



Sequence Modelling and Generation with Transformer

Lei Li



Language
Technologies
Institute

Carnegie Mellon University
School of Computer Science

Paper Presentation Assignment

- Please prepare slides and upload to canvas **7 days before** your presentation day.
- Please use the template provided and follow the guideline.

Can GenAI design molecules with desired functions?

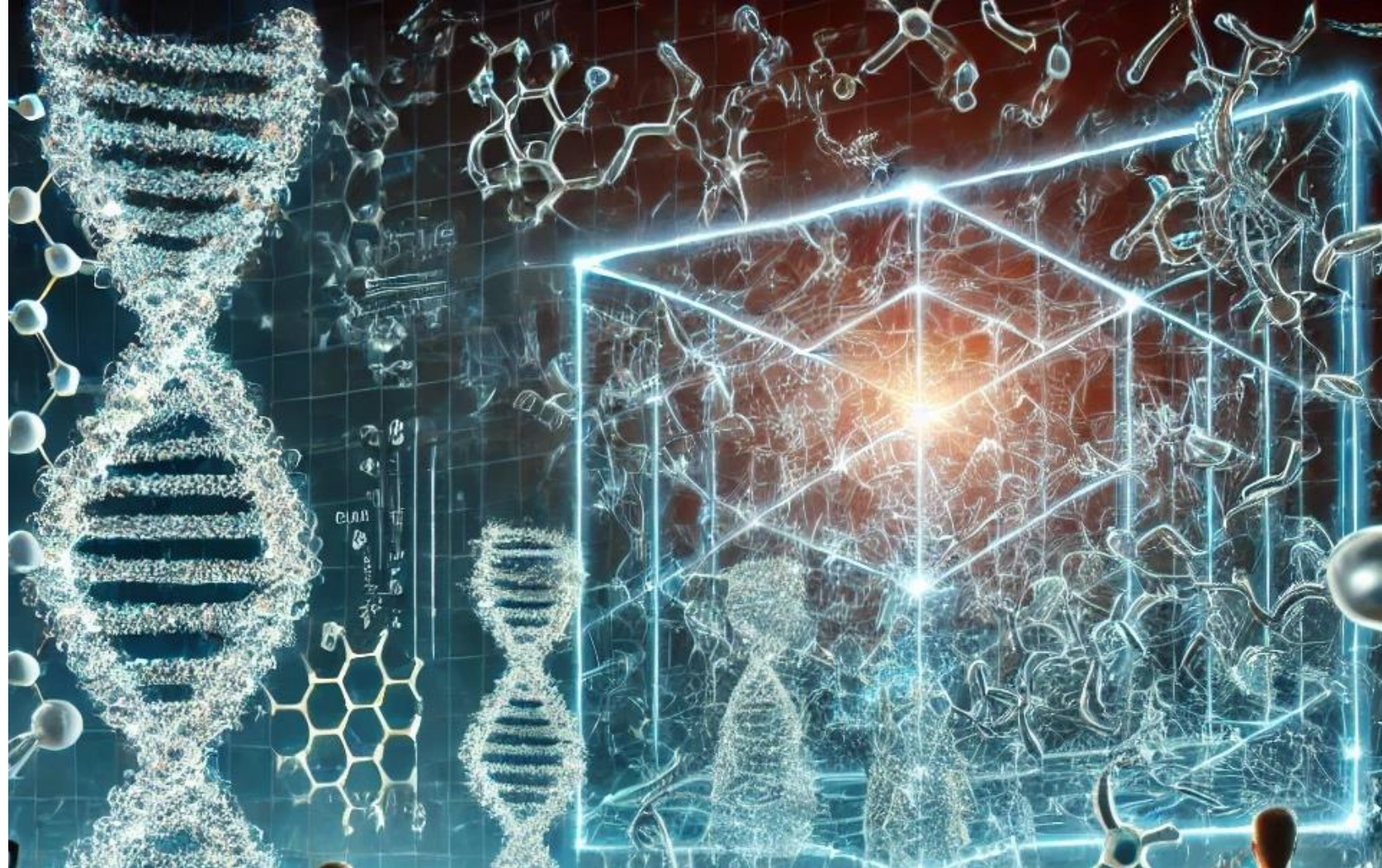
Medicine

Vaccine

Enzyme - Biocatalysts

Biosensors (e.g. GFP)

New materials



Commonality and Distinction in Language and Molecule Generation

- Both are sequences of Discrete Tokens
- Both have discrete Structures
- Geometry (Unique for molecules)

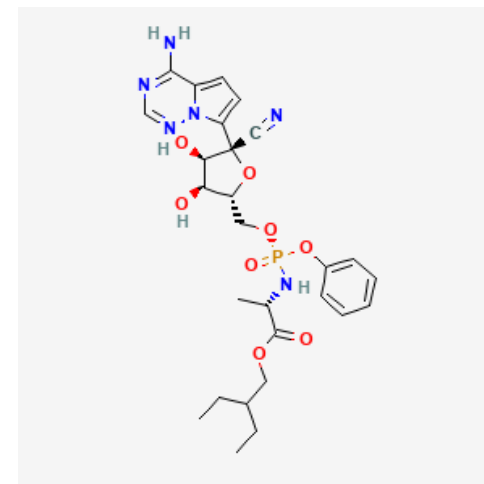
Discrete Sequences of Tokens

It was the best of times, it
was the worst of times, it was
the age of wisdom, it was the
age of foolishness, it was the
epoch of belief, it was the
epoch of incredulity, it was
the season of Light, it was the
season of Darkness, ...

Remdesivir: $C_{27}H_{35}N_6O_8P$

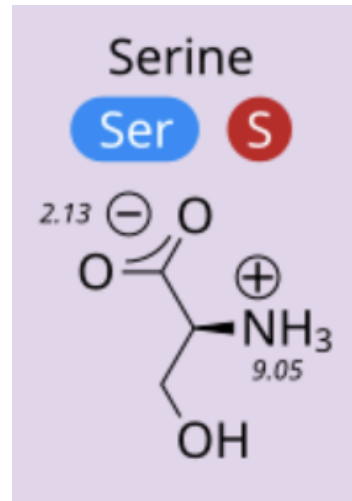
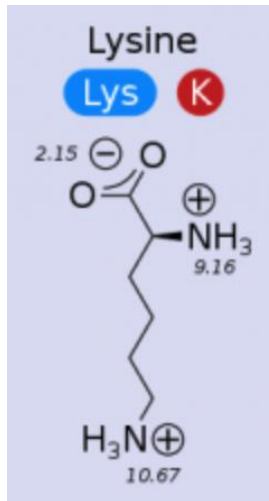
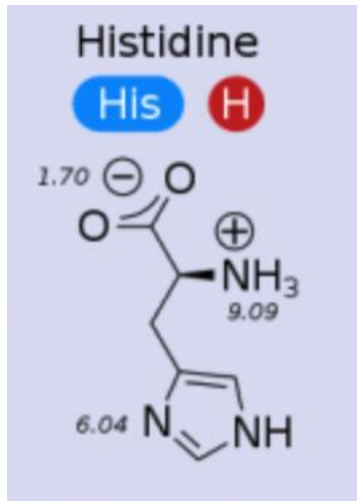
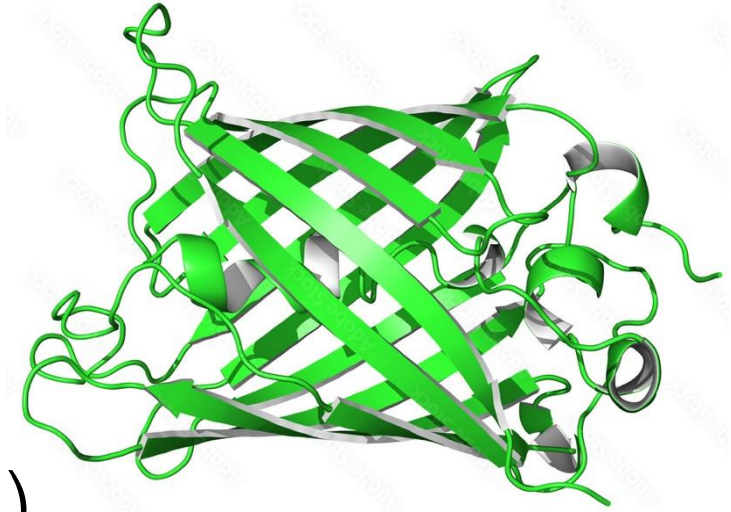
SMILES representation:

```
CCC(CC)COC(=O)C(C)NP(=O)(OCC1C(C(C(O1)(C#N)C2=CC=C3N2N=CN=C3N)O)O)OC4=CC=CC=C4
```



From Human Language to Protein Sequence

- Proteins are building blocks of life
- Important biological functions
- sequence of amino acid residues (20 types)



