Machine Learning in Particle Theory - MITP Summer School 2023

Eilam Gross

Particle Flow with Deep Learning

✅ Lecture 1: GNN+Attention

✓ Lecture 2: Transformers + Set Generation
(with the help of **N. Kakati** and **N. Soybelman**)

- Lecture 3: Hyper Graphs + TSPN Particle Flow
(with the help of **N. Kakati**)

Syllabus

- ✅ Graph Neural Nets
  - ✅Set to Graph
- ✓ Attention is all you need
  - ✓ Transformers, ✅ Slot Attention (SA)
  - ✓ Set Prediction Networks with a Transformer and SA (TSPN-SA)
- ✓ Constrained Variational Auto Encoder (cVAE)
- Particle Flow
  (Reconstructing Particles in Jets using TSPN-SA,
  Hyper-Graph PFlow [HGPflow])
- ✓ Simulation of PF Objects (Using TSPN-SA, cVAE)

Attention Is All You Need

https://arxiv.org/abs/1706.03762

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

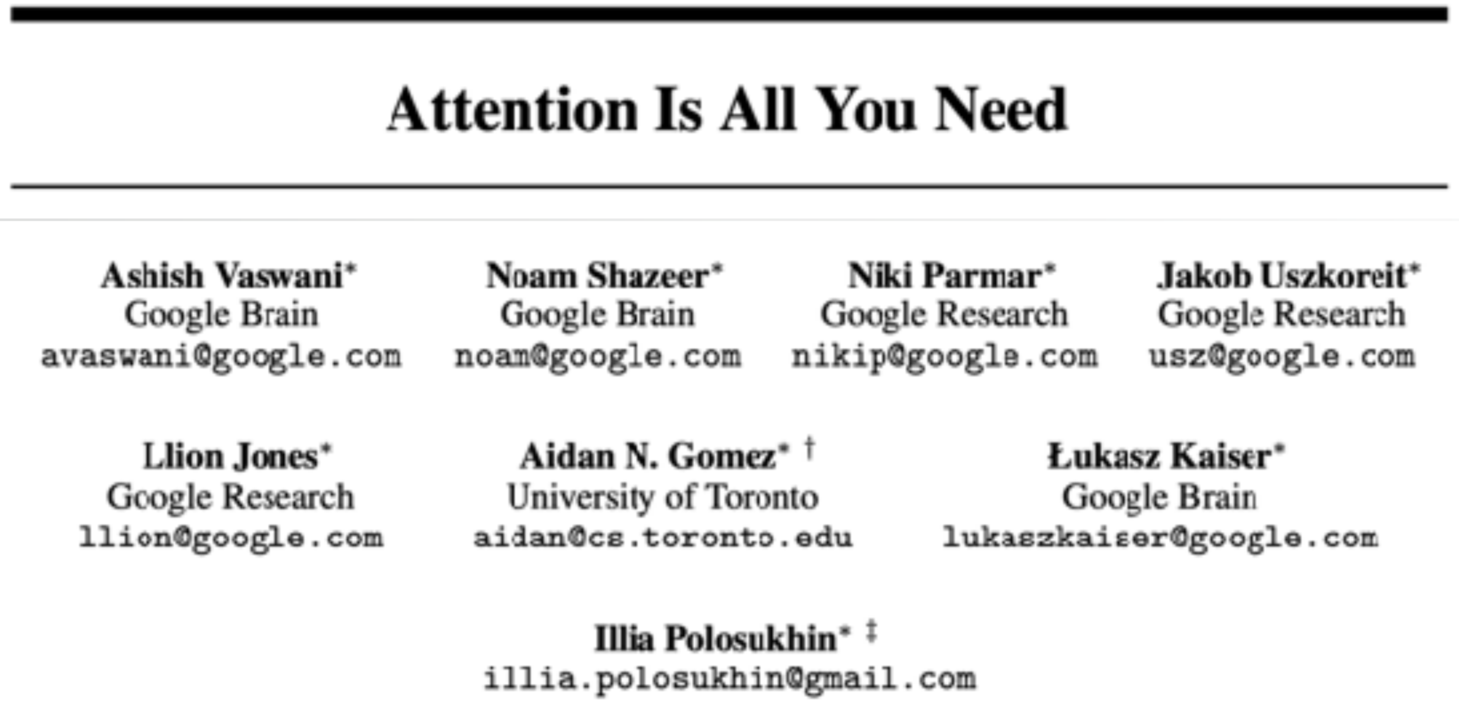
Attention is All You Need

Level 1 N. Kakati

Once upon a time, people in AI were working in peace..

The field was making good progress

One bright summer morning, a paper showed up



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

AND THEN EVERYTHING CHANGED!

Where is Transformer?

It's everywhere…

DALL·E 2

Text to Image

Google MUSE
Text to Image

Midjourney

Text to Image

Image Classification

Computer Vision (CV)

ChatGPT

Generative Pre-trained Transformers

Google BERT

Bidirectional Encoder Representations from Transformers

Sentiment Analysis

Google Translate

Q & A

grammarly

Natural language processing (NLP)

Jet tagging

Fast Simulation

Particle Reconstruction

Super Resolution

What is a Transformer?

- Looks very complicated
- It'll make sense once we understand the components

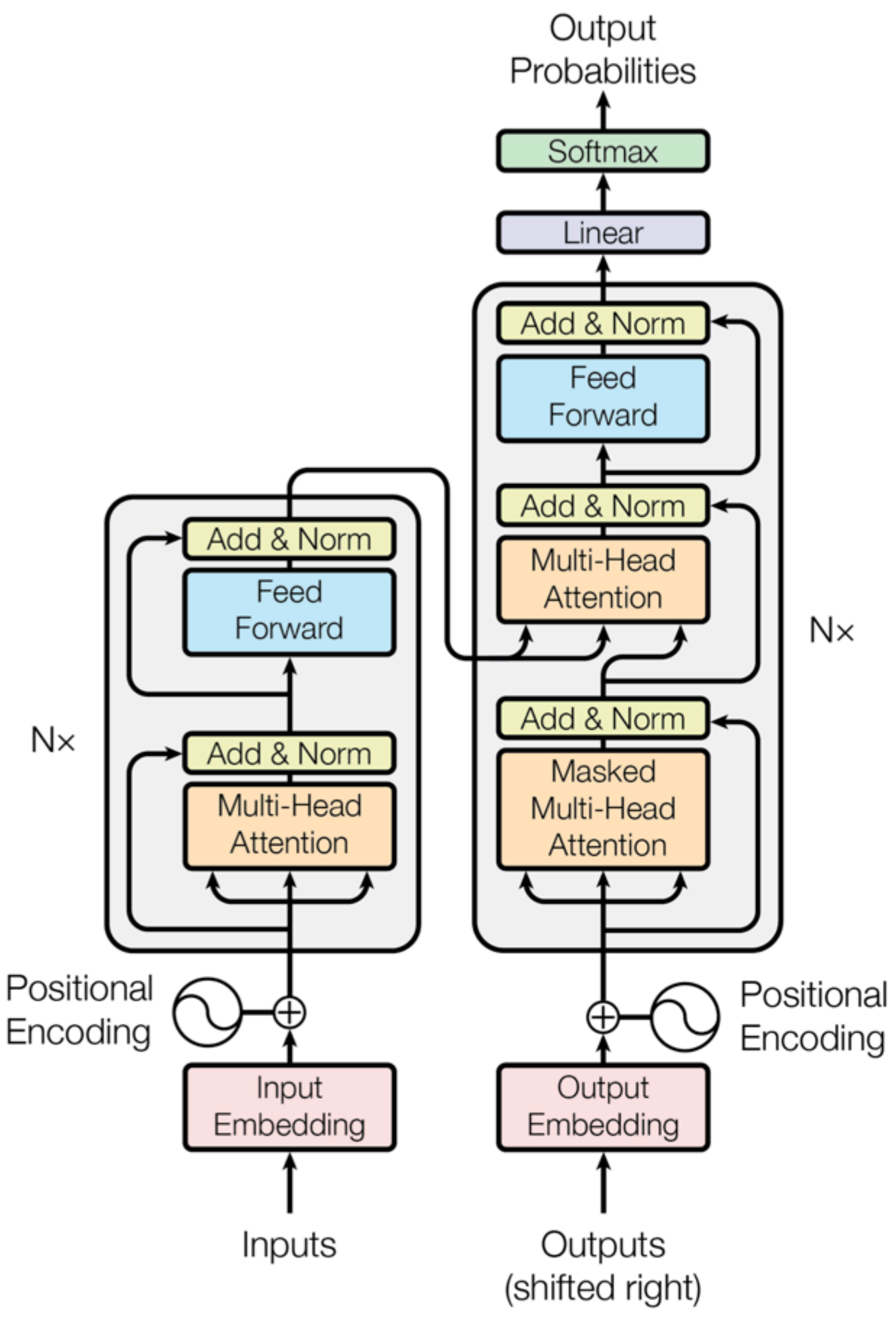
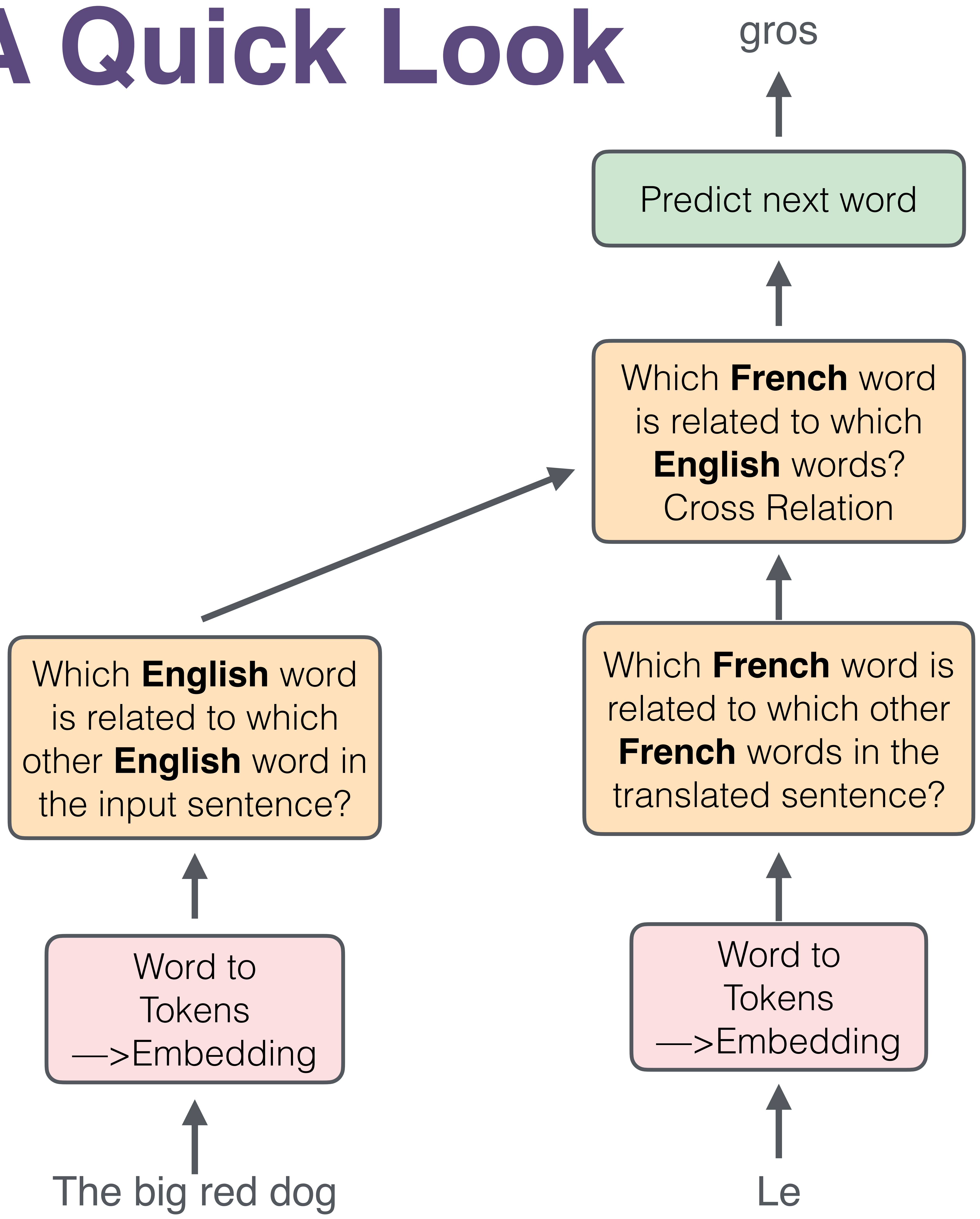


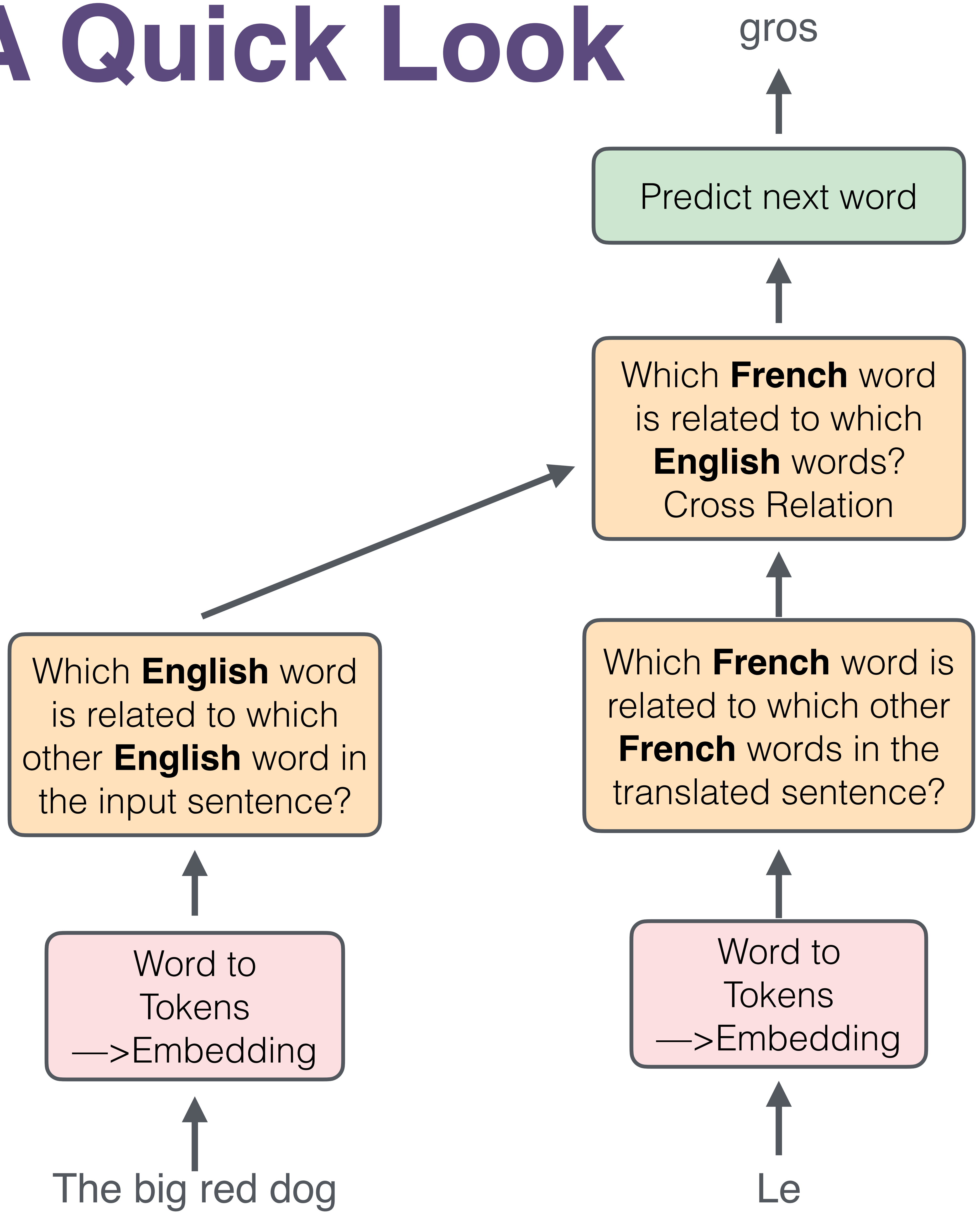
Figure 1: The Transformer - model architecture.

A Quick Look



- We are doing translation
  - English to French
- English sentence
  - The big red dog
- Someone told you that the first word in French is "Le"
  - You need to predict the next words one by one and complete the sentence

A Quick Look



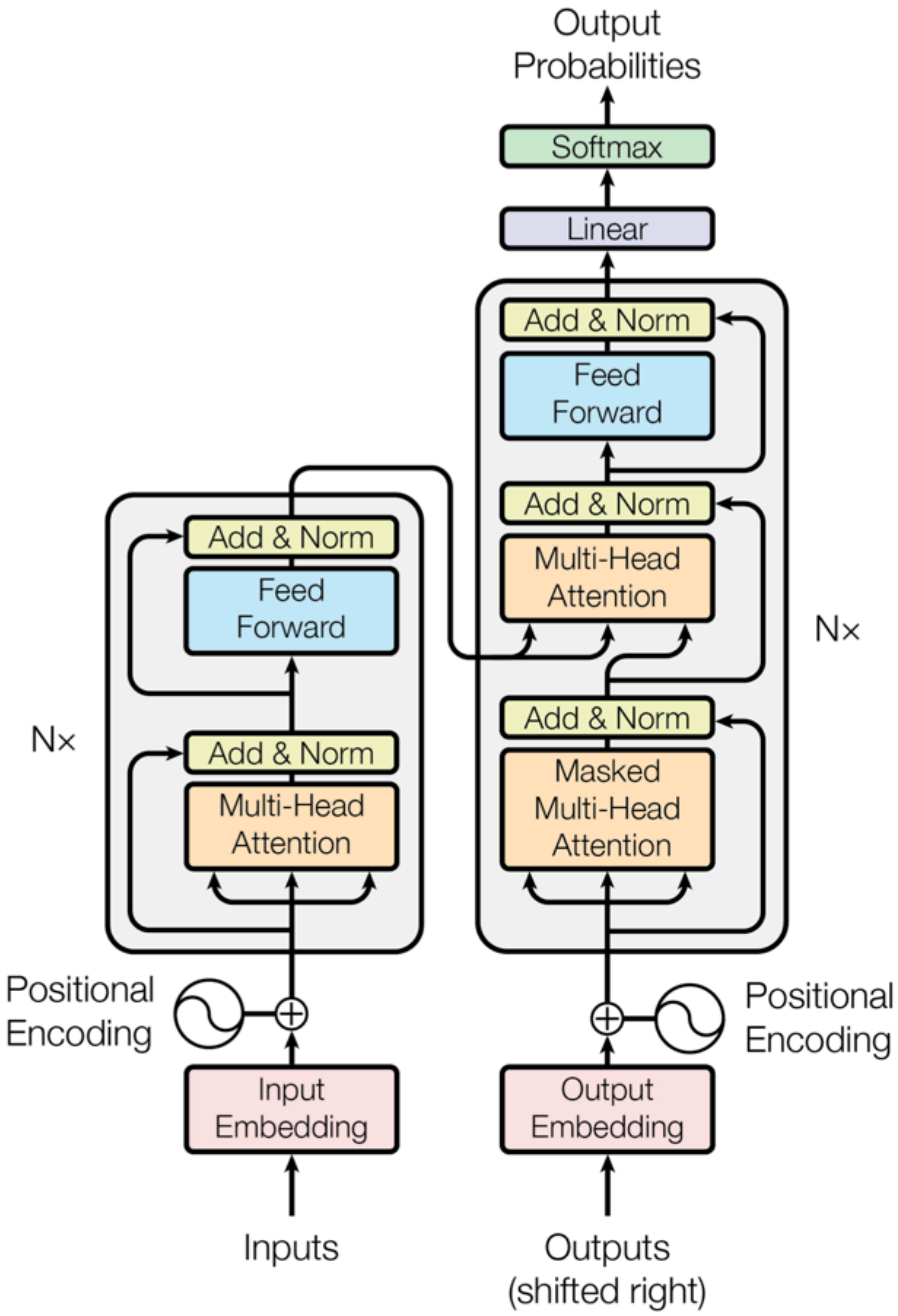


Figure 1: The Transformer - model architecture.

