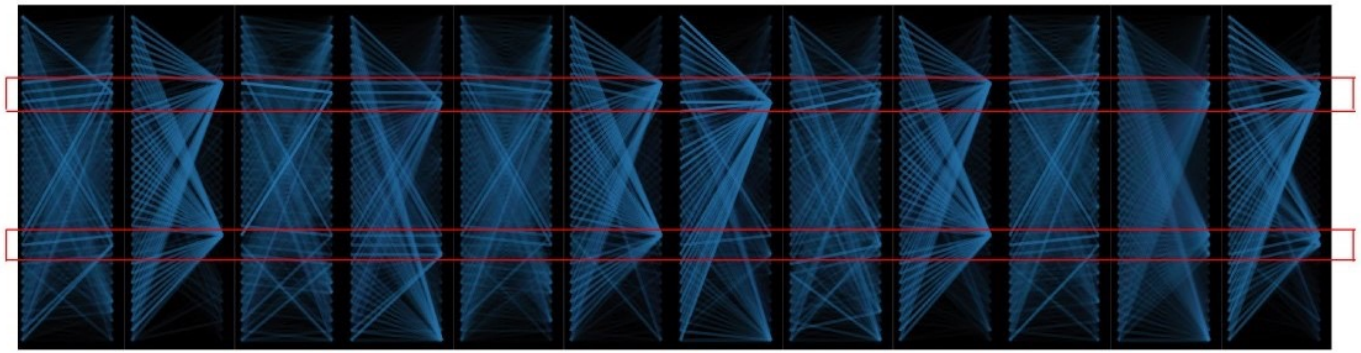


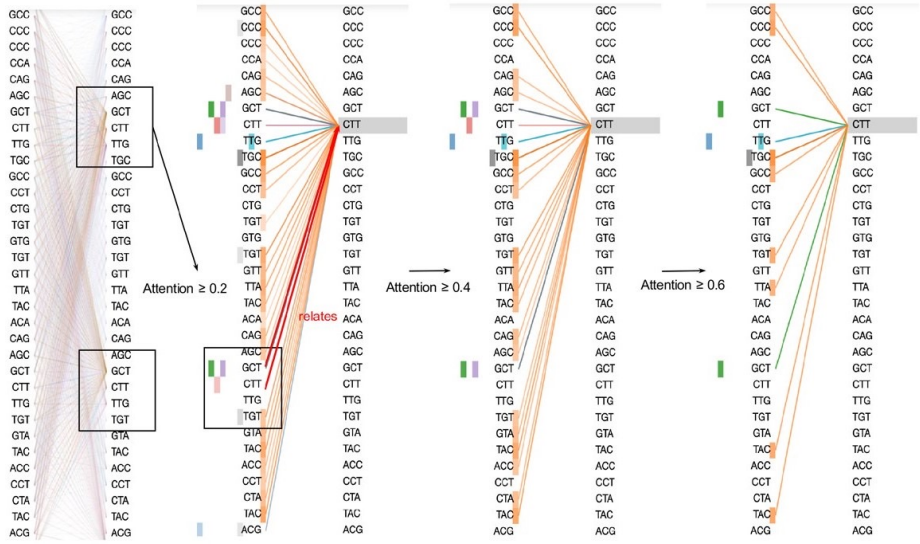
Multiclass accuracy F1-score MCC

3. Prediction of splice sites

DNABERT

- **Interested in more details ?**
  - Check the publication: https://academic.oup.com/bioinformatics/article/37/15/2112/6128680





Recap

- Architectures of encoding sequences:
  - Historically: RNNs, LSTMs (special type of RNNs)
    - ➢Process sequence sequentially
  - Nowadays: Transformers
    - ➢Process sequence in parallel instead of sequentially
- Transformers key concepts:
  - Self-attention mechanism
  - Multi-head attention
  - Positional encoding
- Sequences in biology
  - DNA → RNA → Protein
- Biomedical applications:
  - DNABert

Additional Readings

- Hochreiter & Smidhuber. Long Short-Term Memory. *Neural Computation* (1997)
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Vaswani et al. Attention Is All You Need. *NeurIPS* (2017)
- https://jalammar.github.io/illustrated-transformer/
- Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ACL* (2019)
- Ji et al. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics* (2021)

Any Feedback?

Give us feedback on the lecture:

- https://go.epfl.ch/cs502-lecture-5-feedback