VLLPDNHYLSTQSAISKDPNE
KRDHMLLEFVTAAGIT

Today's Topic



- Language Models
- Transformer Model
 - Embedding
 - Multihead Attention, Decoder Self-Attention
 - FFN
 - Layernorm
- Training Techniques and Performance of Transformer
- Code walkthrough

Language Model

- A probabilistic model of discrete sequences, including human languages and biological languages

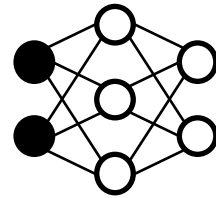
Probability("Pittsburgh is a city of bridges")

Probability("VLLPDNHYLSTQSALSKDPN")

Probability Model for Next Token

$P(\text{next word } y_t \mid \text{Prompt } x, \text{previous words } y_{1:t-1})$

Santa Barbara has very nice _____



beach	0.5
weather	0.4
snow	0.01

Pittsburgh is a city of _____

bridges	0.6
corn	0.02

Mathematics of Language Model

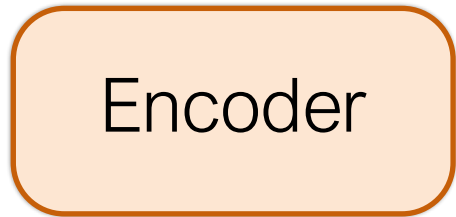
Probability("Pittsburgh is a city of bridges")
= $P(\text{"Pittsburgh"}) \cdot P(\text{"is"} | \text{"Pittsburgh"})$
· $P(\text{"a"} | \text{"Pittsburgh is"}) \cdot P(\text{"city"} | \dots) \cdot P(\text{"of"} | \dots)$
· $P(\text{"bridges"} | \dots)$

$$\text{Prob.}(x_{1..T}) = \prod_{t=1}^T \underbrace{P(x_t | x_{1..t-1})}_{\text{Predicting using Neural Nets}}$$

Predicting using Neural Nets
(Transformer network, CNN, RNN)

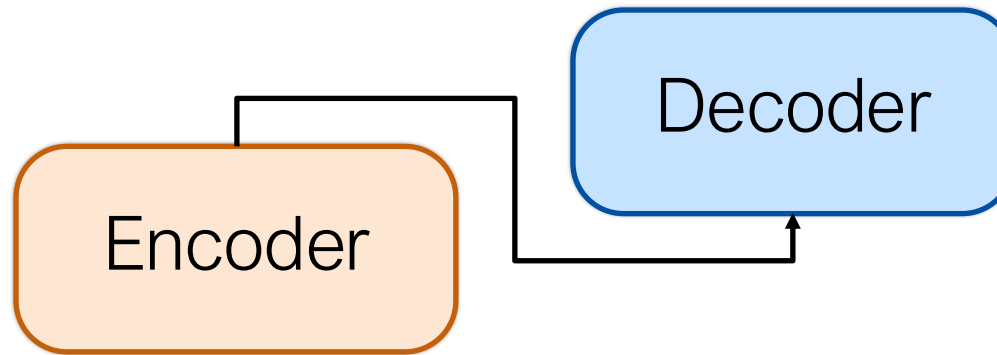
Type of Language Models

Encoder-only
Masked LM
Non-autoregressive



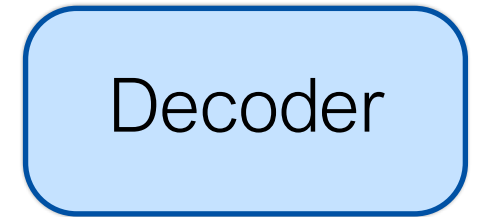
e.g. BERT
RoBERTa
ESM (for protein)

Encoder-decoder



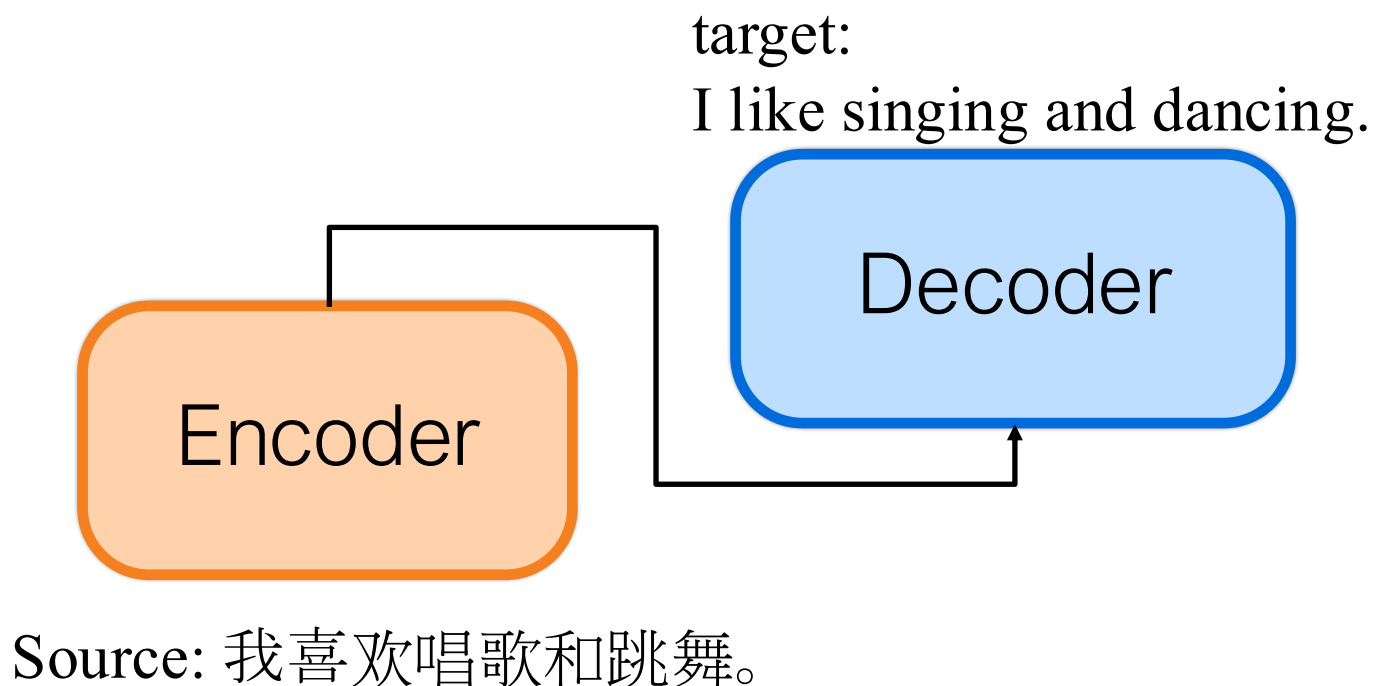
e.g. T5

Decoder-only
Autoregressive



e.g. GPT
LLaMA
ProGen (for protein)

Encoder-Decoder Paradigm



$$p_{\theta}(y|x) = \prod_i \underline{p(y_i|x, y_{1:i-1})}$$

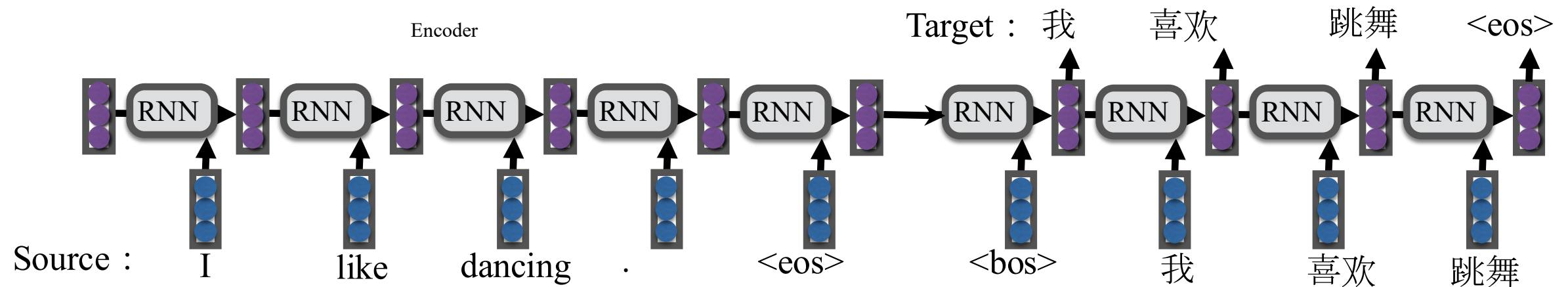
conditional prob. modeled by
neural networks (Transformer)

Sequence to Sequence Learning

- Conditional text generation: directly learning a function mapping from source sequence to target sequence