Attention is All You Need

Level 2 N. Kakati

- Pretty good explanation here
  - https://www.youtube.com/watch?v=TQQlZhbC5ps
  - YouTube channel link
- This part of the lecture is mainly based on that video

Embedding

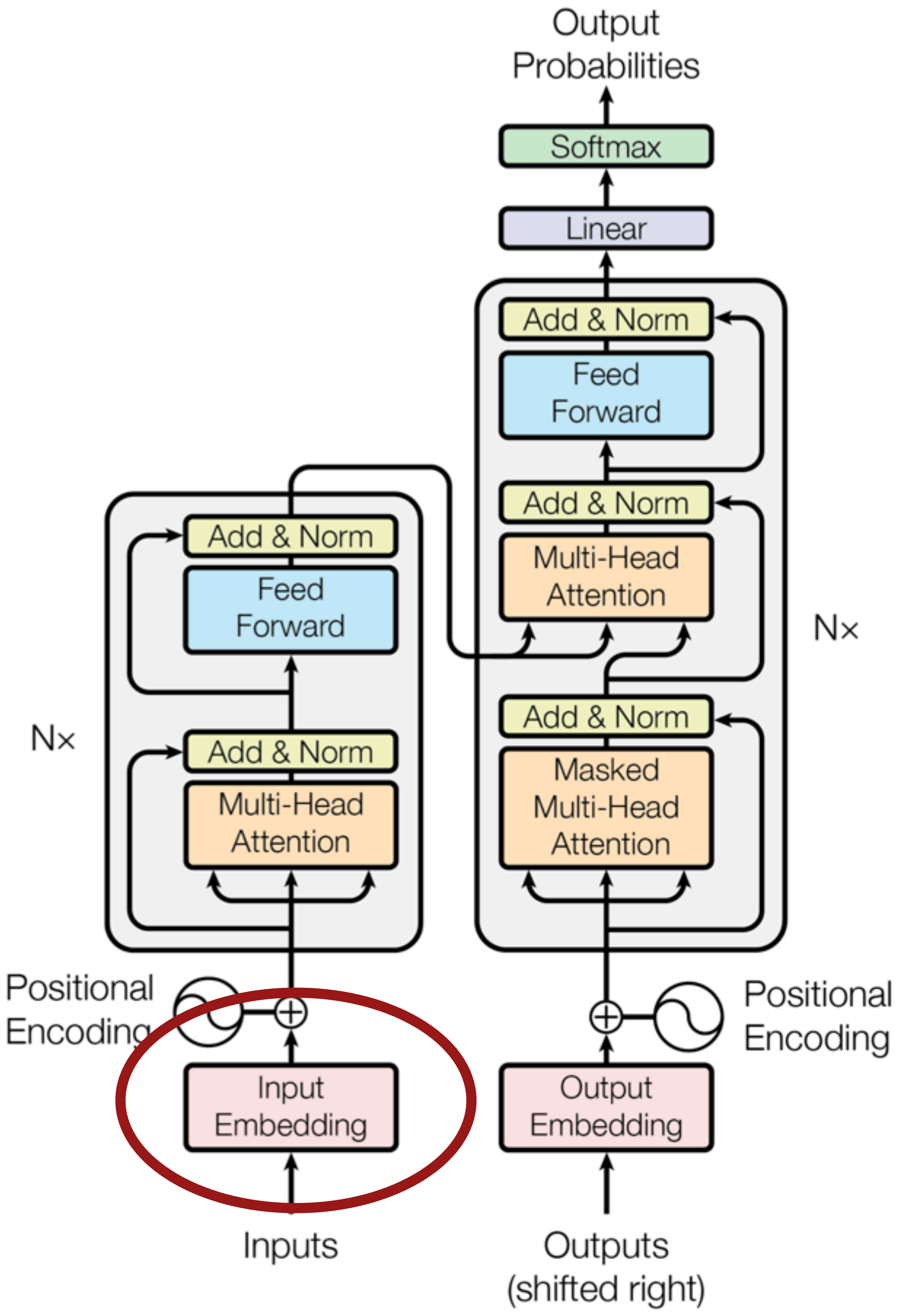
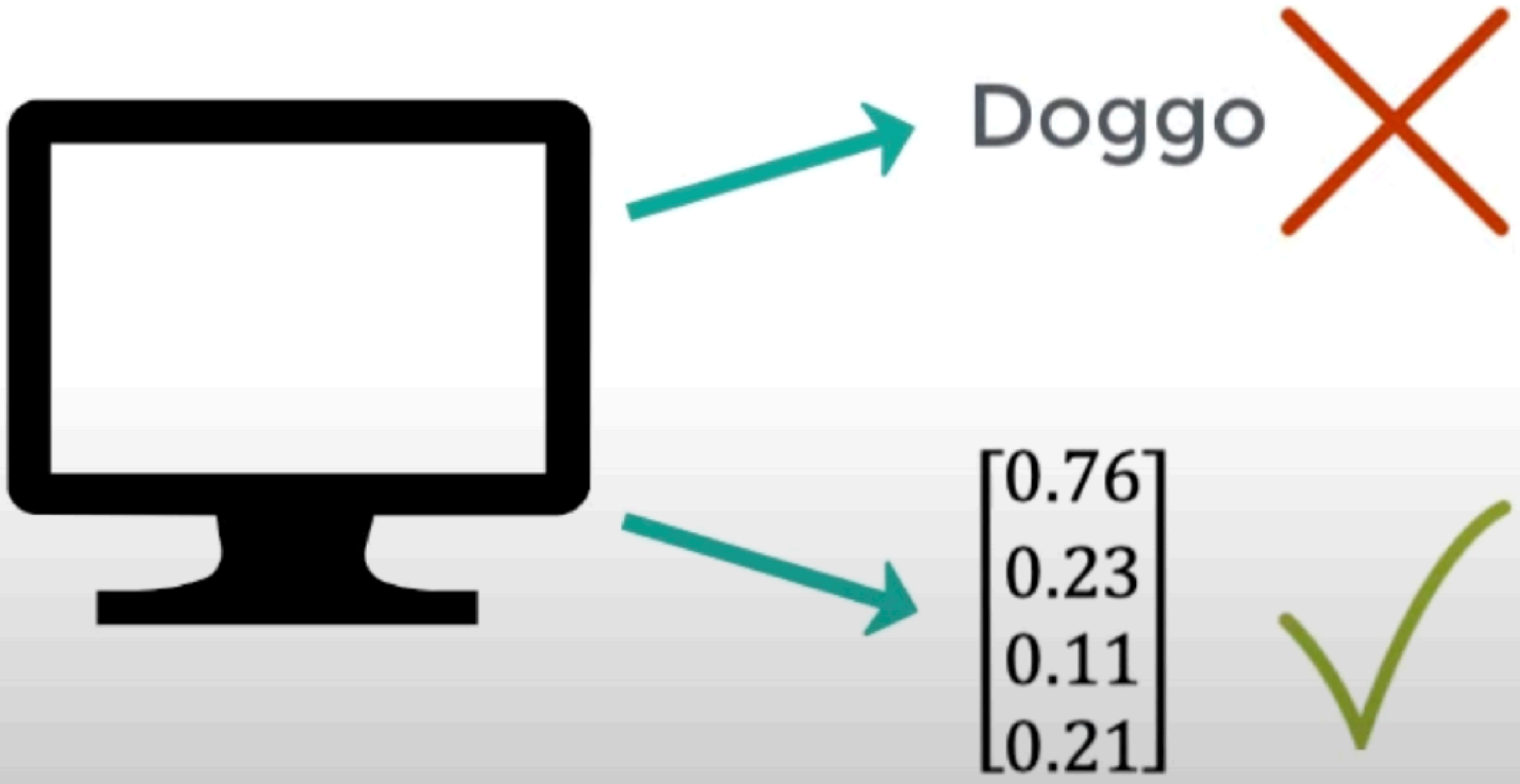
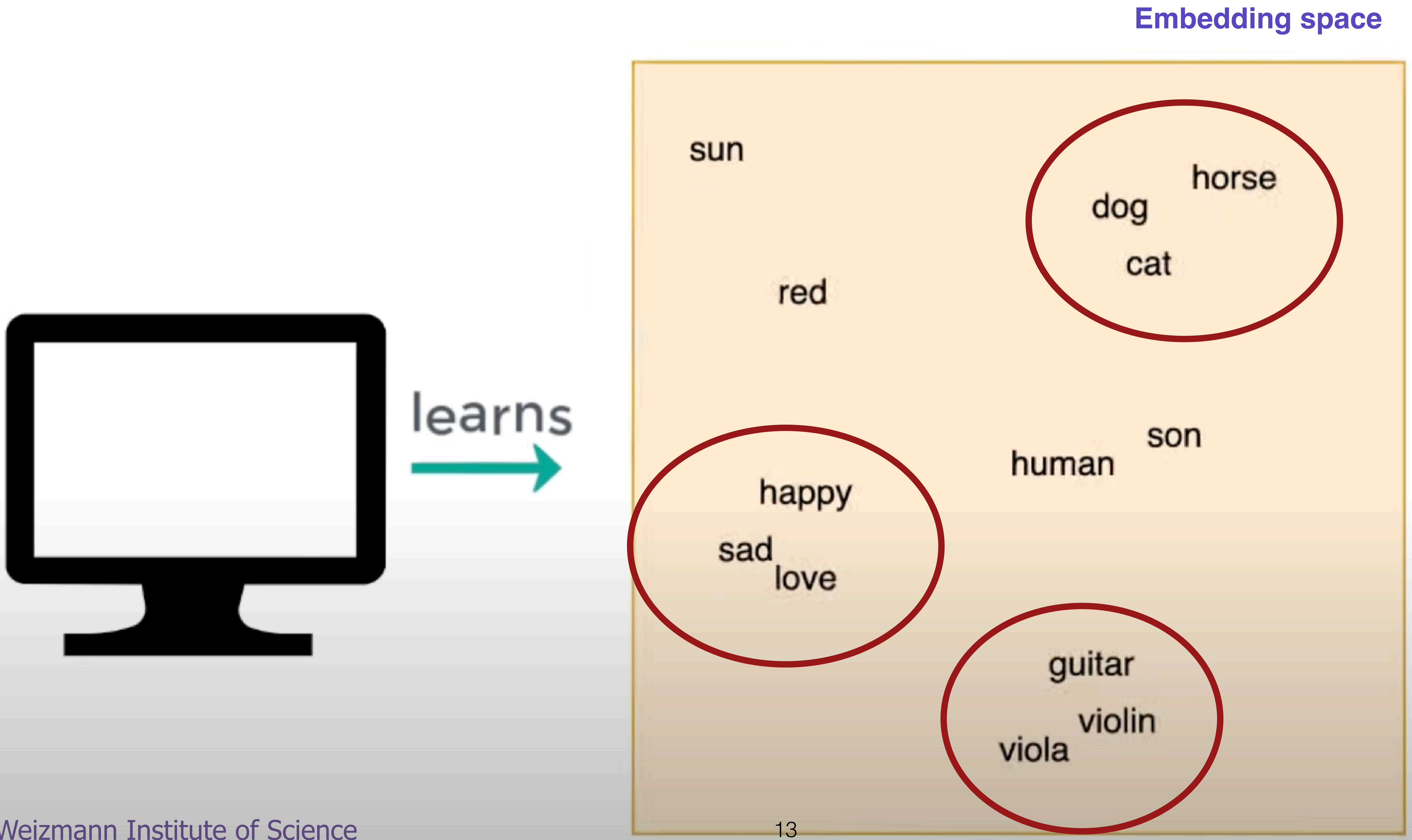


Figure 1: The Transformer - model architecture.

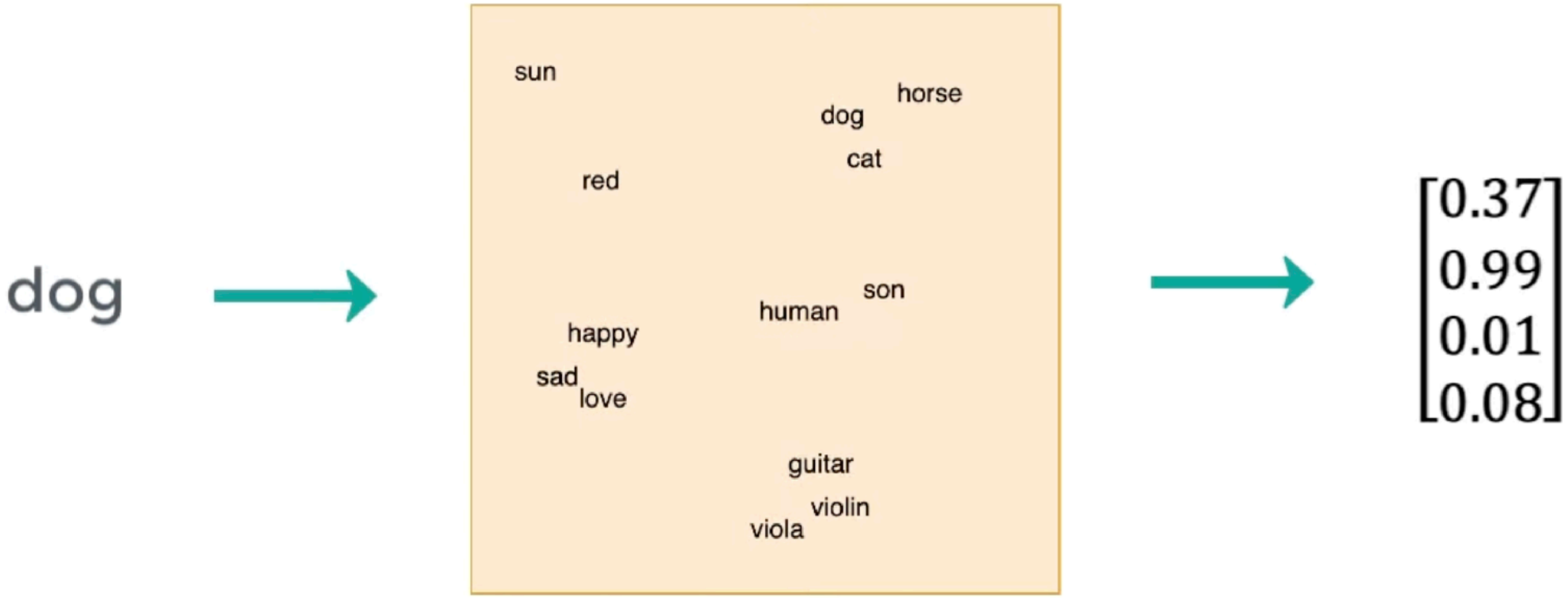
Embedding



Embedding


learns
Embedding space
sun
horse
dog
cat
red
son
human
happy
sad
love
guitar
violin
viola

Embedding


dog
sun
horse
dog
cat
red
son
human
happy
sad
love
guitar
violin
viola
0.37
0.99
0.01
0.08

Embedding

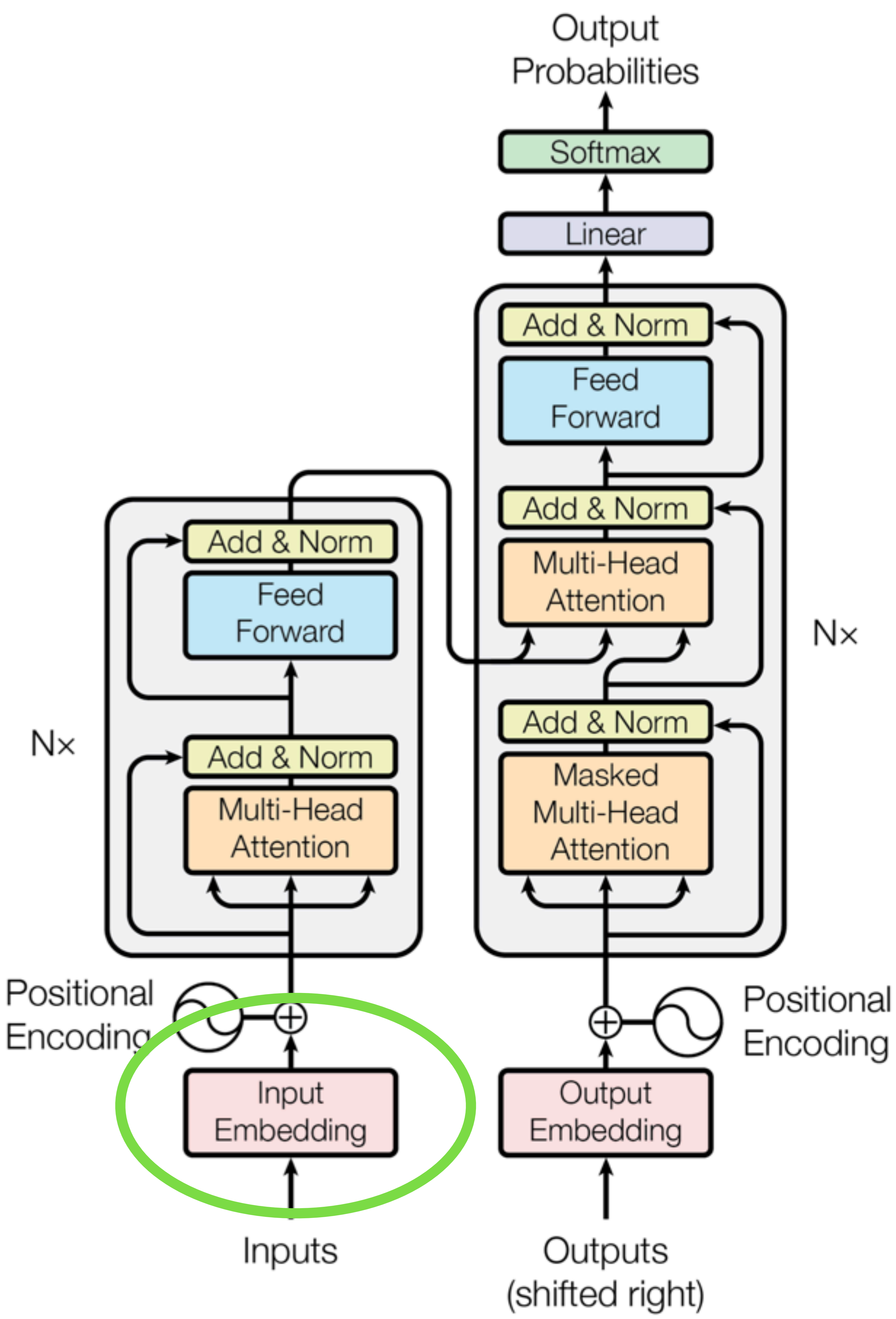


Figure 1: The Transformer - model architecture.