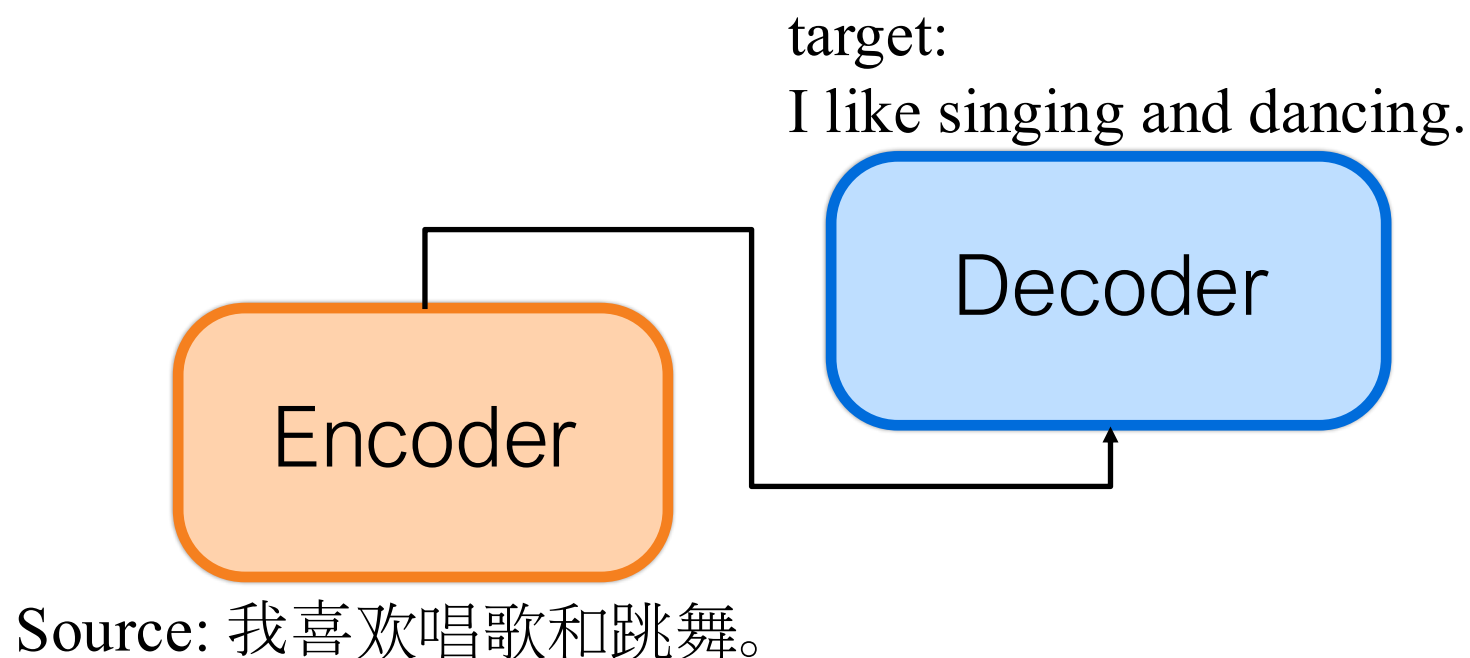
$$p_{\theta}(y|x) = \prod_t p(y_t|x, y_{1:t-1}; \theta)$$

- Previous encoder/decoder: LSTM or GRU



Motivation for a new Architecture

- Full context and parallel: use Attention in both encoder and decoder
- no recurrent ==> concurrent encoding



Motivation of Attention: translation need different alignment of words

Je veux essayer un costume que j'ai vu dans une boutique en face de l'hôtel

I want to try on a suit that I saw in a shop that's across the street from the hotel

私は ホテルの 向かいにある お店で 見た スーツを 着て みたいです。

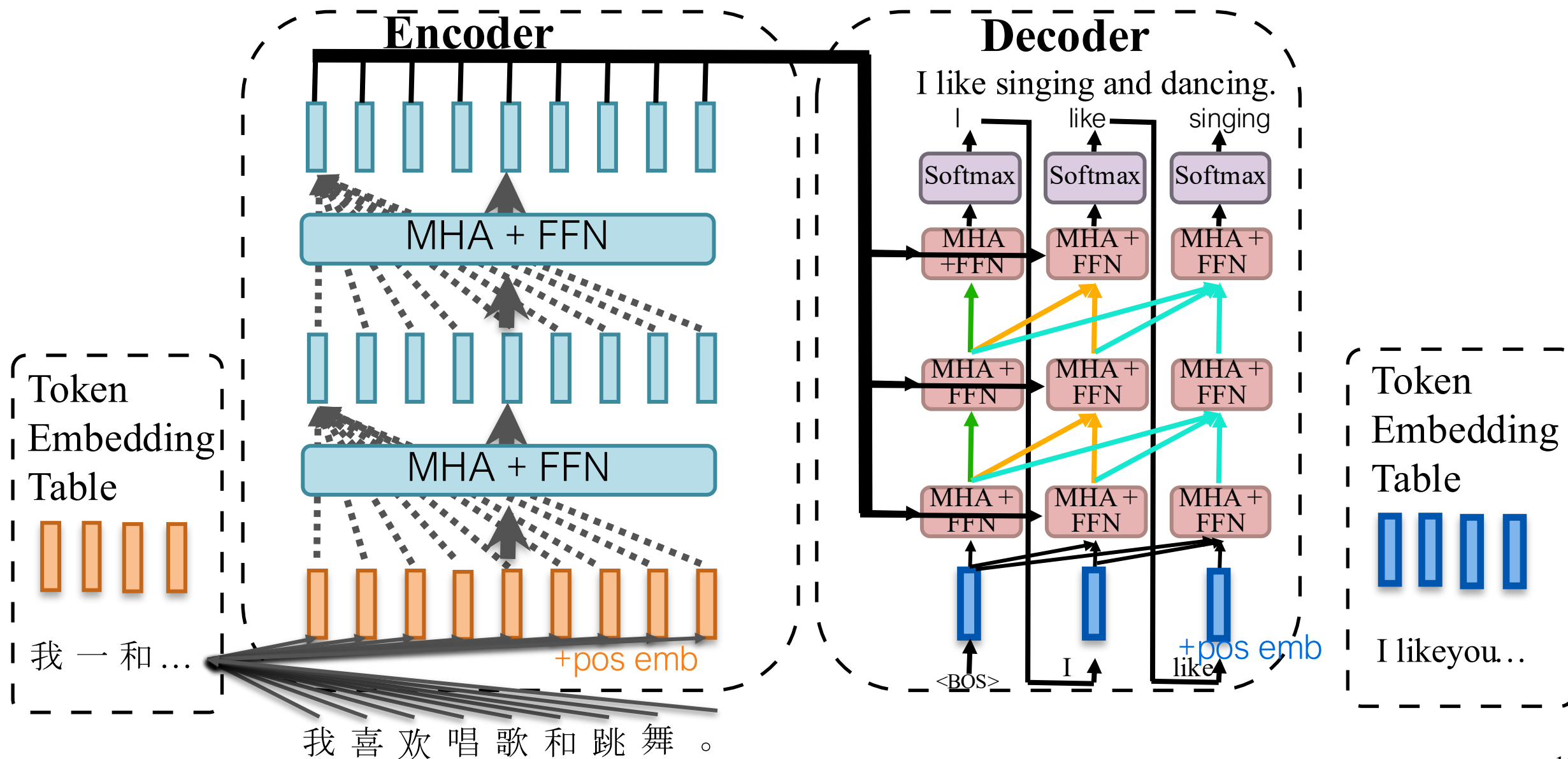
Transformer-based Models for Biomolecule

- Protein Language Models:
 - ESM, ESM-Fold
 - ProGen
 - AlphaFold (with EvoFormer)
- Genomics
 - DNABERT / DNABERT2
 - Evo, Evo2
 - RNABERT
- Single Cell: scGPT, scFoundation

Today's Topic

- Encoder-decoder Architecture
- • Transformer Model
 - Embedding
 - Multihead Attention, Decoder Self-Attention
 - FFN and SwiGLU
 - Layernorm
 - RoPE
- Training Techniques and Performance of Transformer
- Code walkthrough

Transformer



Embedding

- Token Embedding:
 - token index $\rightarrow$ vector
 - Shared (tied) input and output embedding from lookup table
- Positional Embedding:
 - to distinguish words in different position, Map position labels to embedding, dimension is same as Tok Emb, for t-th pos, i-th dim