Positional encoding

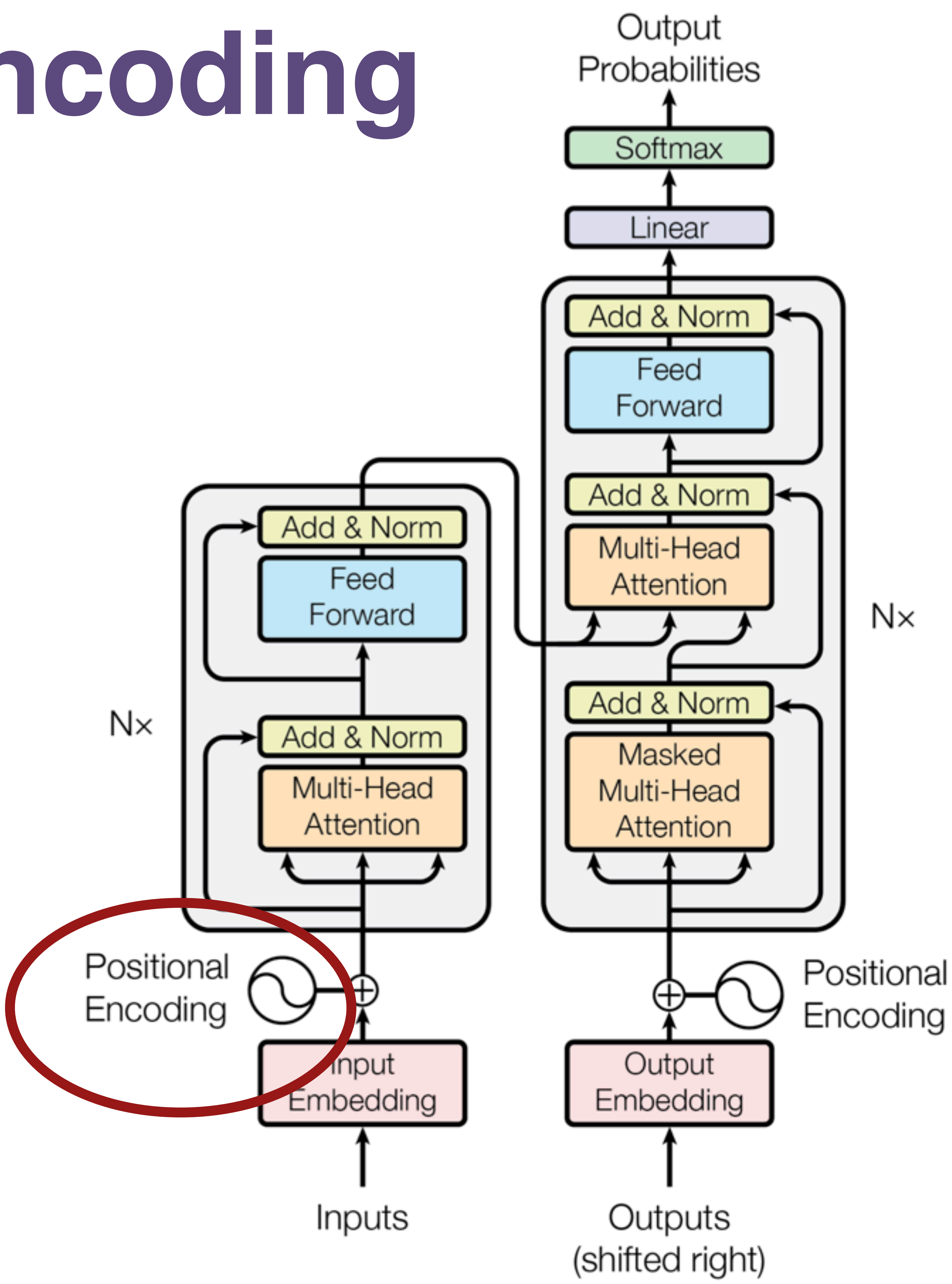
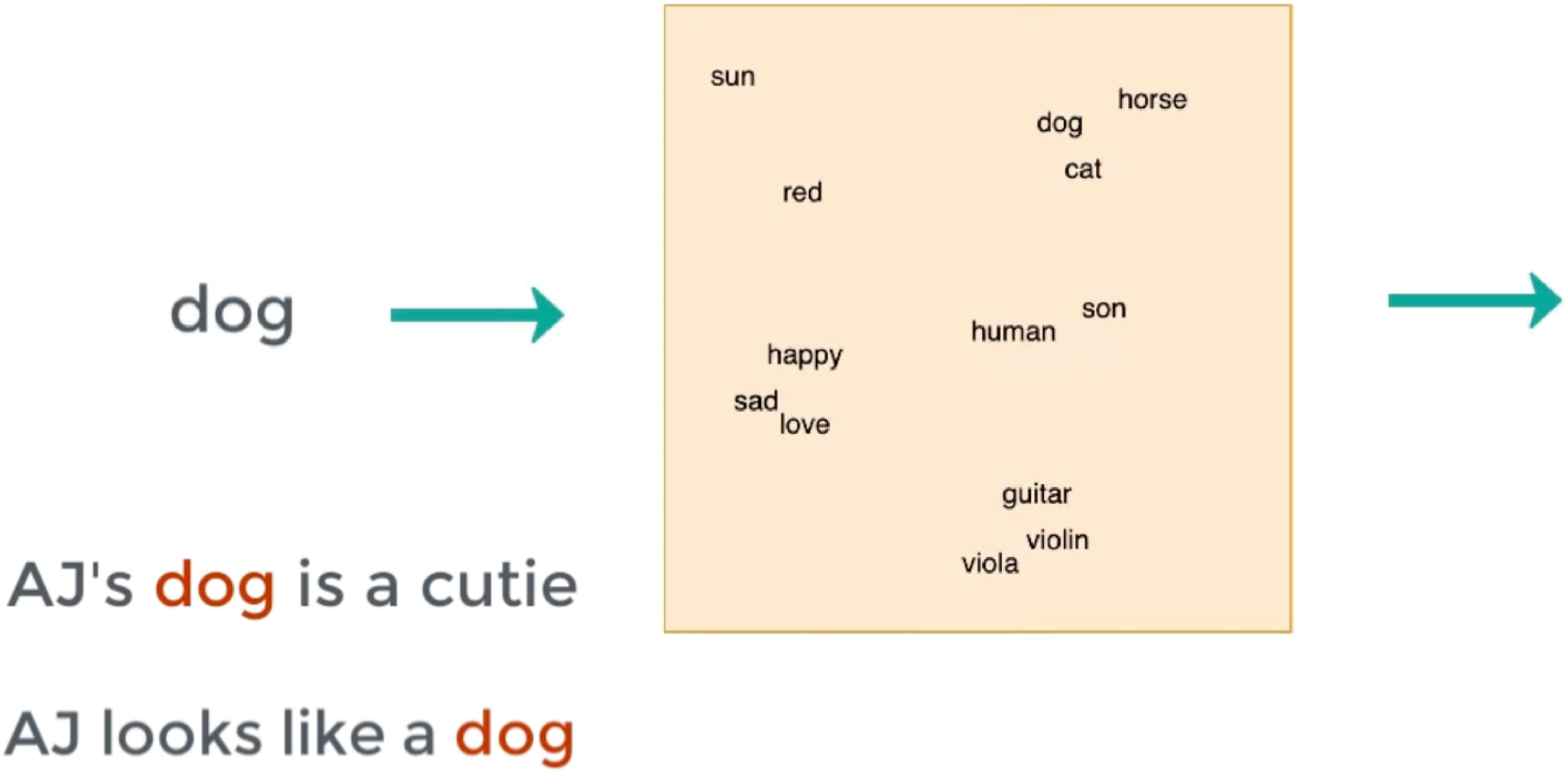


Figure 1: The Transformer - model architecture.

Positional encoding

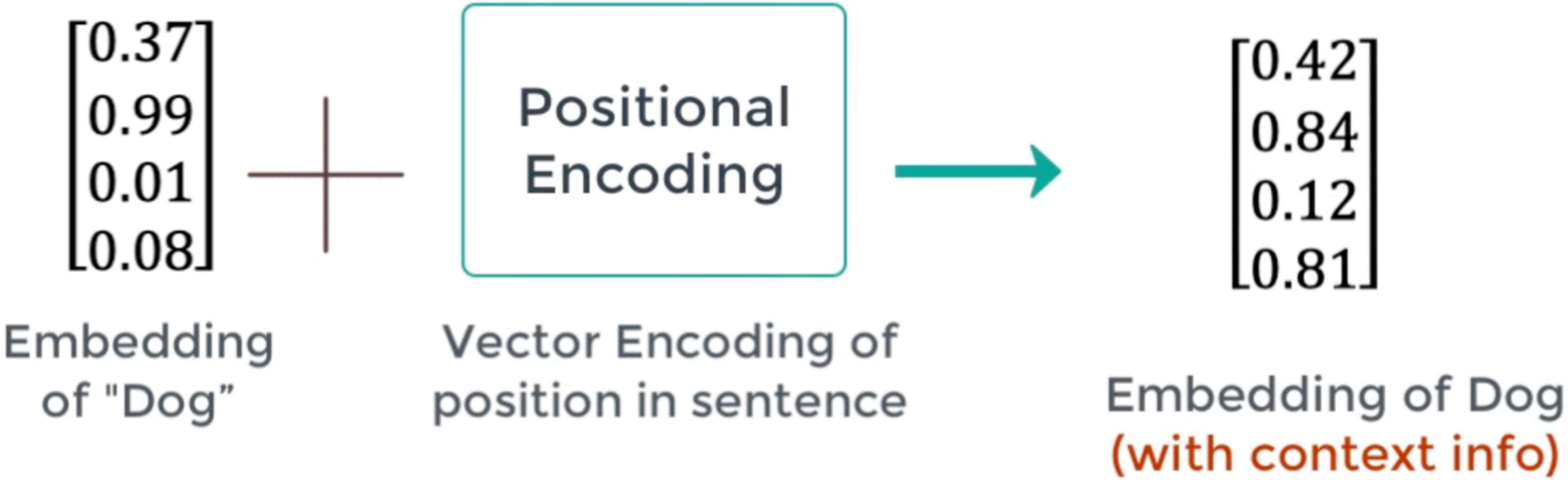


$$\begin{bmatrix} 0.37 \\ 0.99 \\ 0.01 \\ 0.08 \end{bmatrix}$$

Positional Encoding

Positional encoder: Vector that gives context based on position of word in sentence

Positional Encoding



Positional encoding

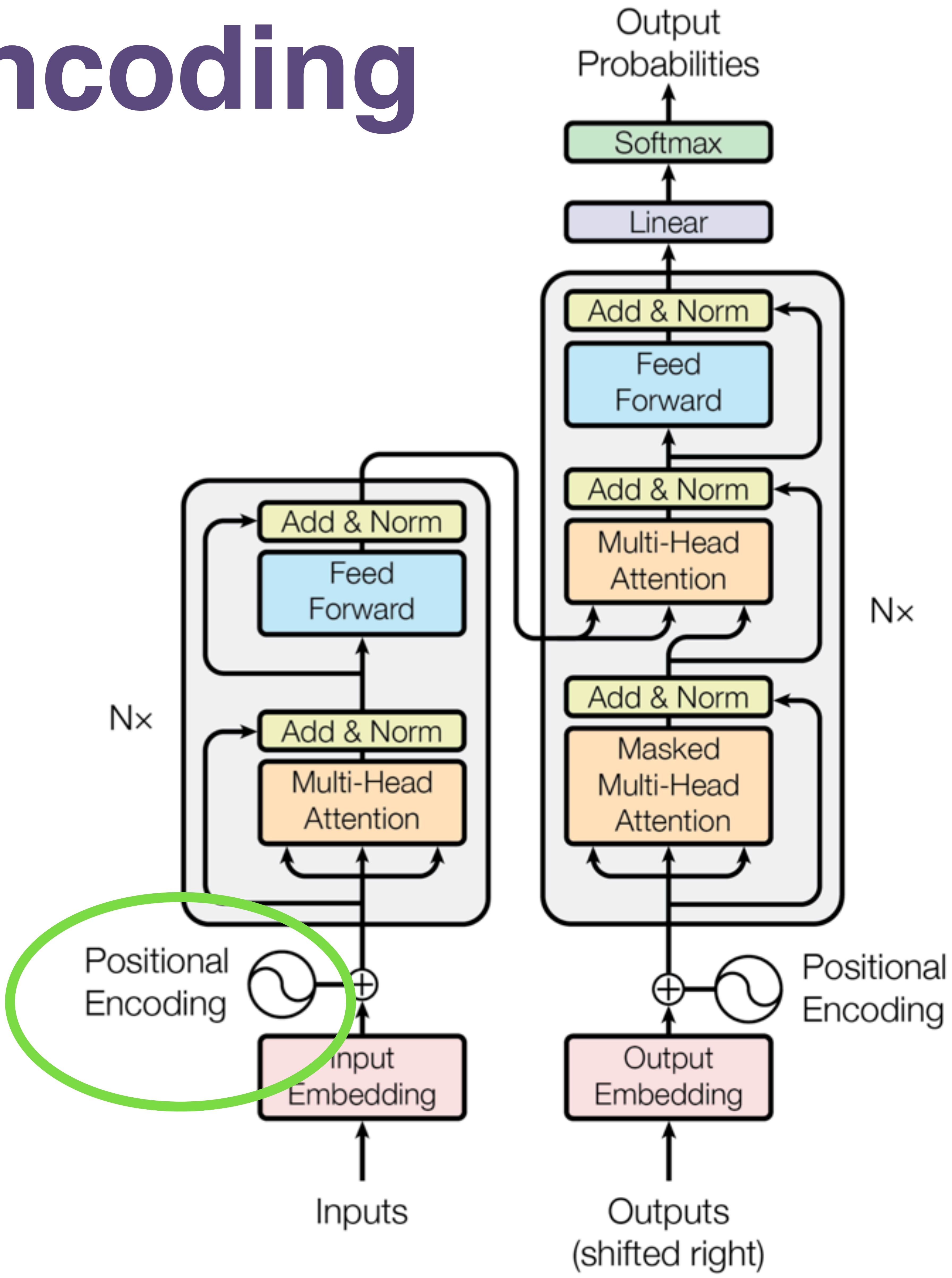


Figure 1: The Transformer - model architecture.

Attention

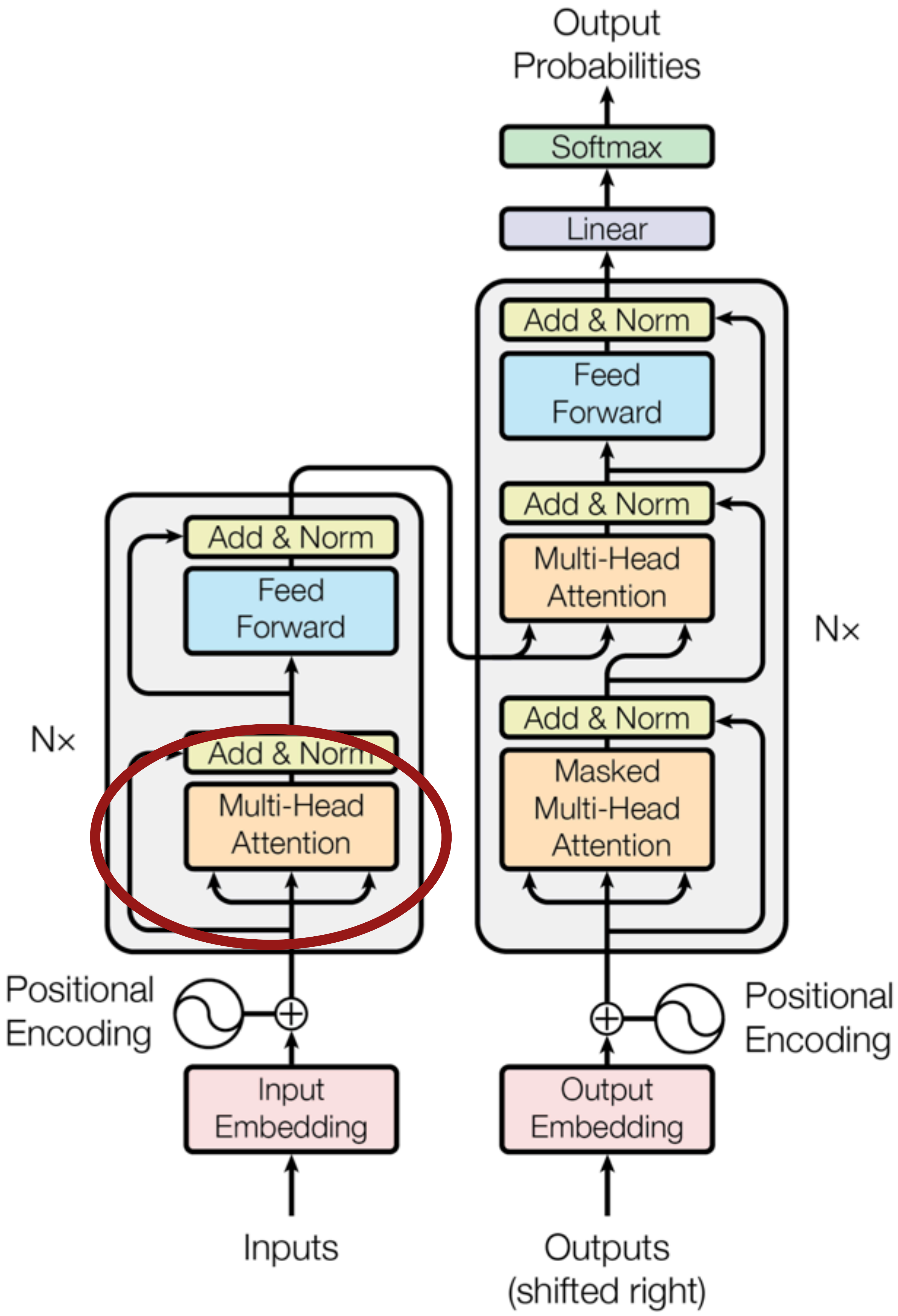
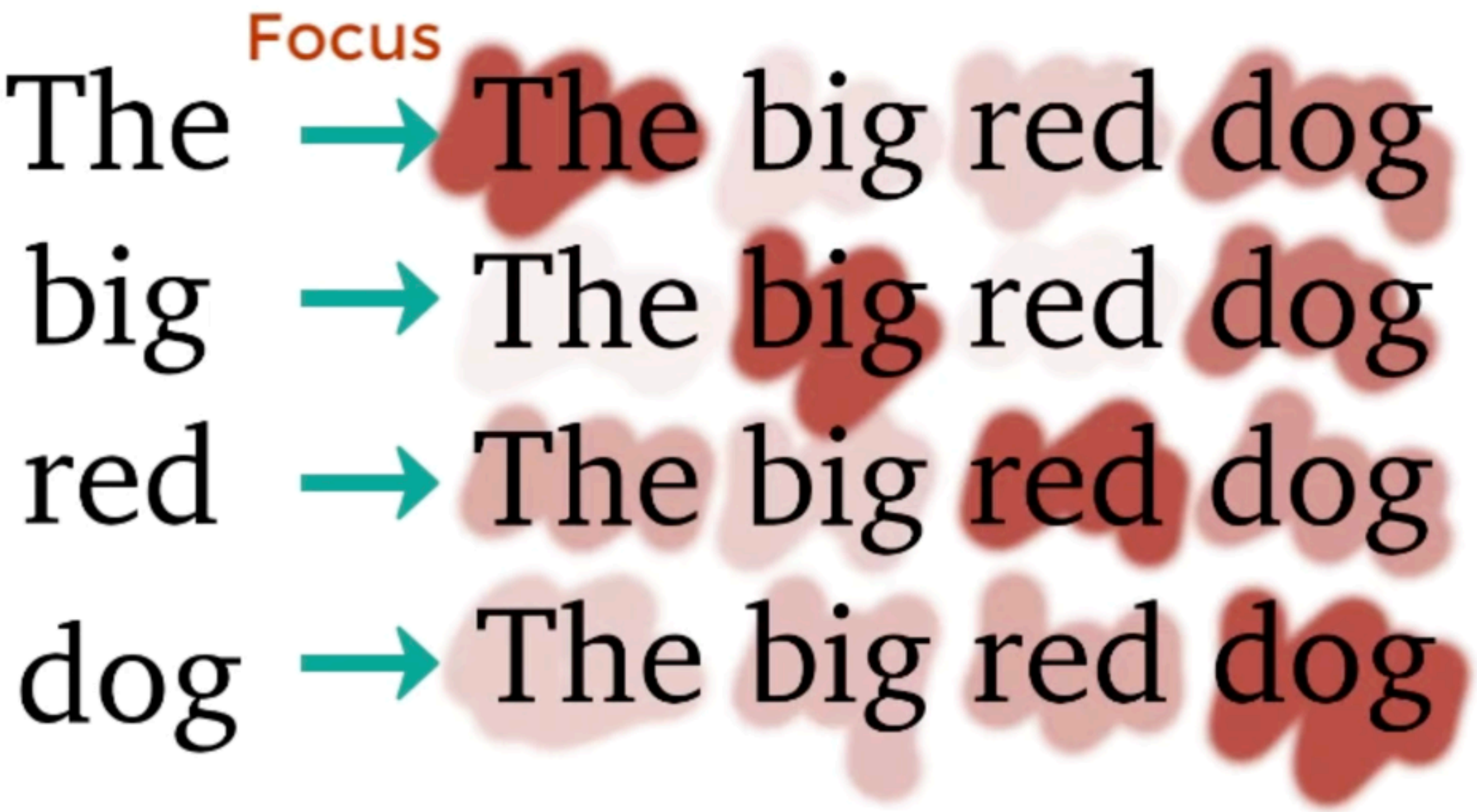


Figure 1: The Transformer - model architecture.

Attention

How relevant one word is to the others?



Attention matrix

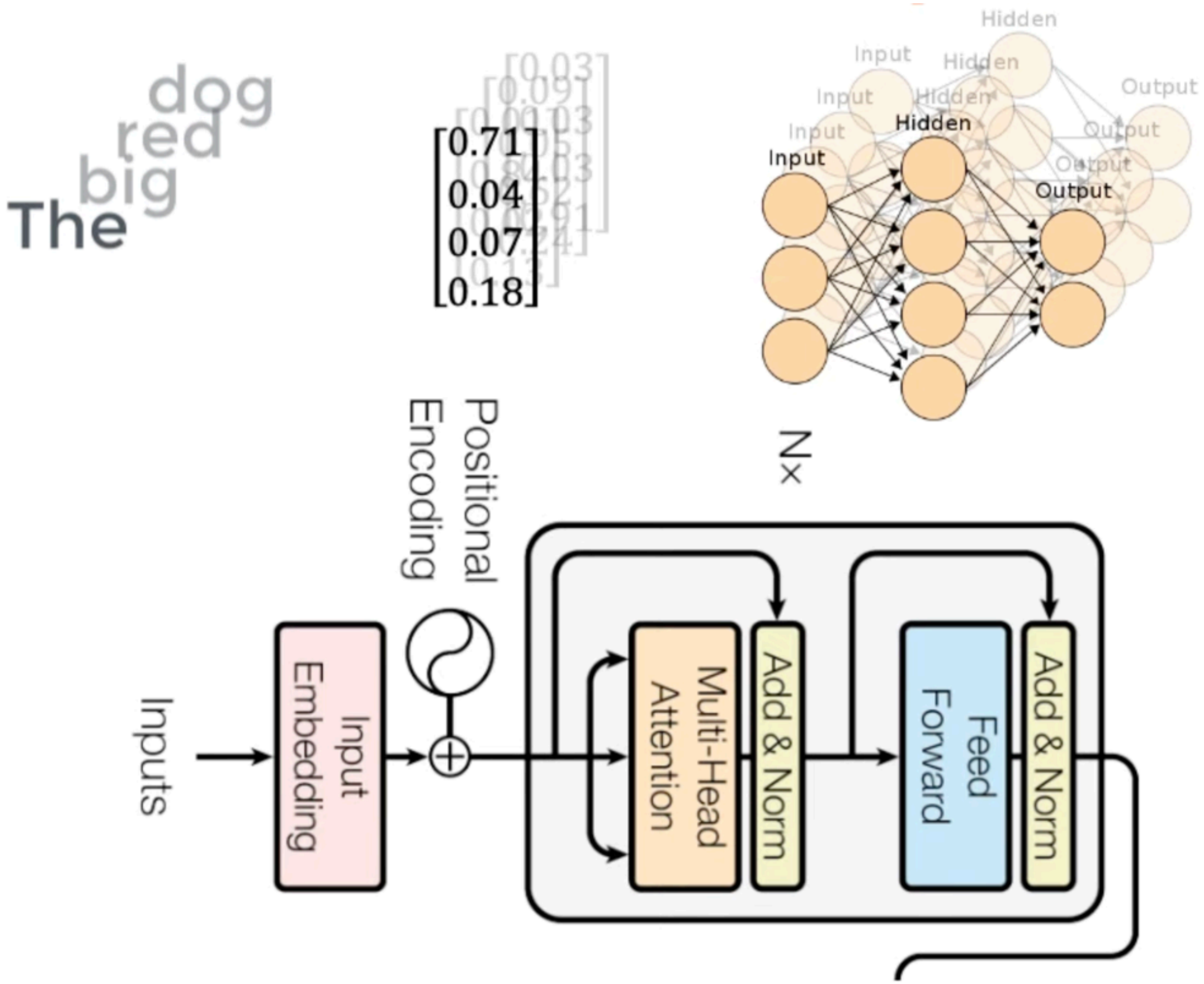
$$[0.71 \quad 0.04 \quad 0.07 \quad 0.18]^T$$

$$[0.01 \quad 0.84 \quad 0.02 \quad 0.13]^T$$

$$[0.09 \quad 0.05 \quad 0.62 \quad 0.24]^T$$

$$[0.03 \quad 0.03 \quad 0.03 \quad 0.91]^T$$

FeedForward



Attention + FeedForward

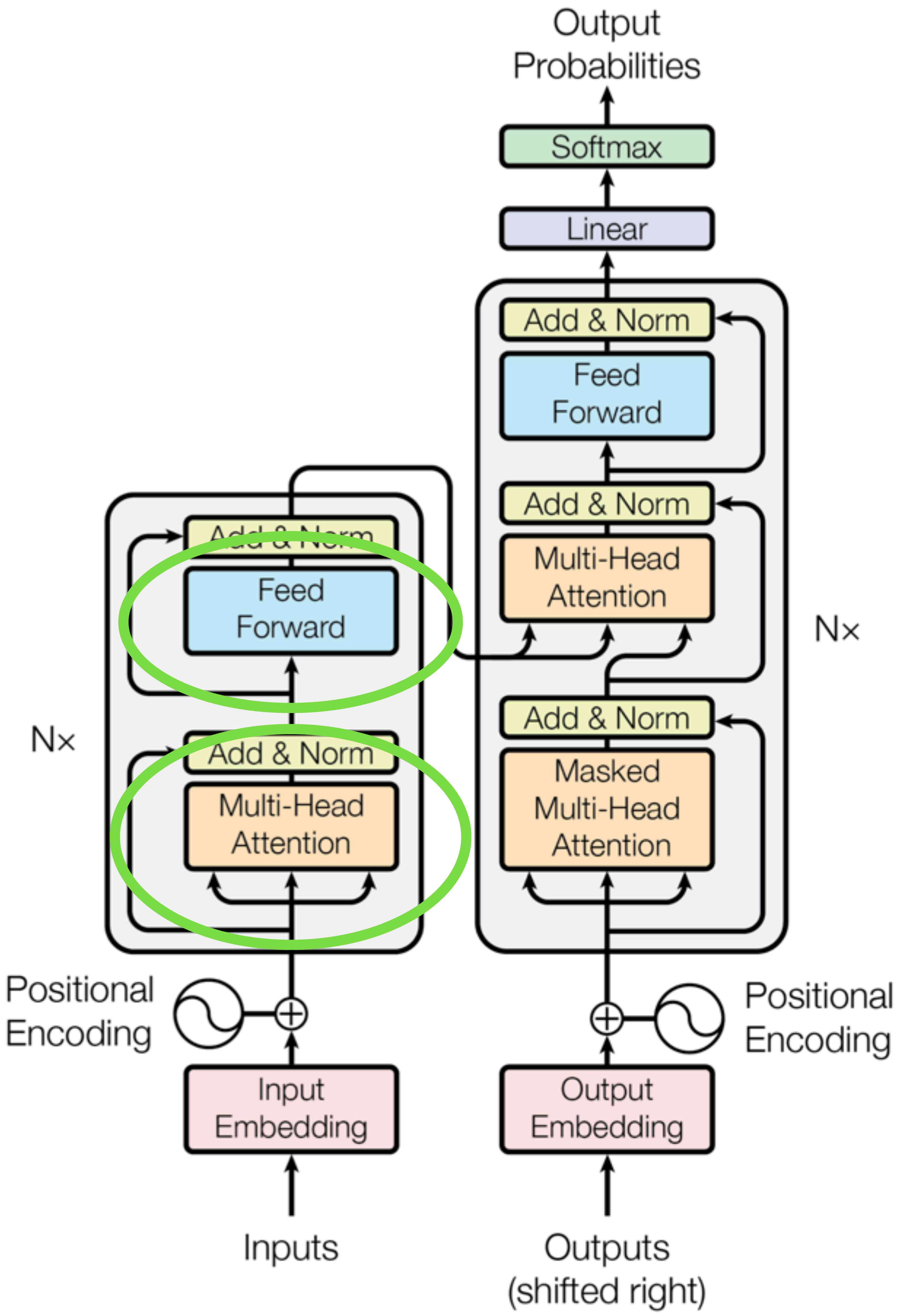


Figure 1: The Transformer - model architecture.

Decoder

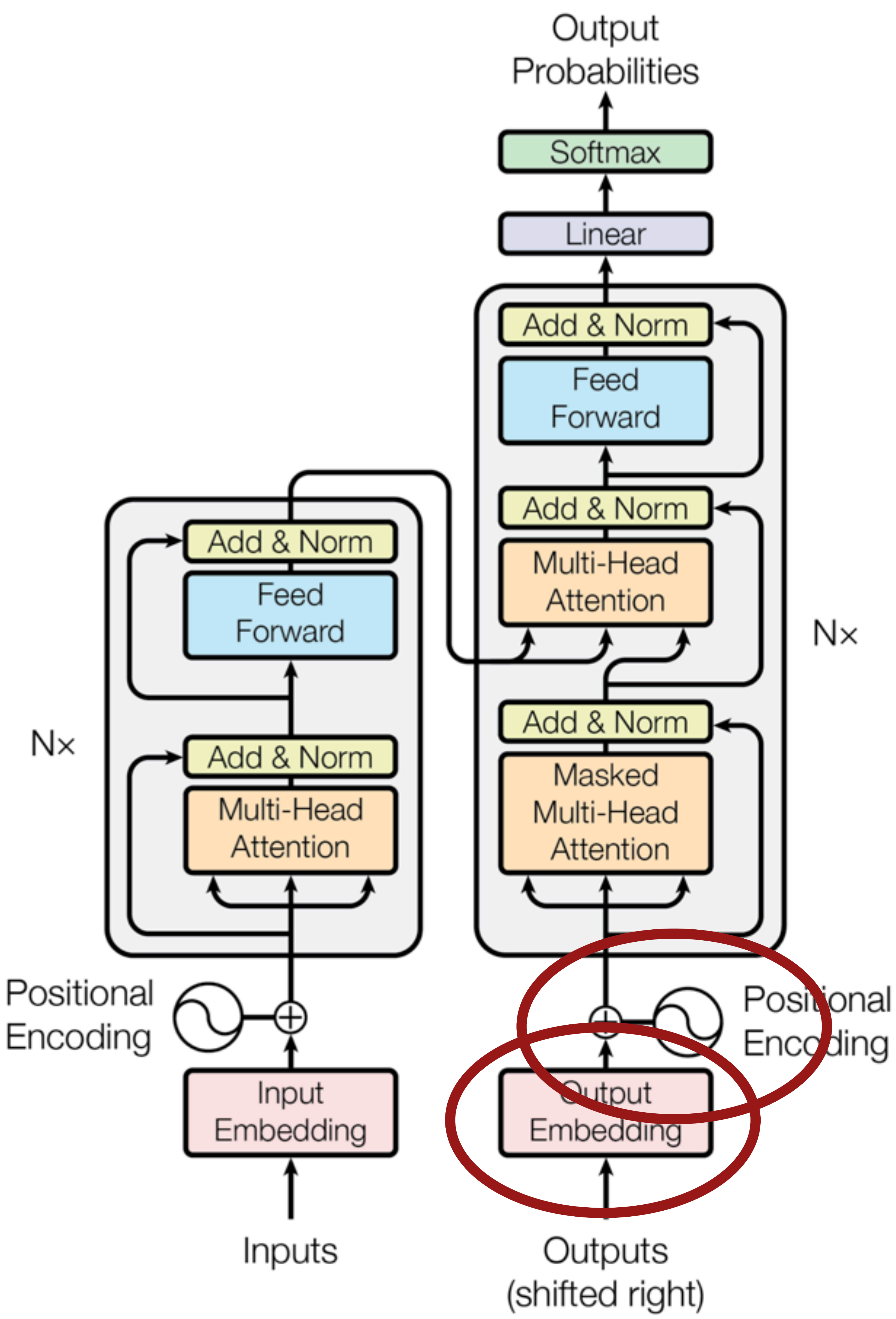


Figure 1: The Transformer - model architecture.

Decoder

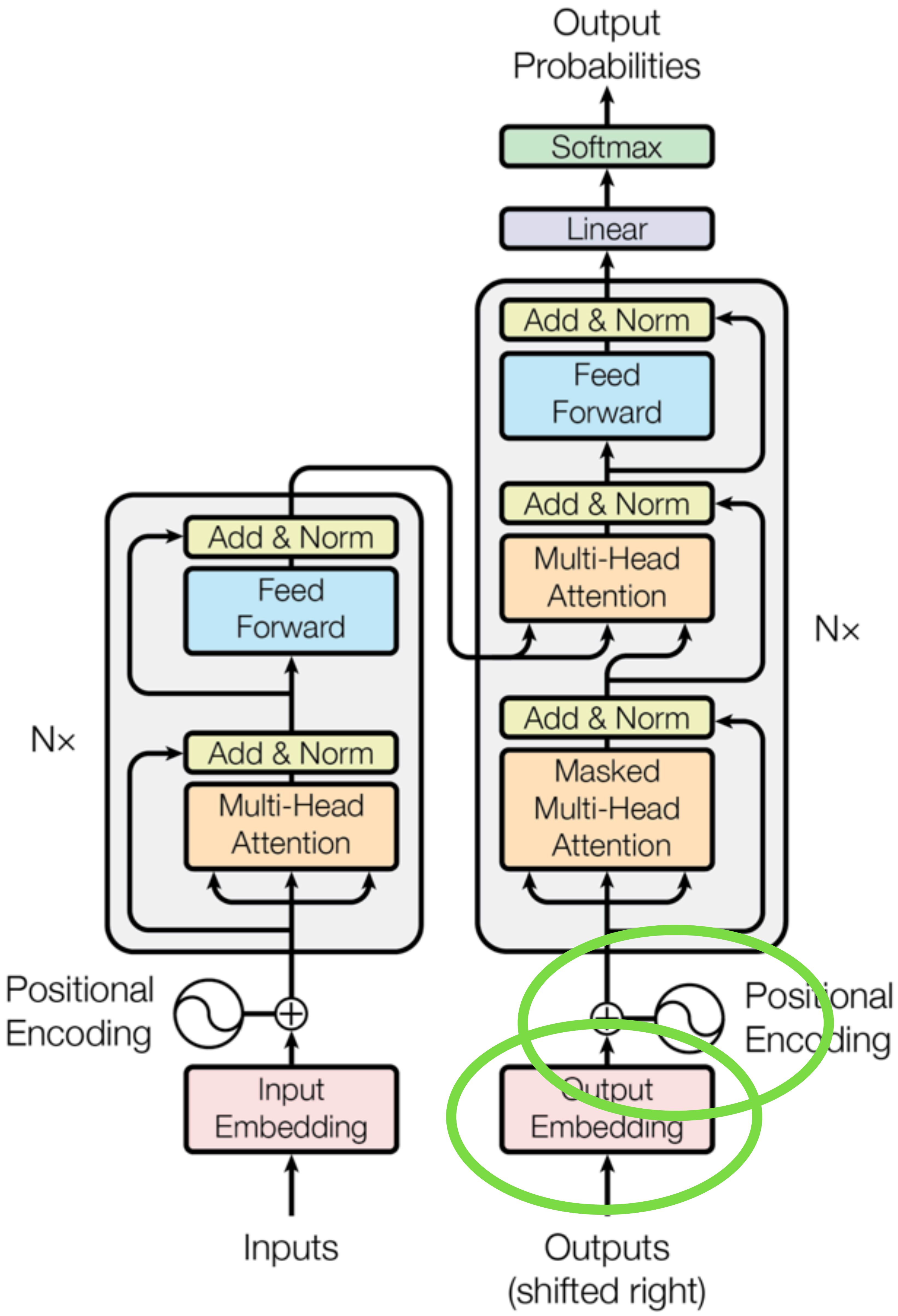


Figure 1: The Transformer - model architecture.