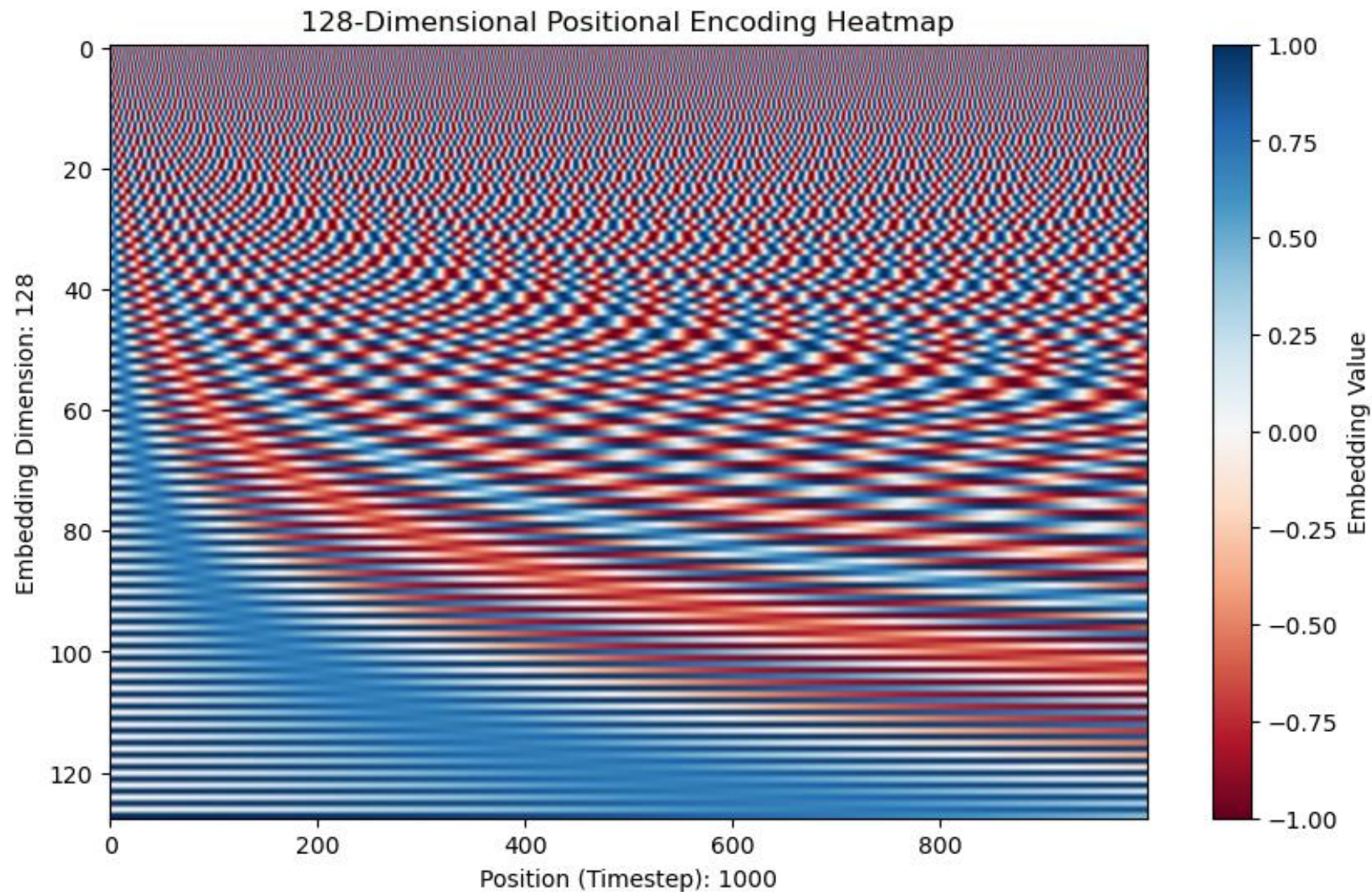
$$PE_{t,2i} = \sin\left(\frac{t}{1000^{2i/d}}\right)$$

$$PE_{t,2i+1} = \cos\left(\frac{t}{1000^{2i/d}}\right)$$

Sinusoidal Positional Embedding

for t-th pos, i-th dim

$$PE_{t,2i} = \sin\left(\frac{t}{1000^{2i/d}}\right)$$
$$PE_{t,2i+1} = \cos\left(\frac{t}{1000^{2i/d}}\right)$$



Multi-head Attention

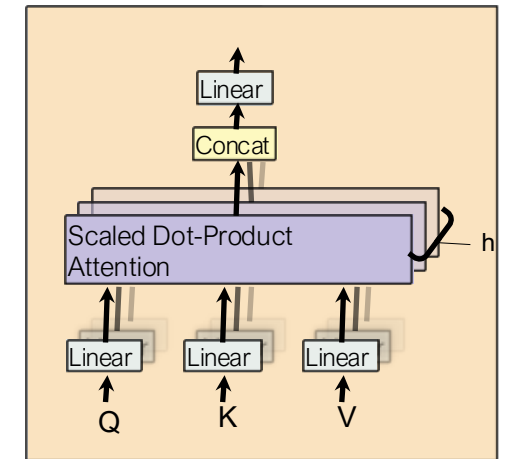
- Instead of one vector for each token
- break into multiple heads

- each head perform attention

$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V)$$

$$= \text{Concat}(\text{Head}_1, \text{Head}_2, \dots, \text{Head}_h)W^O$$



Multi-head Attention

X are input embeddings from previous layer (num of tok * dim)

Query

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline \text{green} & \text{green} & \text{green} & \text{green} \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{W}^Q \\ \begin{array}{|c|c|c|c|} \hline \text{purple} & \text{purple} & \text{purple} & \text{purple} \\ \hline \end{array} \end{matrix} = \begin{matrix} \text{Q} \\ \begin{array}{|c|c|c|} \hline \text{purple} & \text{purple} & \text{purple} \\ \hline \end{array} \end{matrix}$$

Key

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline \text{green} & \text{green} & \text{green} & \text{green} \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{W}^K \\ \begin{array}{|c|c|c|c|} \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \end{array} \end{matrix} = \begin{matrix} \text{K} \\ \begin{array}{|c|c|c|} \hline \text{orange} & \text{orange} & \text{orange} \\ \hline \end{array} \end{matrix}$$

Value

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline \text{green} & \text{green} & \text{green} & \text{green} \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{W}^V \\ \begin{array}{|c|c|c|c|} \hline \text{blue} & \text{blue} & \text{blue} & \text{blue} \\ \hline \end{array} \end{matrix} = \begin{matrix} \text{V} \\ \begin{array}{|c|c|c|} \hline \text{blue} & \text{blue} & \text{blue} \\ \hline \end{array} \end{matrix}$$

sent len x sent len

$$\text{softmax} \left(\frac{\begin{matrix} \text{Q} \\ \begin{array}{|c|c|c|} \hline \text{purple} & \text{purple} & \text{purple} \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{array}{|c|c|c|} \hline \text{orange} & \text{orange} & \text{orange} \\ \hline \end{array} \end{matrix}}{\sqrt{d_k}} \right) \begin{matrix} \text{V} \\ \begin{array}{|c|c|c|} \hline \text{blue} & \text{blue} & \text{blue} \\ \hline \end{array} \end{matrix}$$

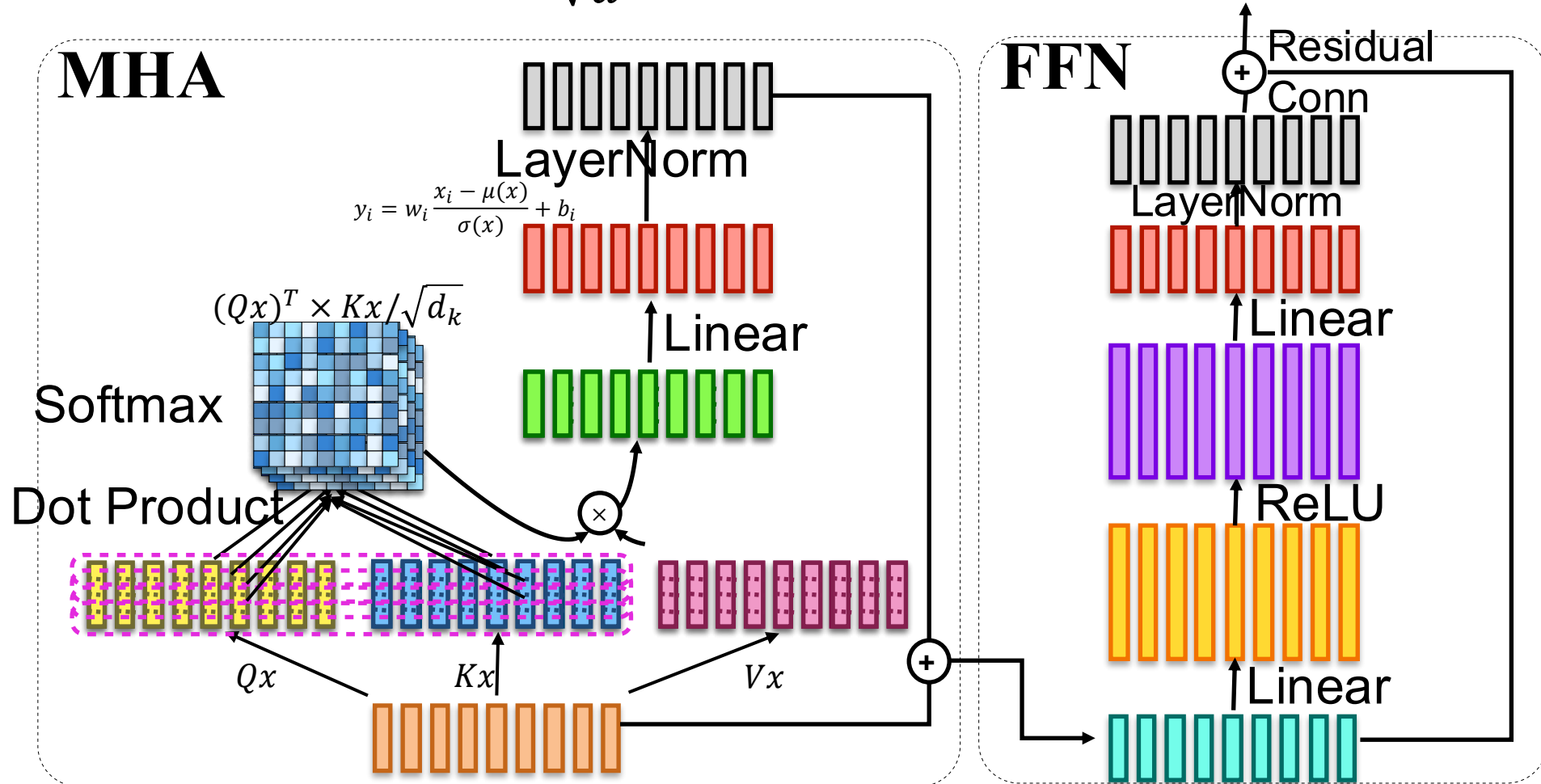
$$= \begin{matrix} \text{Z} \\ \begin{array}{|c|c|c|} \hline \text{pink} & \text{pink} & \text{pink} \\ \hline \end{array} \end{matrix}$$

len x dim

Q: why divided by sqrt(d)?

Multihead Attention and FFN

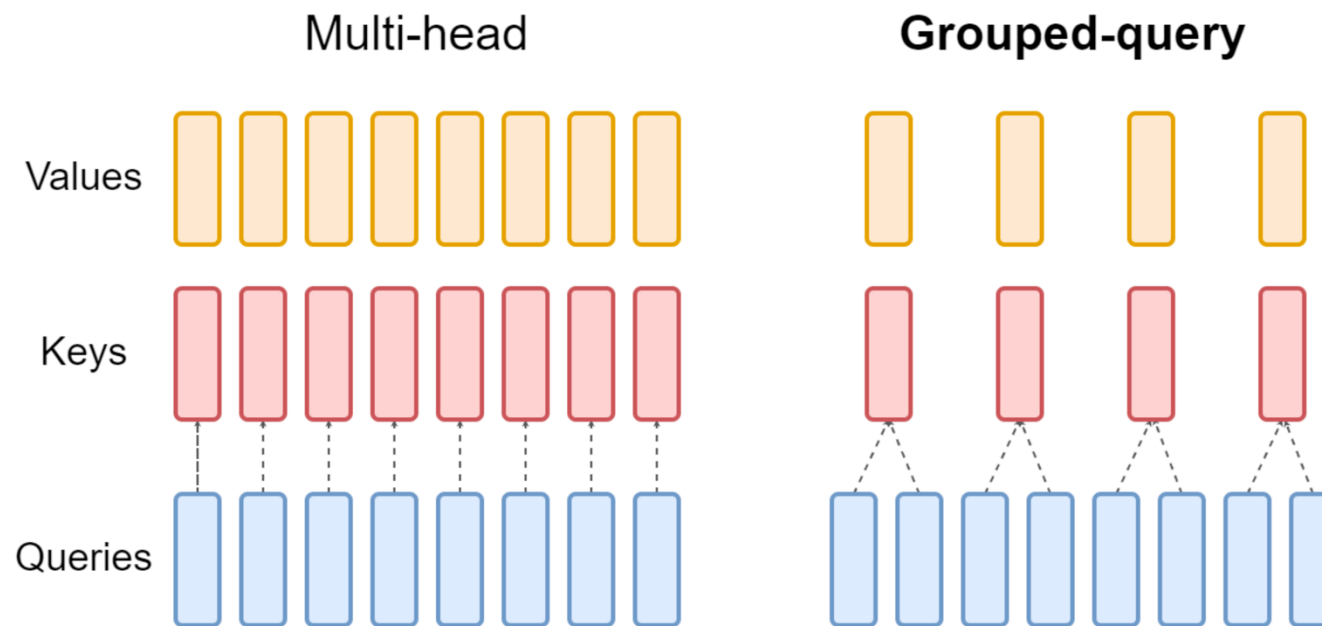
$$\text{Attention}(Q, K, V, x) = \text{Softmax}\left(\frac{(Qx)^T Kx}{\sqrt{d}}\right) \cdot (Vx)^T \quad \text{FFN}(x) = \max(0, x \cdot W_1 + b_1) \cdot W_2 + b_2$$



Attention Variant: Grouped-Query Attention (GQA)

$$\text{Head}_i = \text{Attention}(XW_h^Q, XW_{g(h)}^K, XW_{g(h)}^V)$$

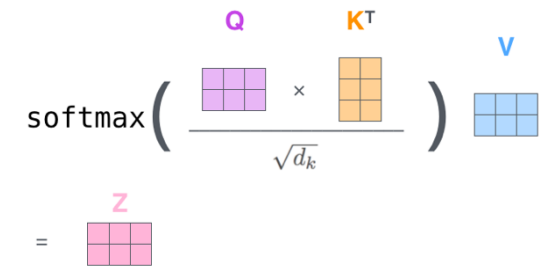
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{Head}_1, \text{Head}_2, \dots, \text{Head}_h)W^O$$



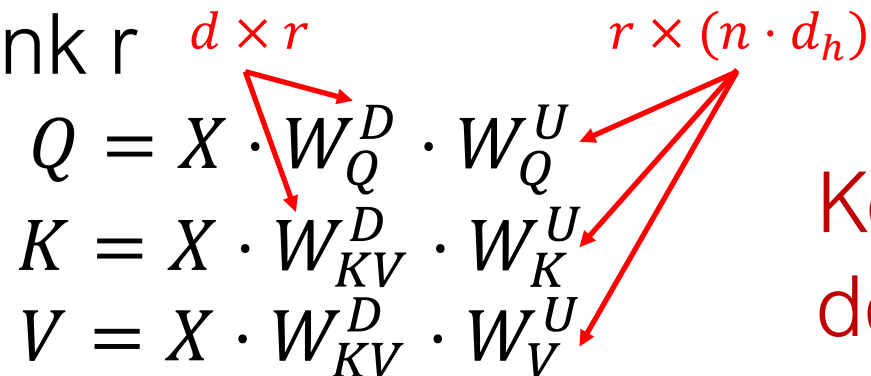
Attention Variant: Multi-head Latent Attention (MLA)

- Keys and values occupy large GPU memory (re-visit later in PagedAttention and KV cache lecture)

- $2 \cdot len \cdot n_h \cdot d_h$


$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \cdot V = Z$$

- instead of caching d_h vector (for each tk, head), reduce to a latent space with rank r

$$\begin{aligned} Q &= X \cdot W_Q^D \cdot W_Q^U \\ K &= X \cdot W_{KV}^D \cdot W_K^U \\ V &= X \cdot W_{KV}^D \cdot W_V^U \end{aligned}$$


Key and Value share down-projection

- First proposed in Deepseek-V2, used in all modern LLMs

Multi-head Latent Attention (MLA)

$$\begin{aligned} Q &= X \cdot W_Q^D \cdot W_Q^U \\ K &= X \cdot W_{KV}^D \cdot W_K^U \\ V &= X \cdot W_{KV}^D \cdot W_V^U \end{aligned}$$

Key and Value share down-projection, and across all heads

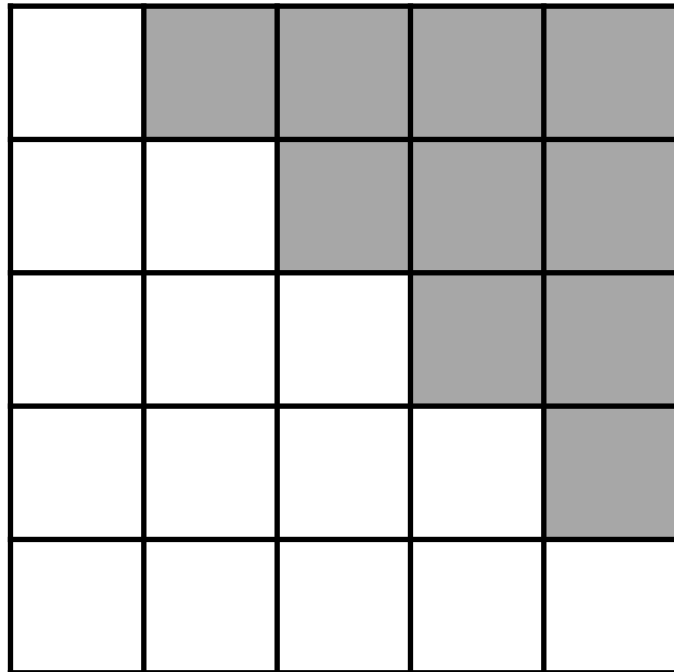
$$\begin{aligned} &MLA(x)_h \\ &= \text{Softmax} \left(\frac{X \cdot W_Q^D \cdot W_{Q,h}^U \cdot (X \cdot W_{KV}^D \cdot W_{K,h}^U)^T}{\sqrt{d_h}} \right) \cdot X \cdot W_{KV}^D \cdot W_{V,h}^U \\ &= \text{Softmax} \left(\frac{C_Q \cdot W_{QK,h} \cdot C_{KV}^T}{\sqrt{d_h}} \right) \cdot C_{KV} \cdot W_{V,h}^U \end{aligned}$$