Masked attention

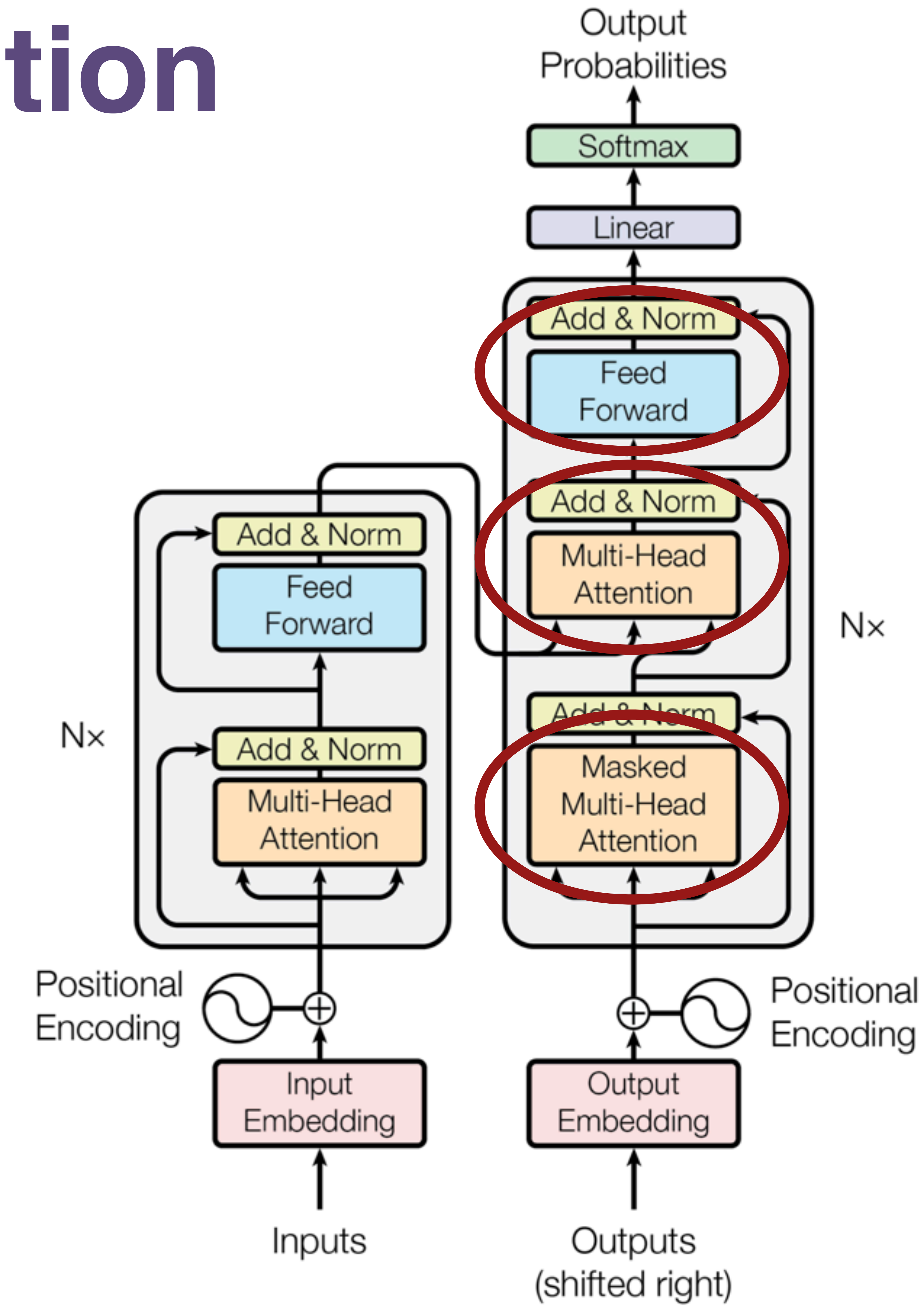
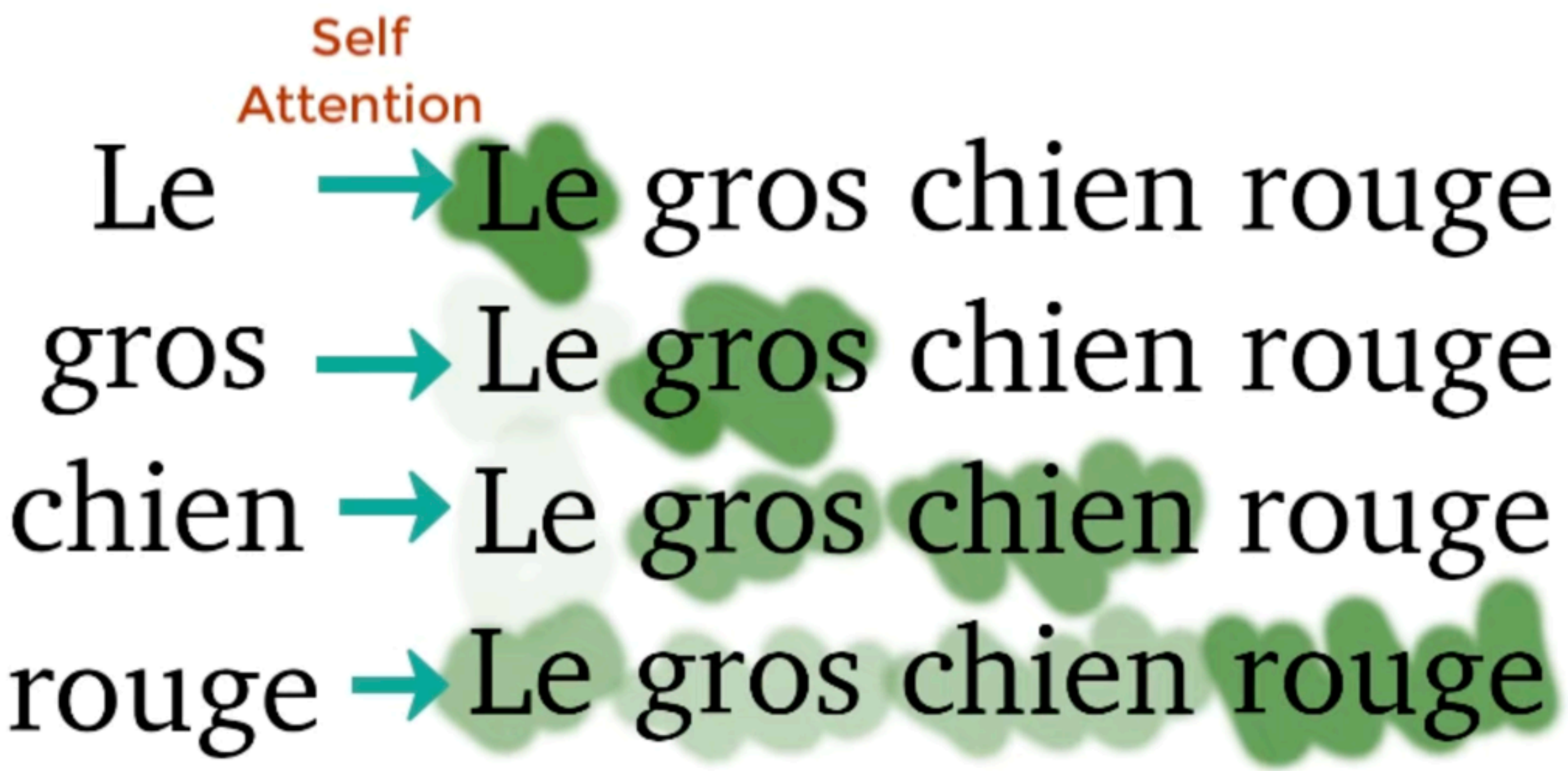
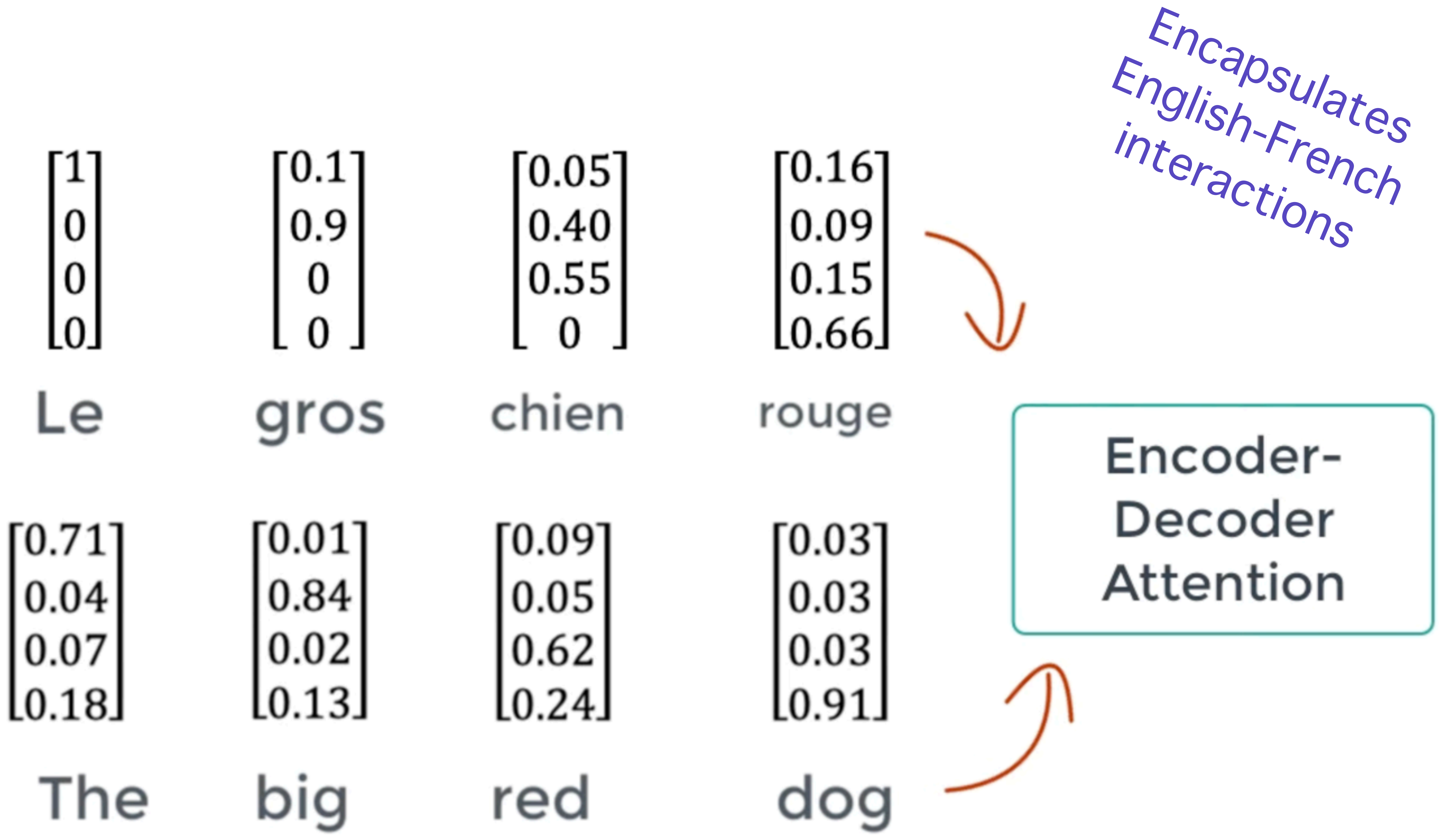


Figure 1: The Transformer - model architecture.

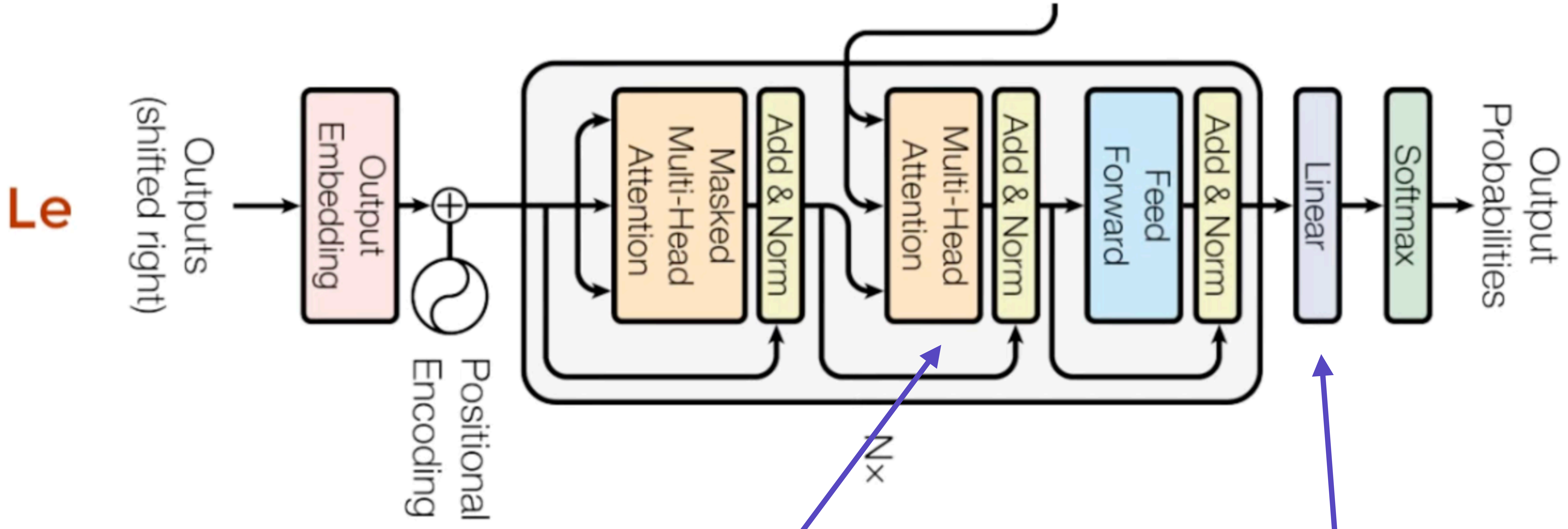
Self attention



Cross attention



Cross attention



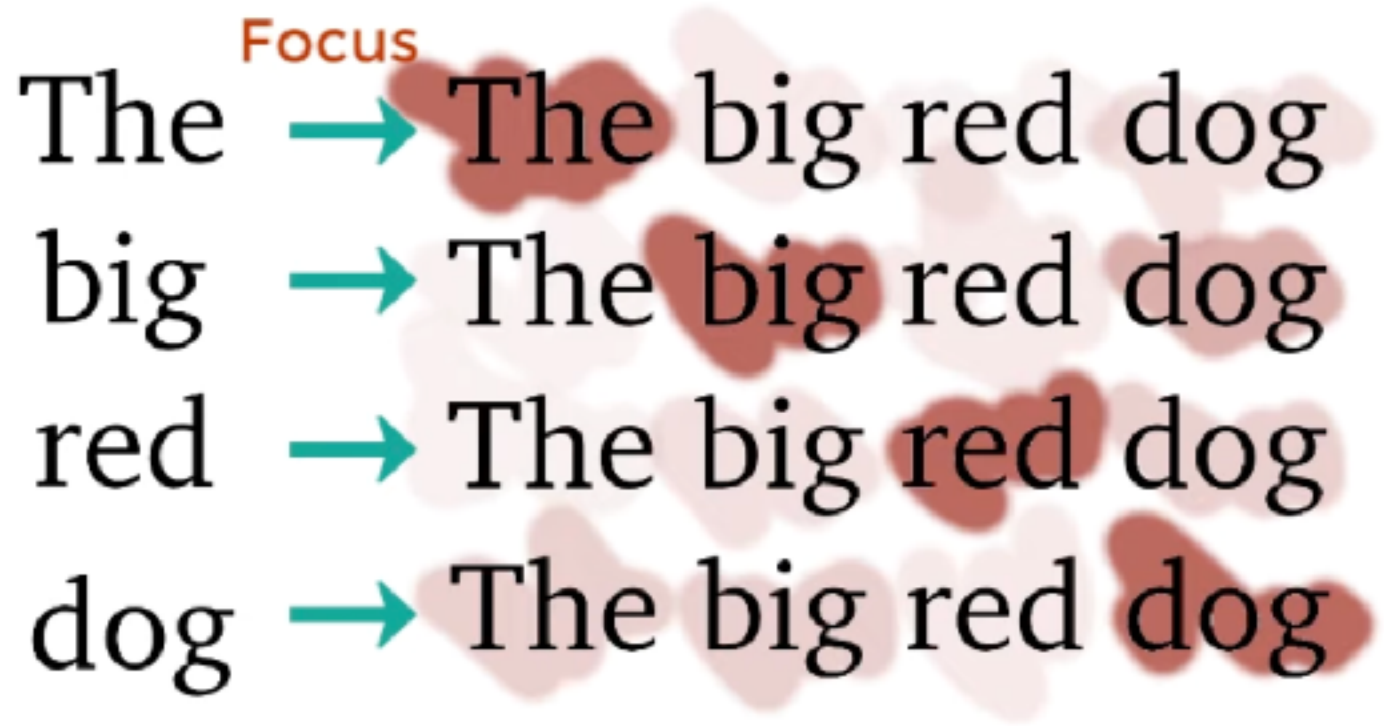
Cross attention
(English-French)

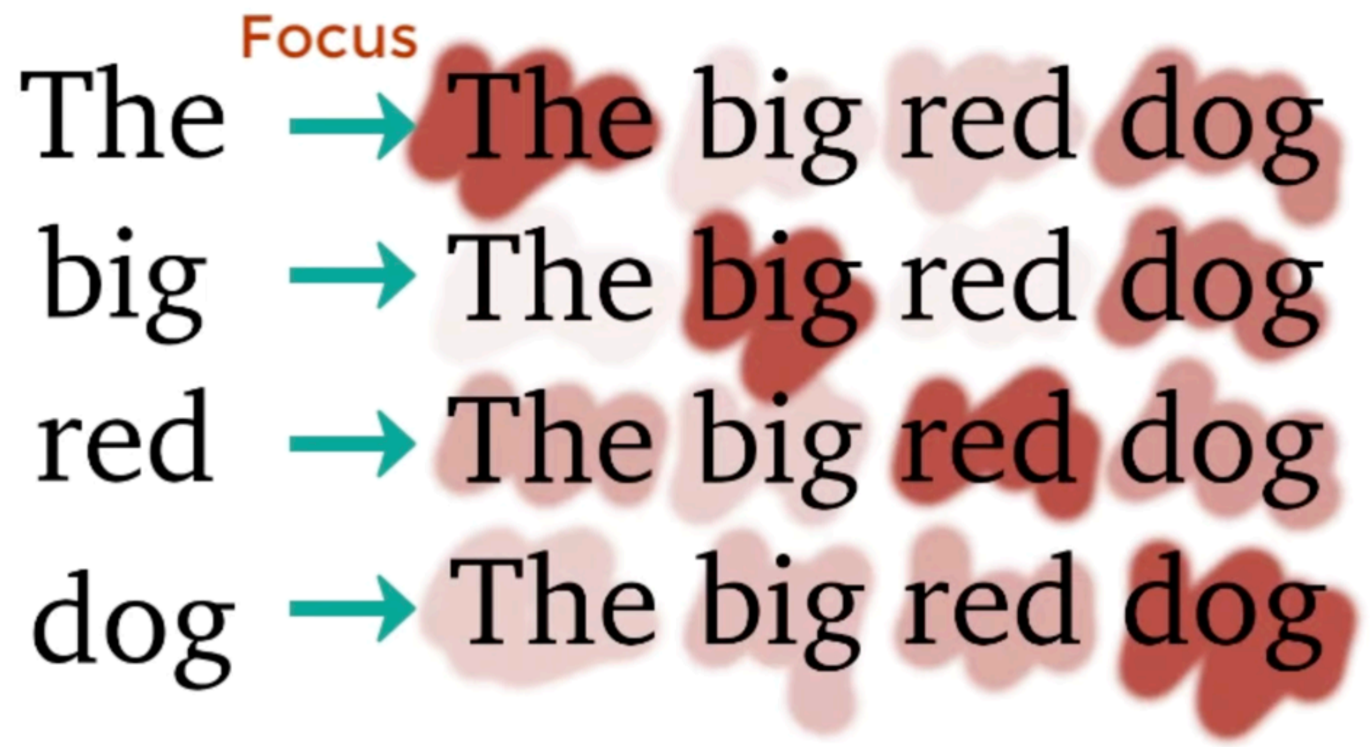
outputs
= # French words

That’s pretty much it. Now let’s look at some details we dropped

"Multihead" attention

- There can be multiple relationships to learn
  - Positional
    - "Is there" - question. "There is" - affirmative
  - Subject verb relationship
- Let's have multiple attentions
  - Multihead attention
- We'll combine all of them once they are computed

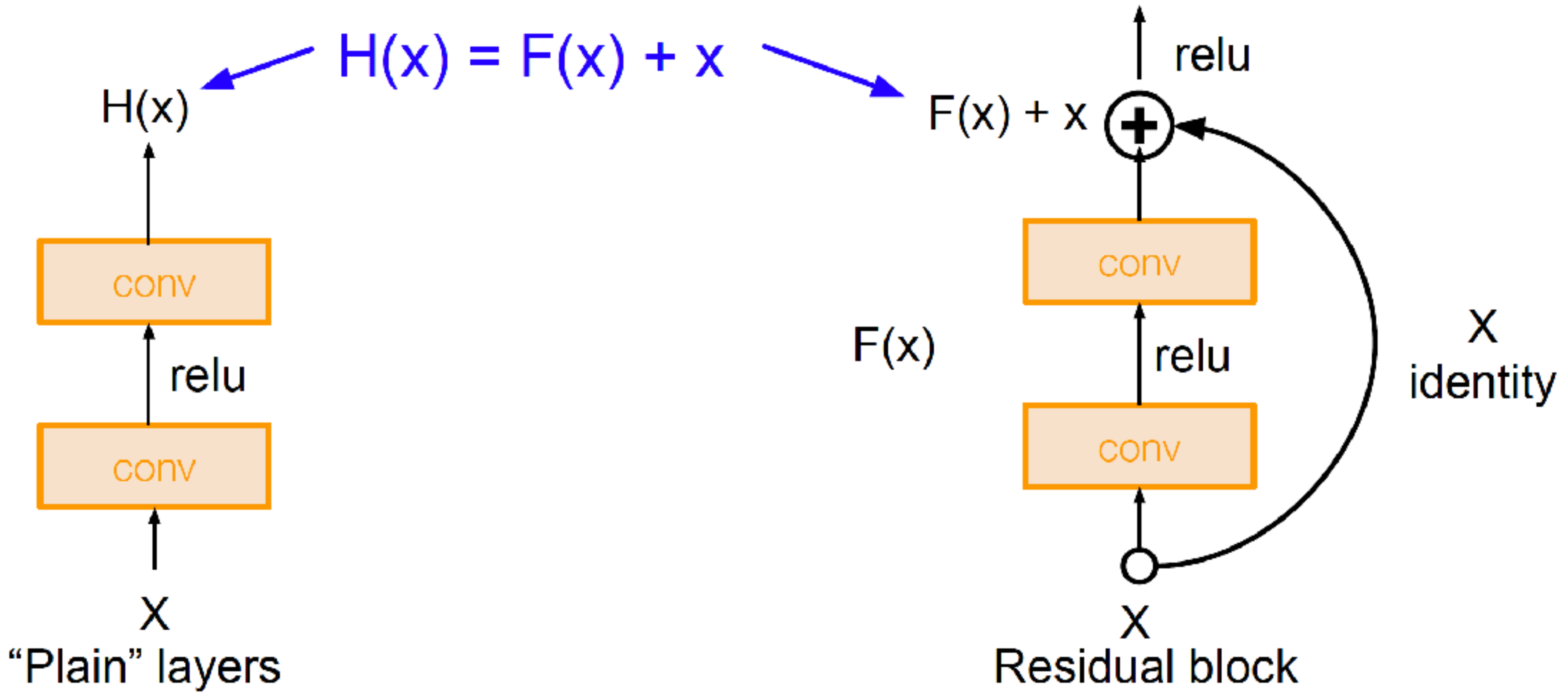




“Masked” attention

- The initial problem we talked about
  - English: The big red dog.
  - French: Le gros chien rouge
- But, when we start we only know the first French word
  - While computing attention, during training, we only need to look at the first word
  - Mask the rest → Masked attention

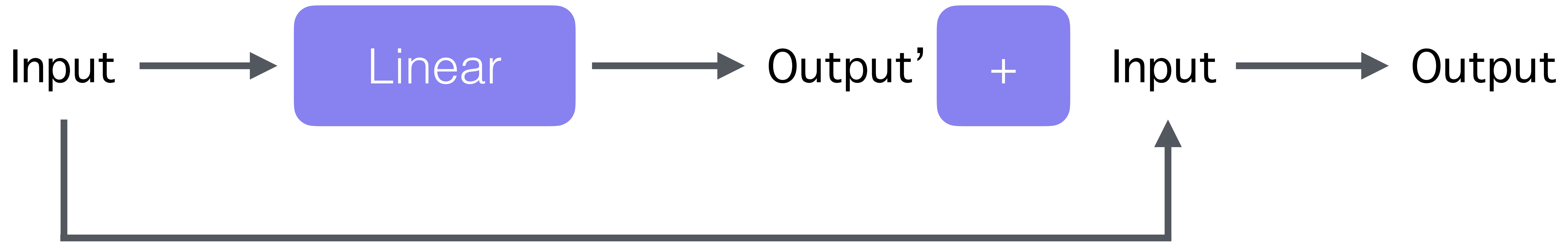
Add & Norm (RESIDUAL NETWORK)



Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

Add & Norm

Add = skip connections



- Helps remembering where it started from
- Useful in deeper networks (in general)

Norm = normalize (layer-wise or batch-wise)

Now we understand the principle

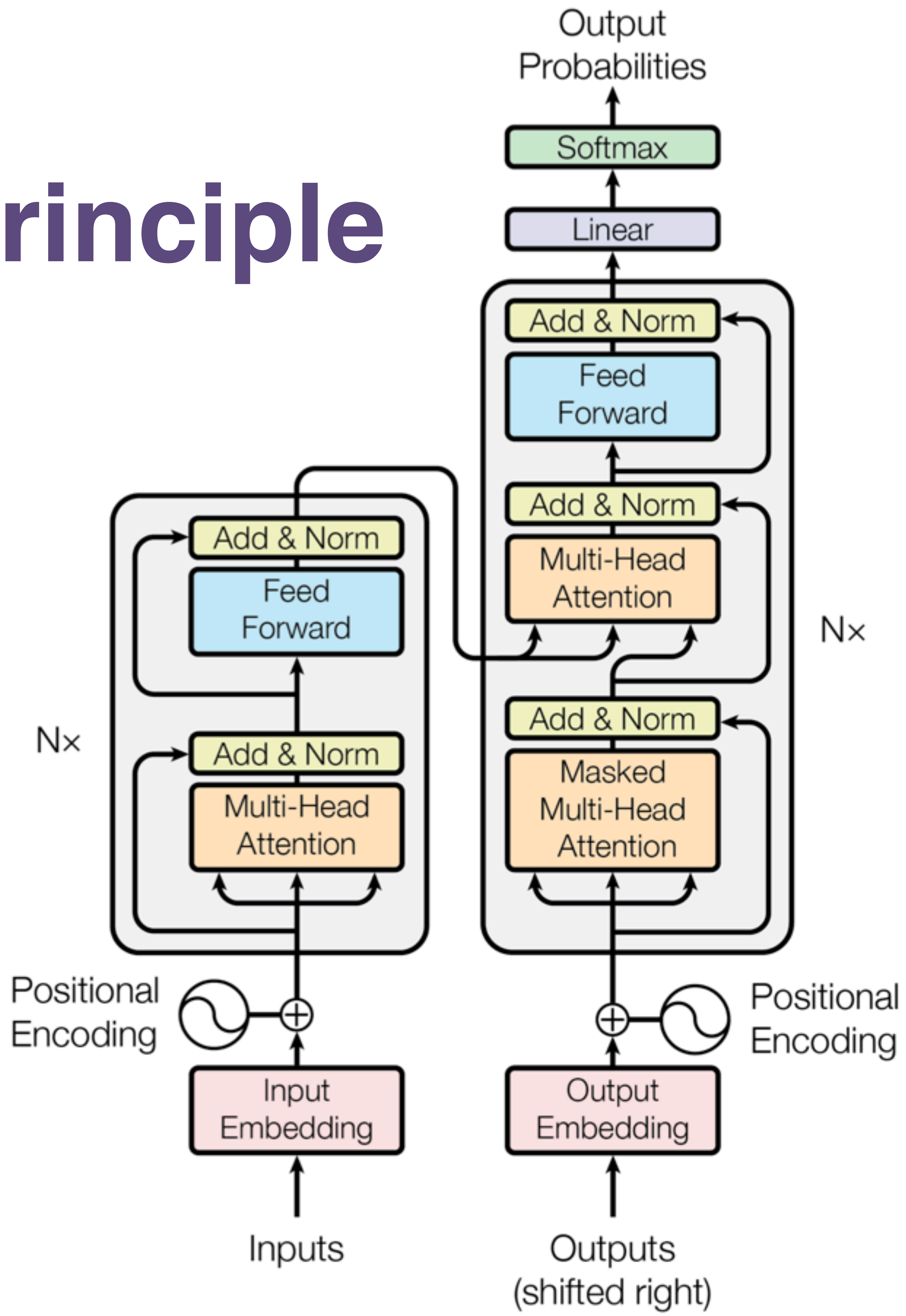


Figure 1: The Transformer - model architecture.

Attention is All You Need

Level 3 E. Gross

[Jay Alammar: The Illustrated Transformer]

[Mehreen Saeed: Positional Encoding]

https://arxiv.org/abs/1706.03762

Recurrent Neural Net in a NutShell

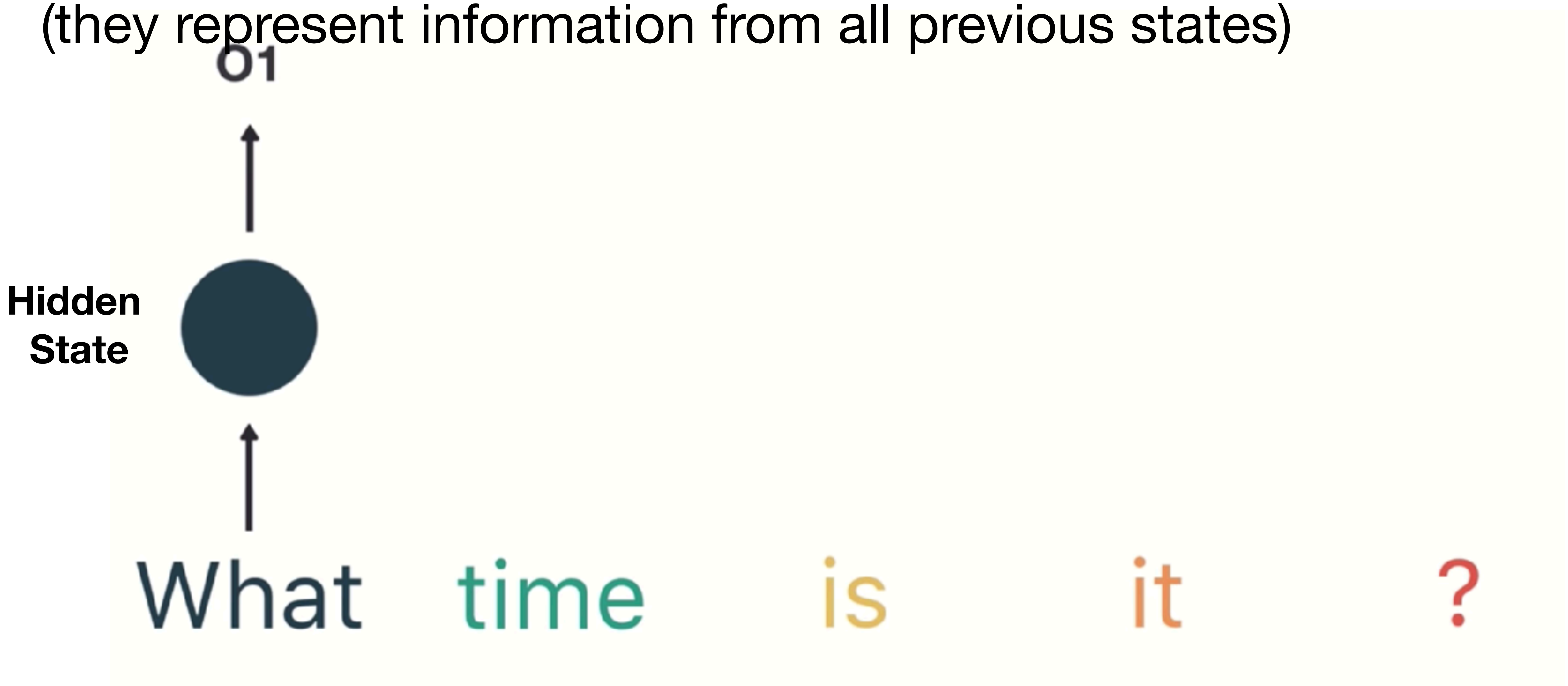
RNN's are good at processing sequence data for predictions

https://www.youtube.com/watch?v=LHXXI4-IEns

Recurrent Neural Net in a NutShell

- How do we do it?

- We use hidden states as memory
  (they represent information from all previous states)

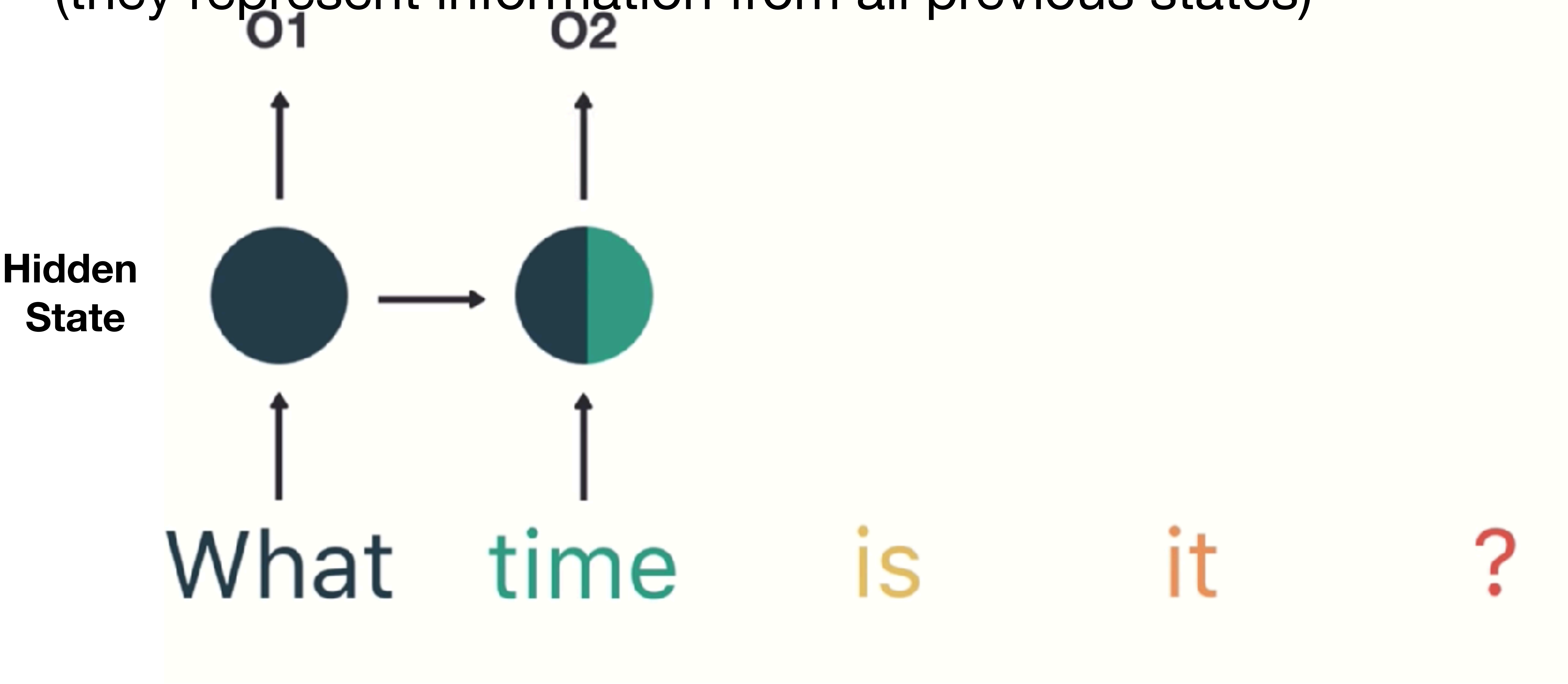


https://www.youtube.com/watch?v=LHXXI4-IEns

Recurrent Neural Net in a NutShell

- How do we do it?

- We use hidden states as memory
(they represent information from all previous states)

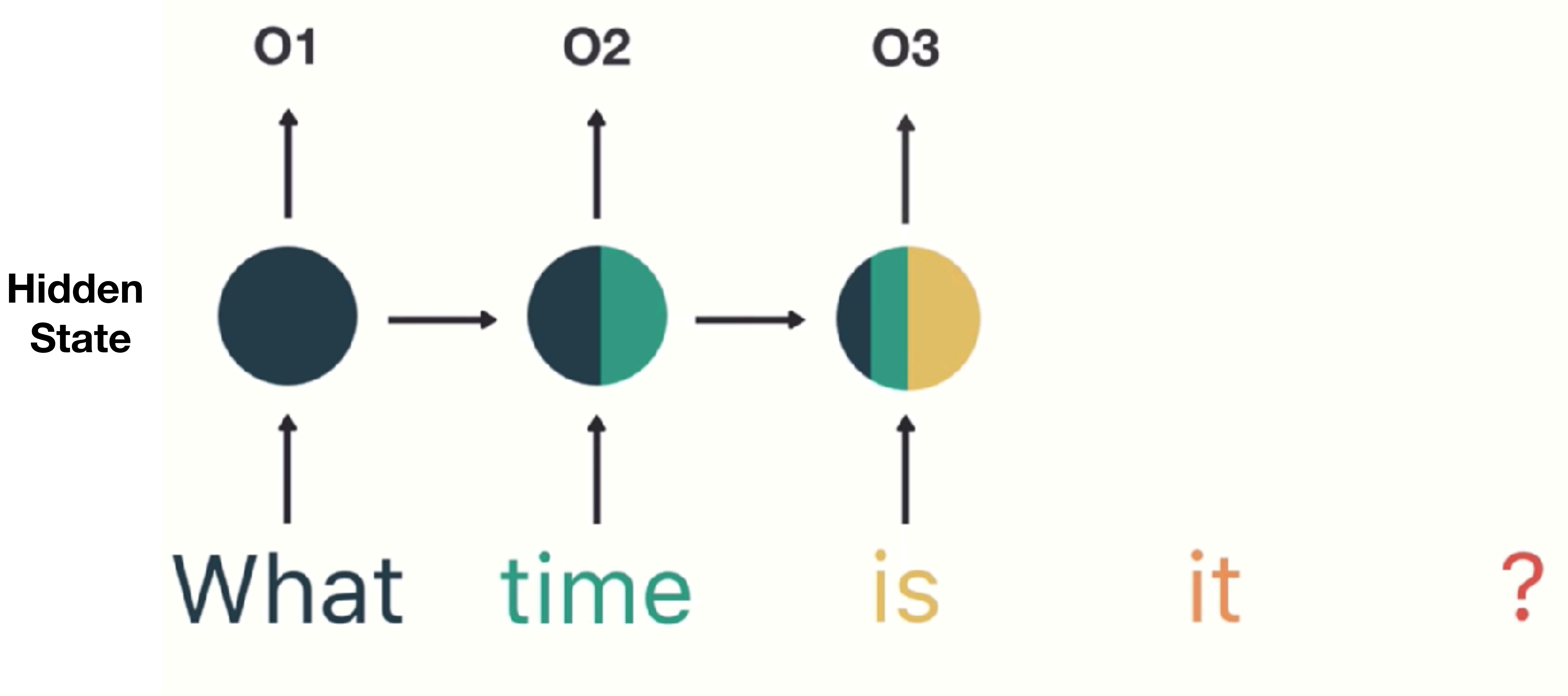


https://www.youtube.com/watch?v=LHXXI4-IEns

Recurrent Neural Net in a NutShell

- How do we do it?