Precomputed/cached

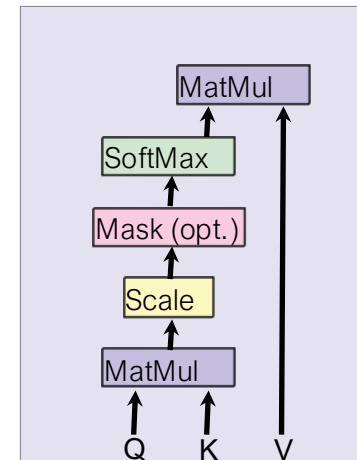
$$\begin{aligned} W_{QK,h} &= W_{Q,h}^U \cdot (W_{K,h}^U)^T \\ C_Q &= X \cdot W_Q^D \\ C_{KV} &= X \cdot W_{KV}^D \end{aligned}$$

Decoder Self-Attention Mask

- Encoder uses bidirectional attention
- Decoder uses unidirectional attention
 - Maskout right side before softmax (-inf)

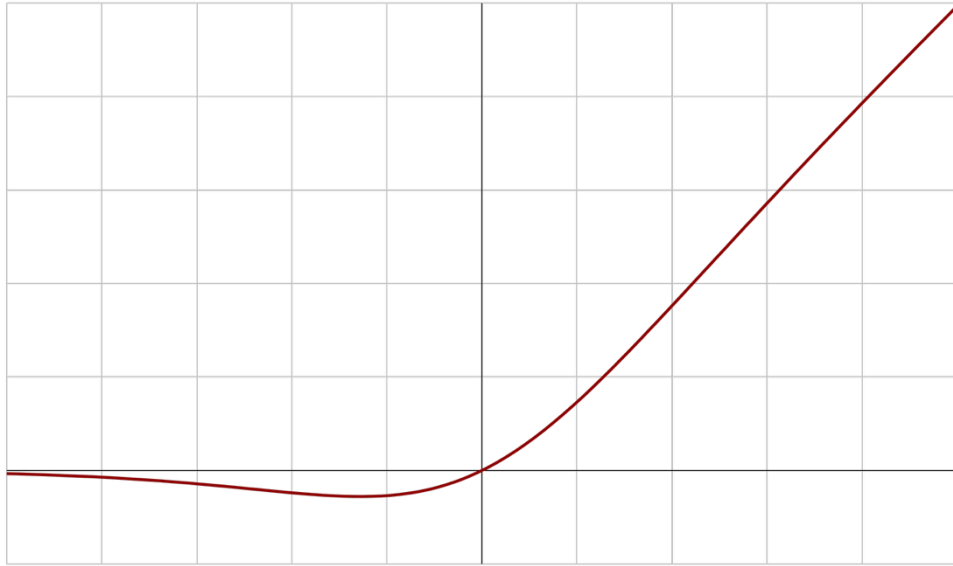


Scaled Dot-Product Attention



FFN with SwiGLU

Swish activation



$$swish(x) = x \sigma(\beta x)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

used in LLaMA and all later LLMs

FFN with ReLU

$$FFN(x) = \max(0, x \cdot W_1 + b_1) \cdot W_2 + b_2$$

dim=4d

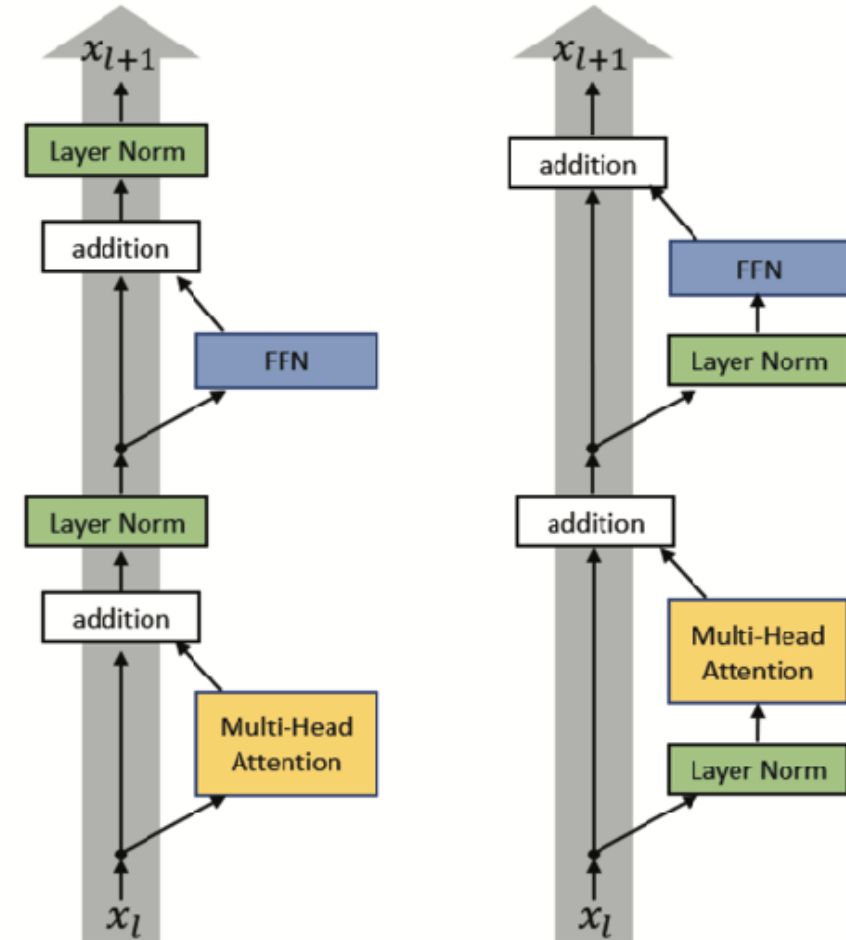
FFN with SwiGLU

$$FFN_{SwiGLU}(x) = (Swish(x \cdot W_1 + b_1) \odot (x \cdot W_2 + b_2)) \cdot W_3 + b_2$$

dim=2/3 * 4d

Residual Connection and Layer Normalization

- Residual Connection
- Make it zero mean and unit variance within layer
- Post-norm
- Pre-norm



Relative Positional Embedding

- “I have a dream”
- “Today I have a dream”
- What are Attention weights for “I” and “dream”?
- Goal: make sure the similarity is only dependent on the distance between two embeddings, independent of their absolute positions