- We use hidden states as memory

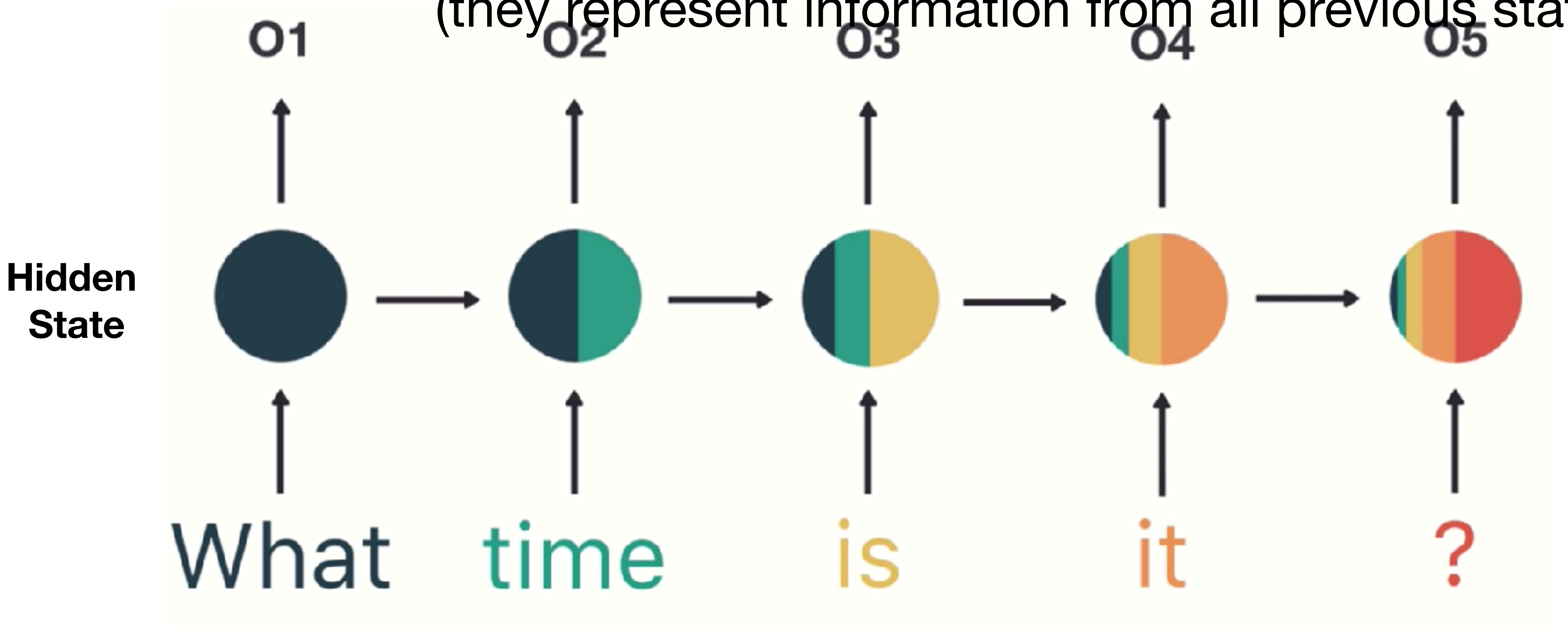


https://www.youtube.com/watch?v=LHXXI4-IEns

Recurrent Neural Net in a NutShell

- How do we do it?

- We use hidden states as memory
  (they represent information from all previous states)



https://www.youtube.com/watch?v=LHXXI4-IEns

Recurrent Neural Net in a NutShell

- Issue of RNN with Short Time Memory

https://www.youtube.com/watch?v=LHXXI4-IEns

Recurrent Neural Net in a NutShell

- Issue of RNN with Short Time Memory

- Back propagation with time —> Vanishing Gradient
  We do not learn very early layers….

https://www.youtube.com/watch?v=LHXXI4-IEns

Transformer Keypoints

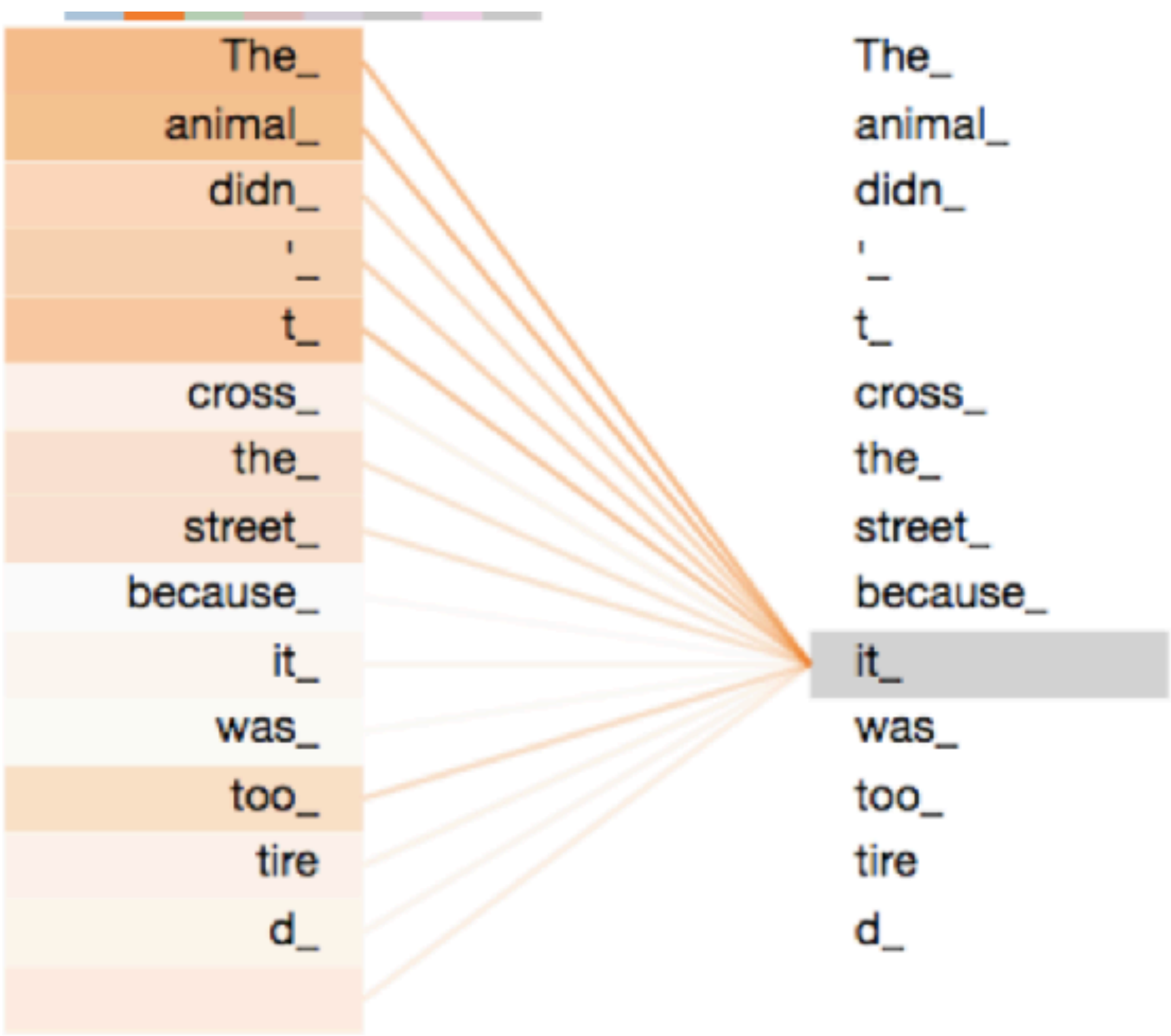
- RNN: Seq to Seq
  Suffers from long term memory
  Transformer not sequential like RNN
  All input fed once through the model and calculation is performed once

- Introduce the concept of Self Attention

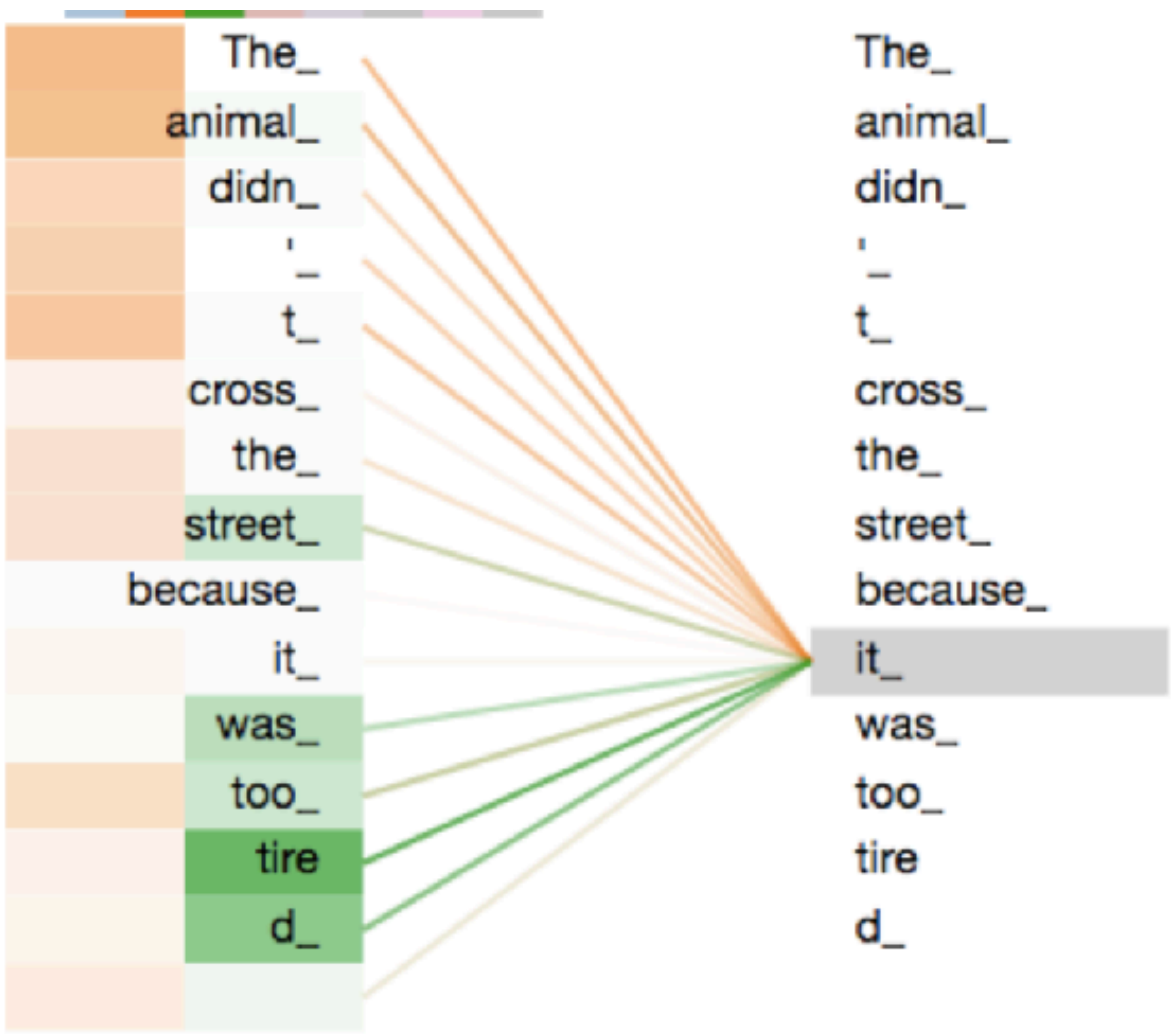
- Multi Head Attention

  God commanded Abraham to sacrifice his son in order to test his faith

One Attention



two Heads



http://jalammar.github.io/illustrated-transformer/

paper :

$$d_{model} = 512 \qquad d_K = d_V = \frac{d_{model}}{h} \qquad d_K = d_V = \frac{512}{8} = 64$$

$$d_K = d_V = \frac{d_{model}}{h}$$

$$X \in R^{N \times d_{model}} \qquad W^K, W^Q \in R^{d_{model} \times d_k} \qquad Q, K \in R^{N \times d_k}$$

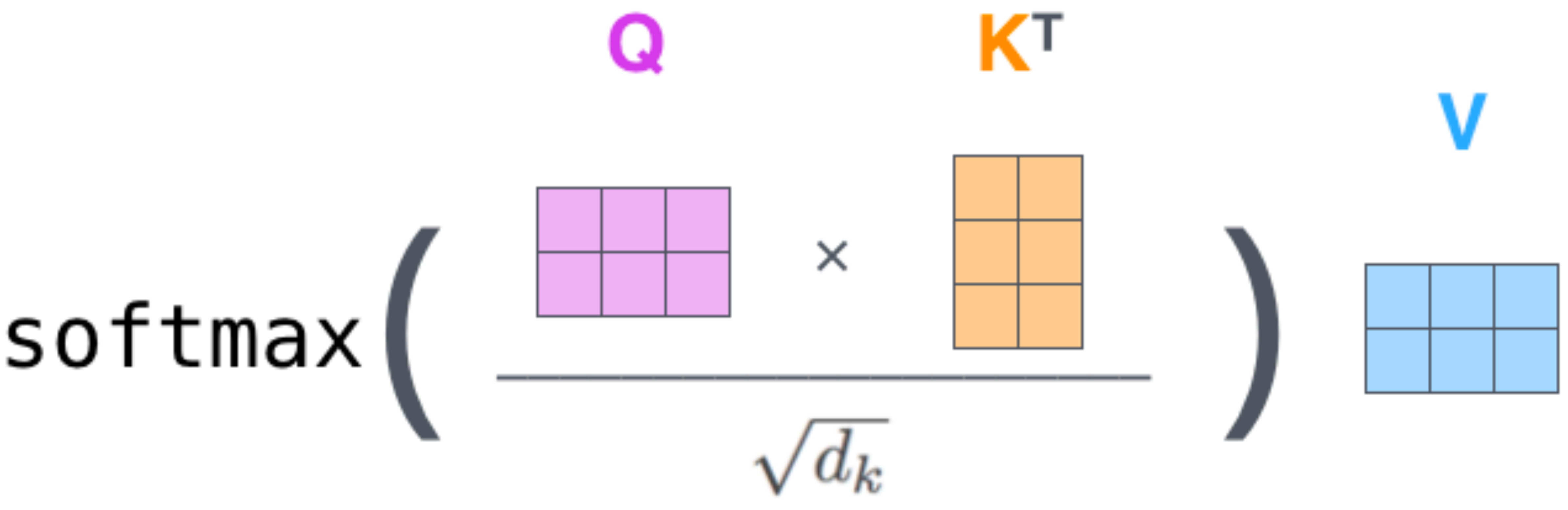
$$X \in R^{N \times d_{model}} \qquad W^V \in R^{d_{model} x d_v} \qquad V \in R^{N \times d_v}$$

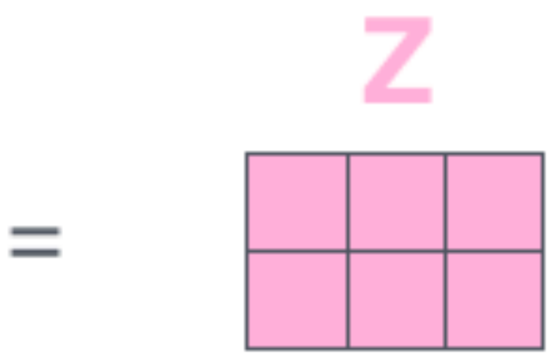
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$Z_i = \Sigma_\ell softmax\left(\frac{1}{\sqrt{d_k}} Q_i \cdot K_\ell^T\right) V_\ell$$

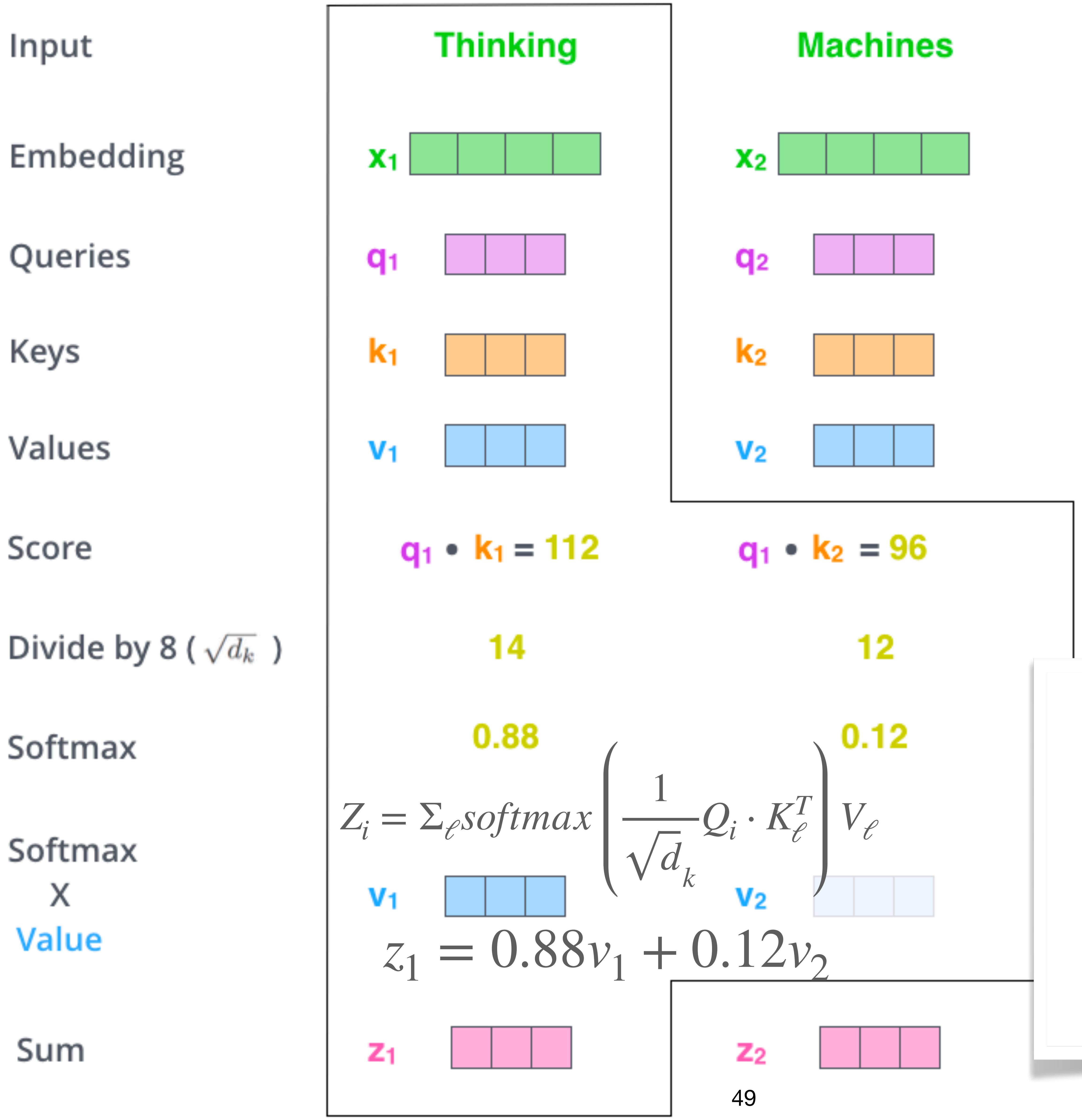
$X \in R^{N \times d_{model}}$ $\quad$ $Q, K \in R^{N \times d_k}$