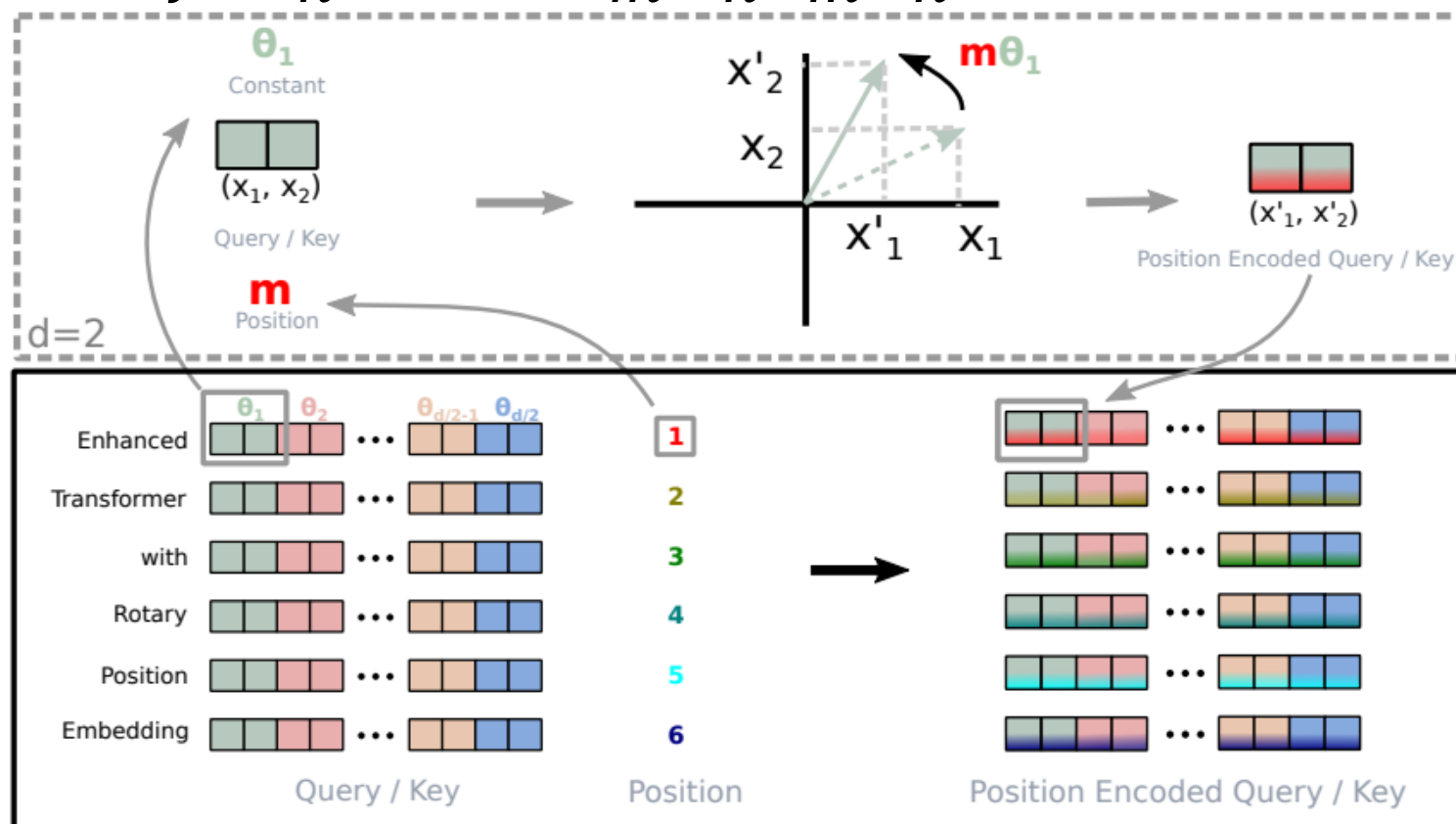
$$Att(x_i, x_j) = Att(x_{i+\Delta}, x_{j+\Delta})$$

Rotary Embedding (RoPE)

make the attention weight depending (only) on position distance

$$f(x_m, m) = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix} \begin{pmatrix} x_{m,1} \\ x_{m,2} \end{pmatrix}$$

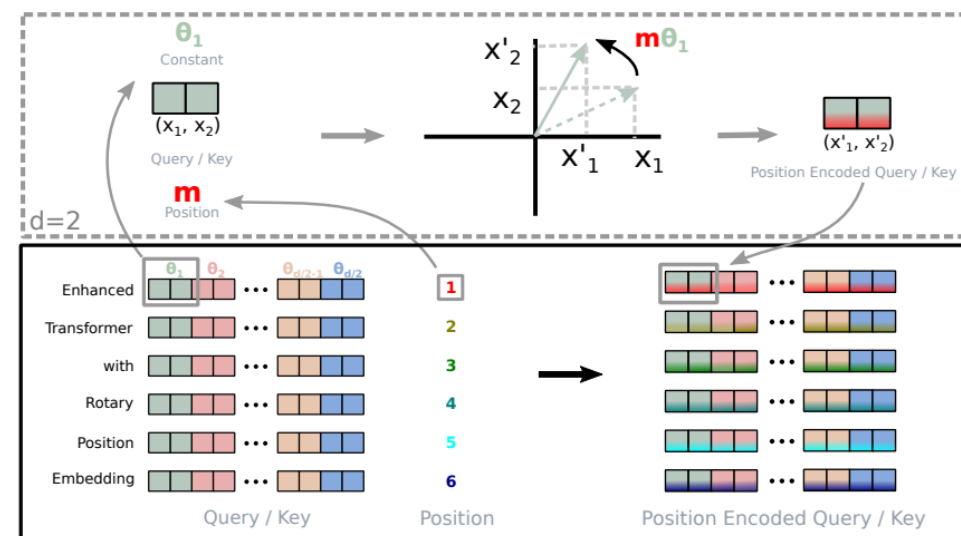
$$f(x_m, m) \cdot f(x_n, n) = x_m^T R_{n-m} x_n$$



Rotary Embedding (RoPE)

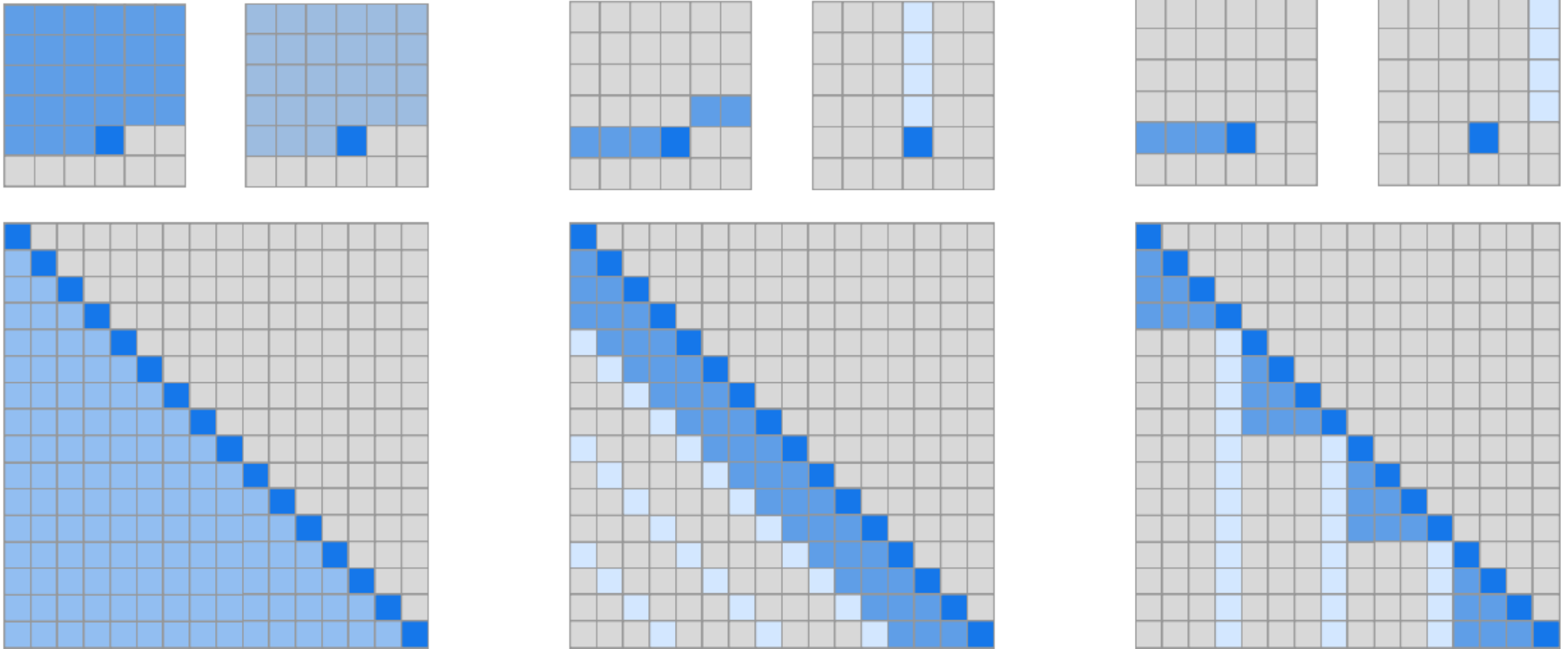
$$f(x_m, m) = \begin{pmatrix} \cos(m\theta_1) & -\sin(m\theta_1) \\ \sin(m\theta_1) & \cos(m\theta_1) \\ & \cos(m\theta_2) & -\sin(m\theta_2) \\ & \sin(m\theta_2) & \cos(m\theta_2) \end{pmatrix} \begin{pmatrix} x_{m,1} \\ x_{m,2} \\ x_{m,3} \\ x_{m,4} \end{pmatrix}$$

$$f(x_m, m)^T \cdot f(x_n, n) = x_m^T R_{n-m} x_n$$



Sparse Attention

- Alternating dense and locally banded sparse attention patterns (GPT3)



(a) Transformer

(b) Sparse Transformer (strided)

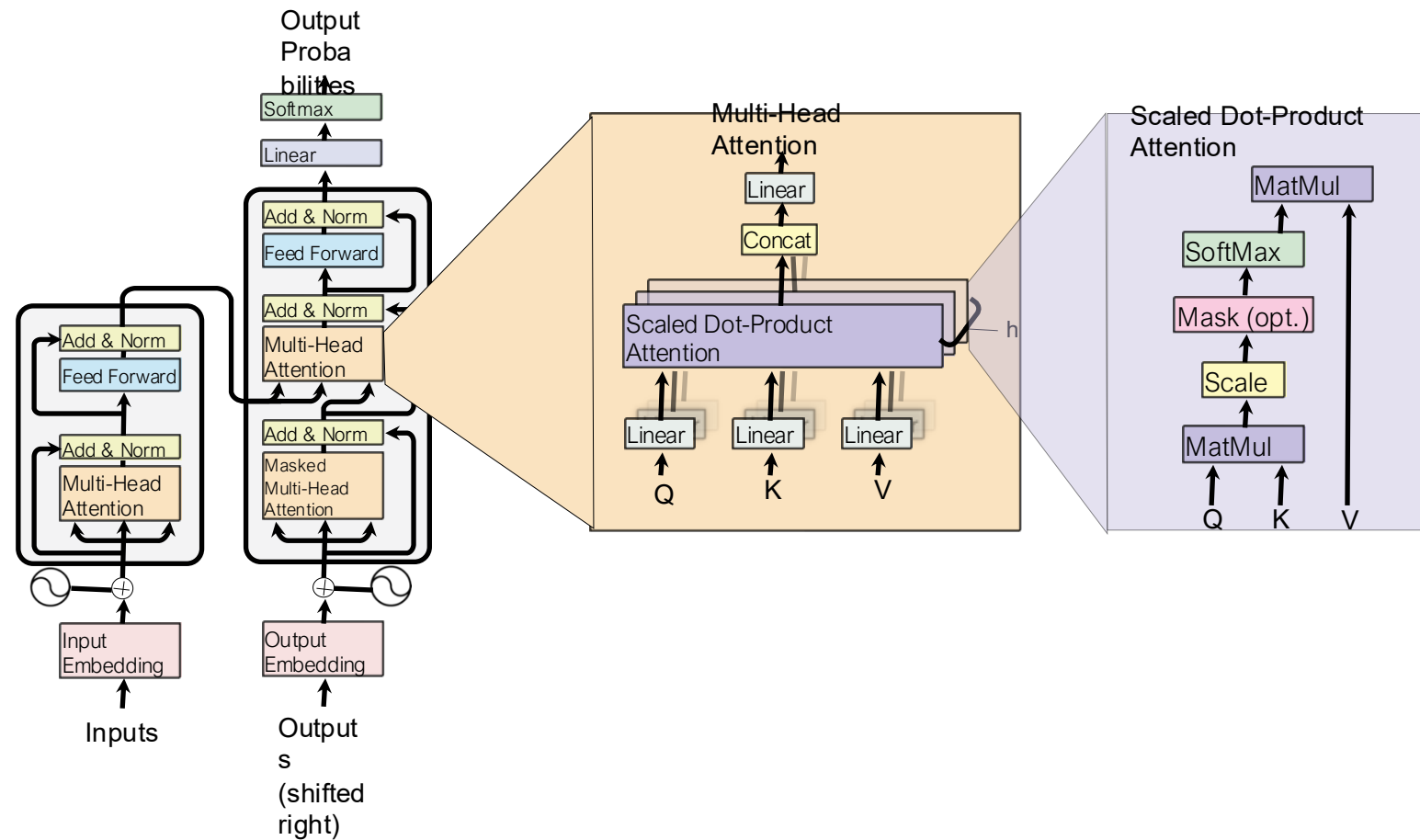
(c) Sparse Transformer (fixed)

Today's Topic

- Encoder-decoder Architecture
- Transformer Model
 - Embedding
 - Multihead Attention, Decoder Self-Attention
 - FFN
 - Layernorm
- ➡ • Training Techniques and Performance of Transformer
- Code walkthrough

Transformer in Original Paper

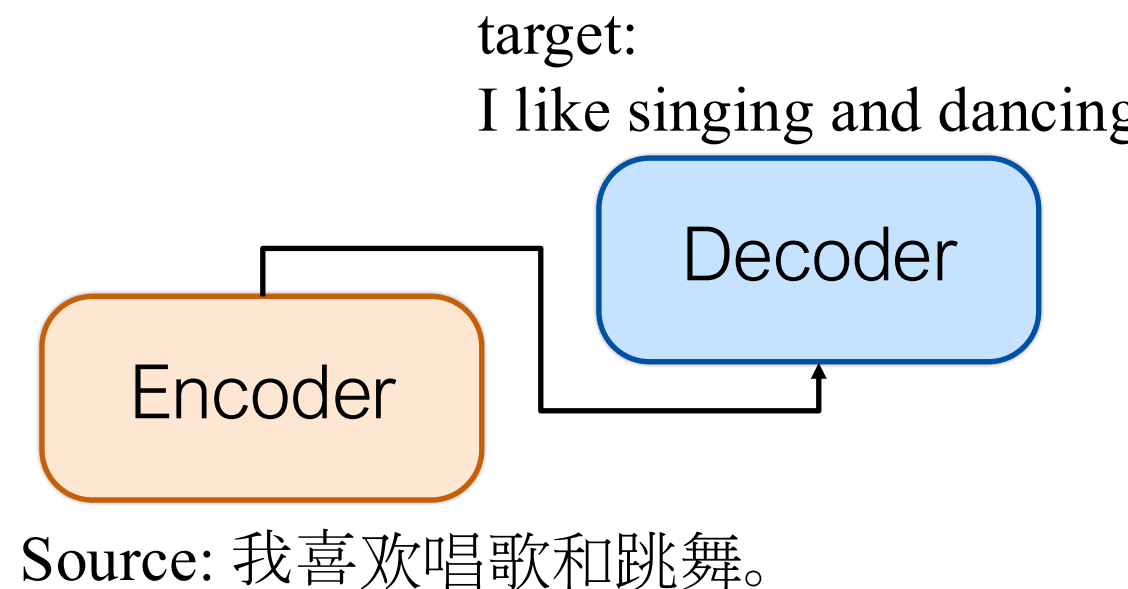
- C layers of encoder (=6)
- D layers of decoder (=6)
- Token Embedding: 512 (base), 1024 (large)
- FFN dim=2048



Training Transformer

$$P(Y|X) = \prod P(y_t | y_{<t}, x)$$

- Training loss: Cross-Entropy
$$l = -\sum_n \sum_t \log f_\theta(x_n, y_{n,1}, \dots, y_{n,t-1})$$
- Teacher-forcing during training.
- pretend to know groundtruth for prefix



Training Transformer for MT

- Dropout
 - Applied to before residual
 - and to embedding, pos emb.
 - $p=0.1 \sim 0.3$
- Label smoothing
 - 0.1 probability assigned to non-truth
- Vocabulary:
 - En-De: 37K using BPE
 - En-Fr: 32k word-piece (similar to BPE)

Label Smoothing

- Assume $y \in R^n$ is the one-hot encoding of label

$$y_i = \begin{cases} 1 & \text{if belongs to class } i \\ 0 & \text{otherwise} \end{cases}$$

- Approximating 0/1 values with softmax is hard

- The smoothed version

$$y_i = \begin{cases} 1 - \epsilon & \text{if belongs to class } i \\ \epsilon / (n - 1) & \text{otherwise} \end{cases}$$

- Commonly use $\epsilon = 0.1$

Training

- Batch
 - group by approximate sentence length
 - still need shufflingHardware
 - one machine with 8 GPUs (in 2017 paper)
 - base model: 100k steps (12 hours)
 - large model: 300k steps (3.5 days)
- Adam Optimizer
 - increase learning rate during warmup, then decrease
 - $\eta = \frac{1}{\sqrt{d}} \min(\frac{1}{\sqrt{t}}, \frac{t}{\sqrt{t_0^3}})$

ADAM

Gradient (via autograd/backprop)



$$m_{t+1} = \beta_1 m_t - (1 - \beta_1) \nabla \ell(x_t)$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) (\nabla \ell(x_t))^2$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}$$

momentum

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}}$$

variance

$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}} + \epsilon} \hat{m}_{t+1}$$

Parameter update

Other Popular Optimizers

- AdamW (used in LLaMA)
 - Similar to Adam but update with weight decay
- Muon (used in deepseek models)
 - Update 2D weight matrix using fast and orthogonal Newton-Schulz iterations.

Model Average

- A single model obtained by averaging the last 5 checkpoints, which were written at 10-minute interval (base)
- decoding length: within source length + 50
 - more on decoding in next lecture

Summary

- Key components in Transformer and LLM Architecture
 - Positional Embedding (to distinguish tokens at different pos)
 - Multihead attention
 - Residual connection
 - layer norm
 - FFN and SwishGLU
 - RoPE

Summary

- Training Technique:
 - Dropout
 - Label smoothing
 - optimization alg
- Pretraining: Mask-labeled recovery of random spans + next token prediction
- Multitask supervised fine-tuning with instruction templates
 - T5, InsturctGPT

Code Go-through

<https://nlp.seas.harvard.edu/annotated-transformer/>