$Q \times K^T \in R^{N \times N}$ $\quad$ $V \in R^{N \times d_v}$



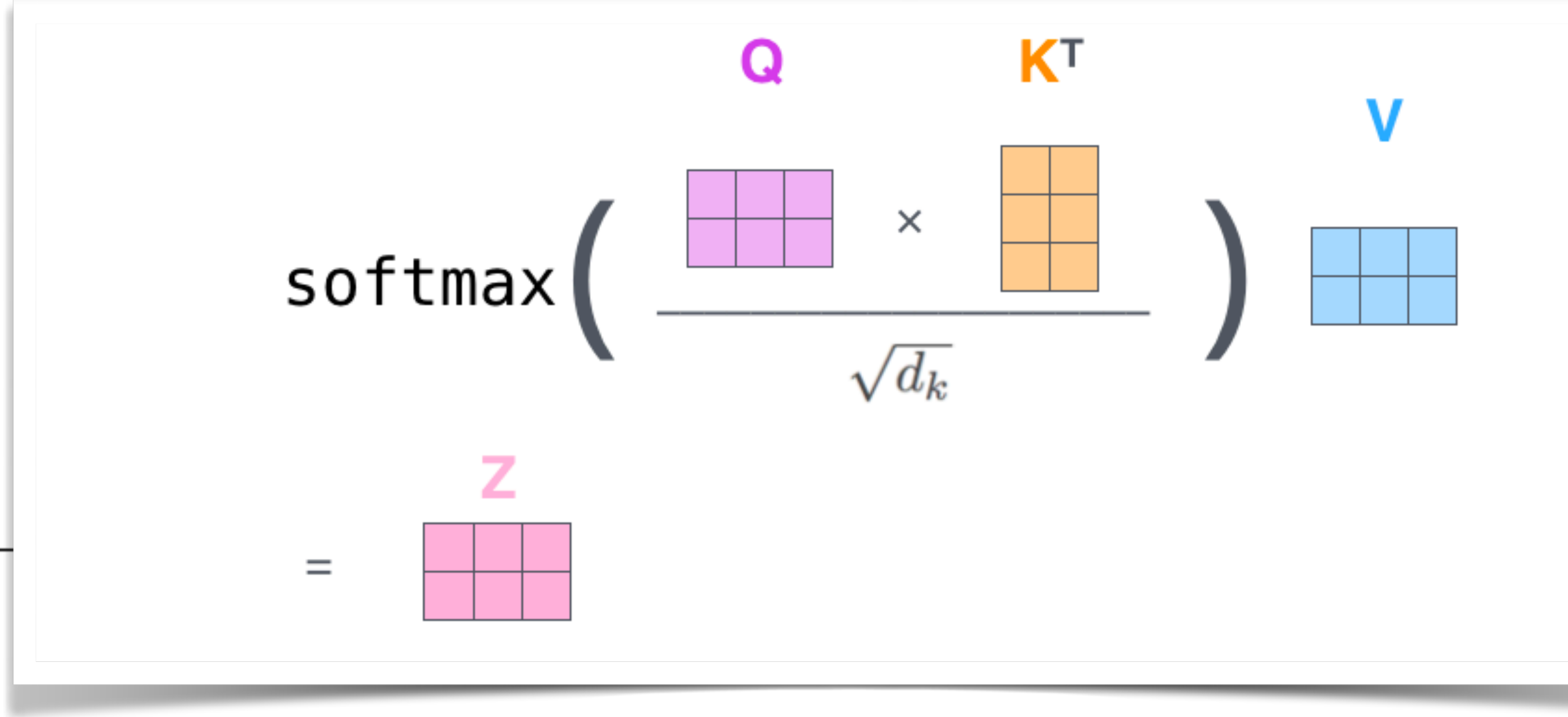


$Z \in R^{N \times d_V}$



$$Z_i = \Sigma_\ell softmax\left(\frac{1}{\sqrt{d_k}} Q_i \cdot K_\ell^T\right) V_\ell$$

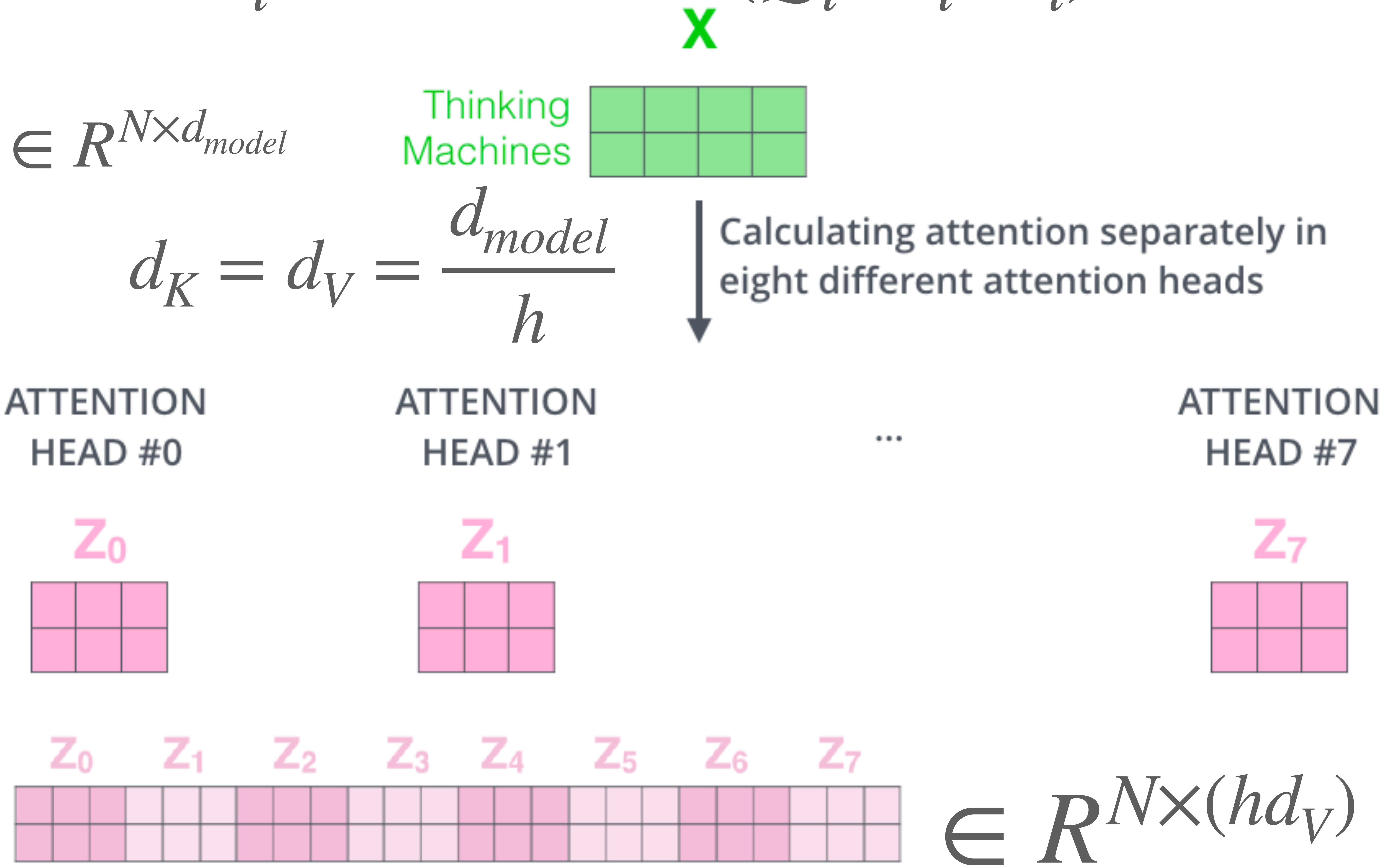
$$z_1 = 0.88 v_1 + 0.12 v_2$$



$$head_i = Attention(Q_i, K_i, V_i)$$

$$X \in R^{N \times d_{model}}$$

$$d_K = d_V = \frac{d_{model}}{h}$$



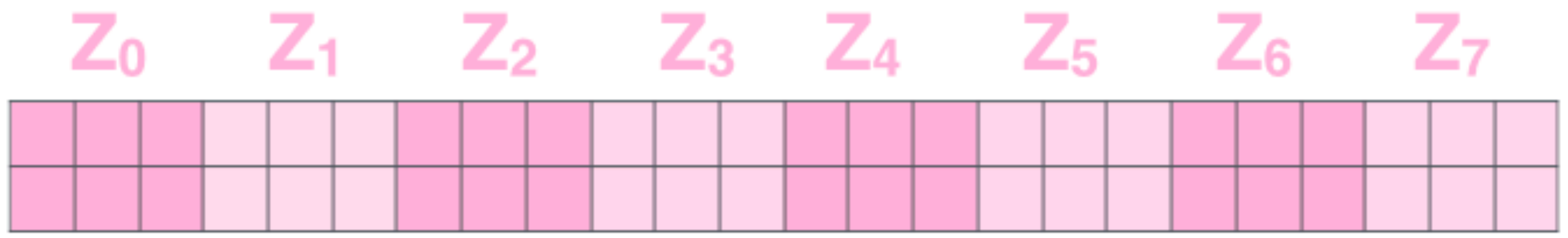
$$Z \in R^{N \times d_V}$$

$$\in R^{N \times (hd_V)} = R^{N \times d_{model}}$$

$$\mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)$$

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$

1) Concatenate all the attention heads

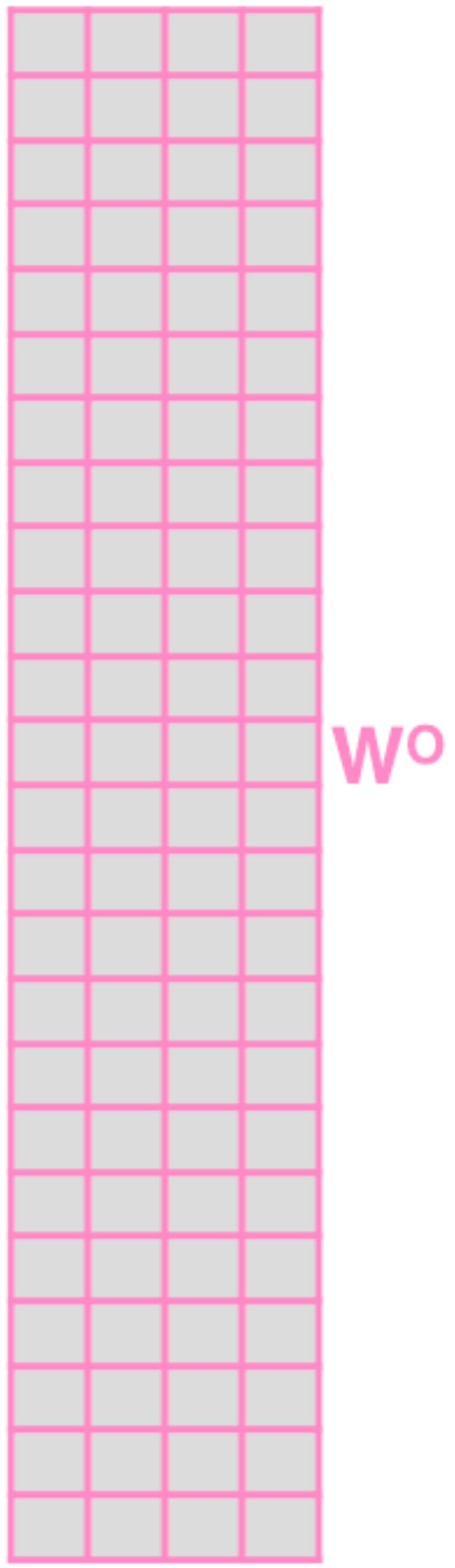


$$\in R^{N \times h d_V}$$

2) Multiply with a weight matrix W^O that was trained jointly with the model

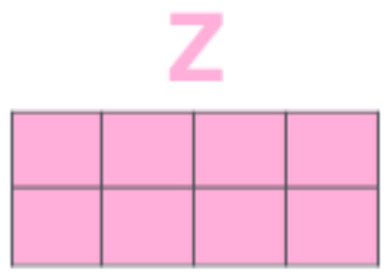
x

$$W^O \in R^{h d_V \times d_{model}}$$



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



$$Z \in R^{N \times d_{model}}$$

1) This is our input sentence*

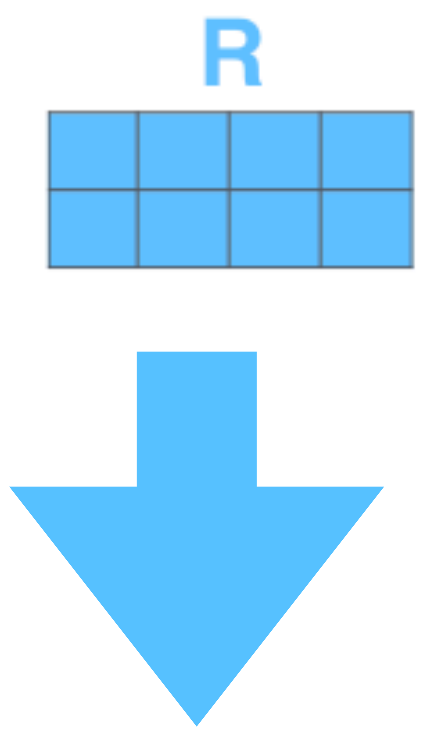
2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

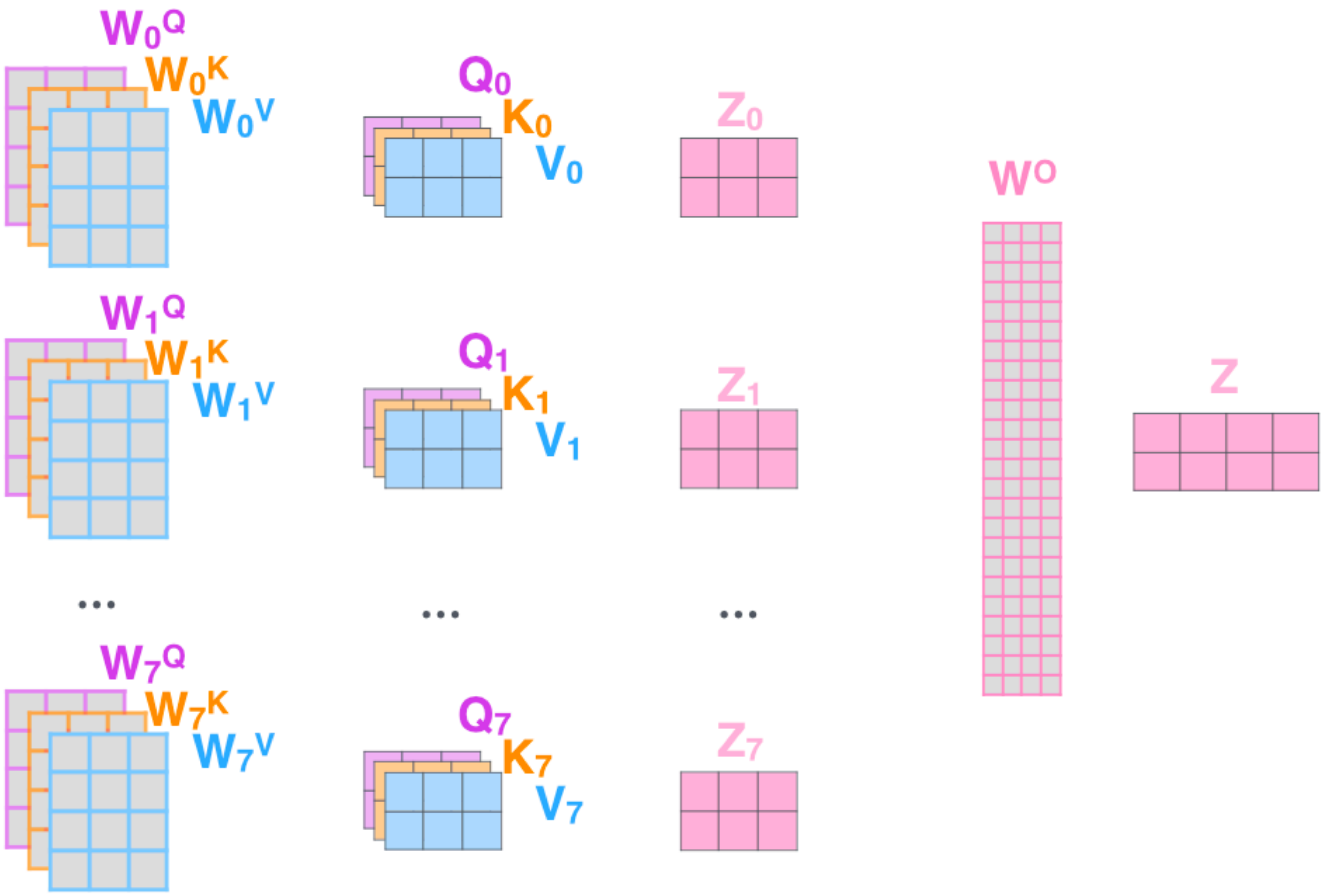
4) Calculate attention using the resulting Q/K/V matrices

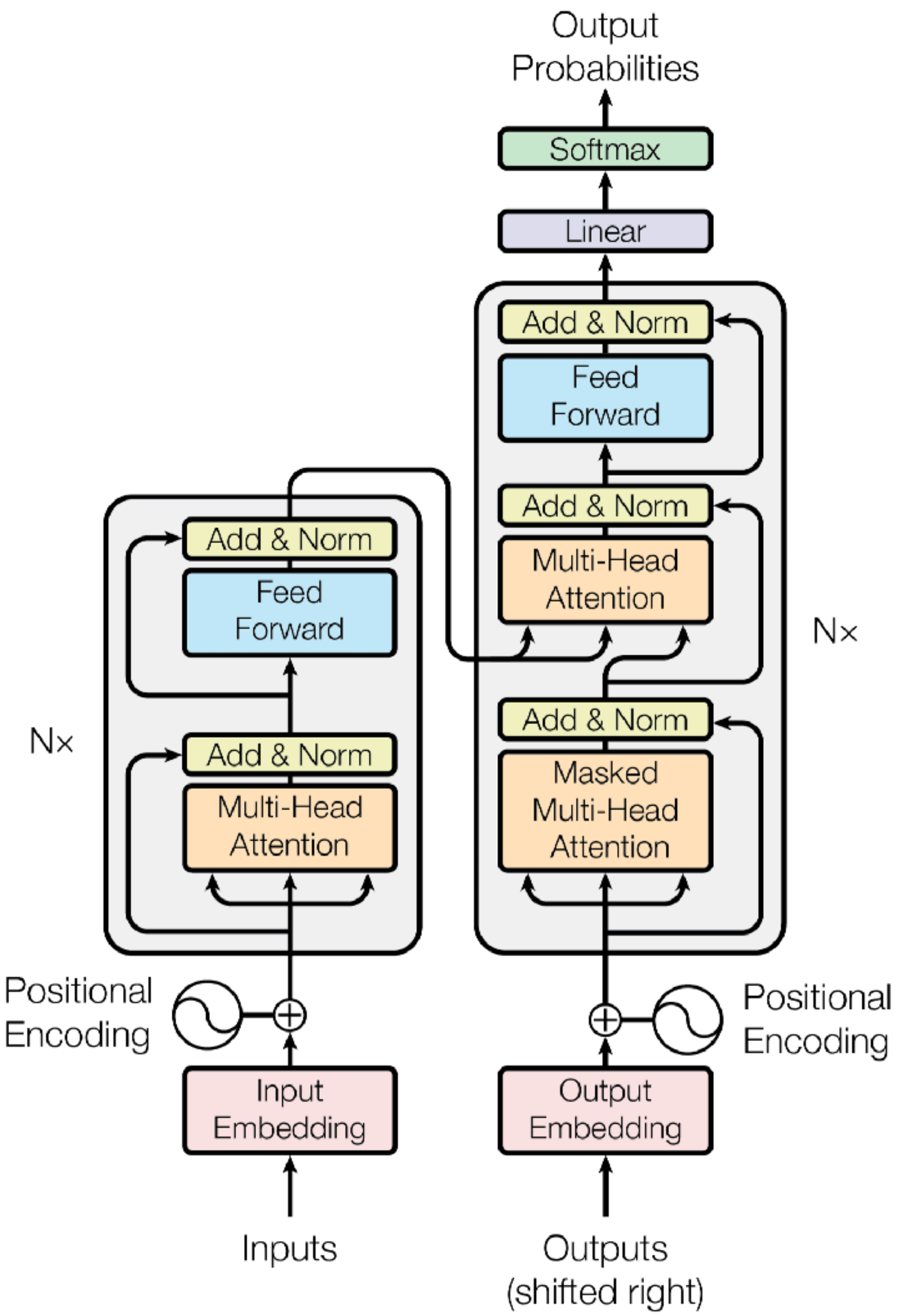
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

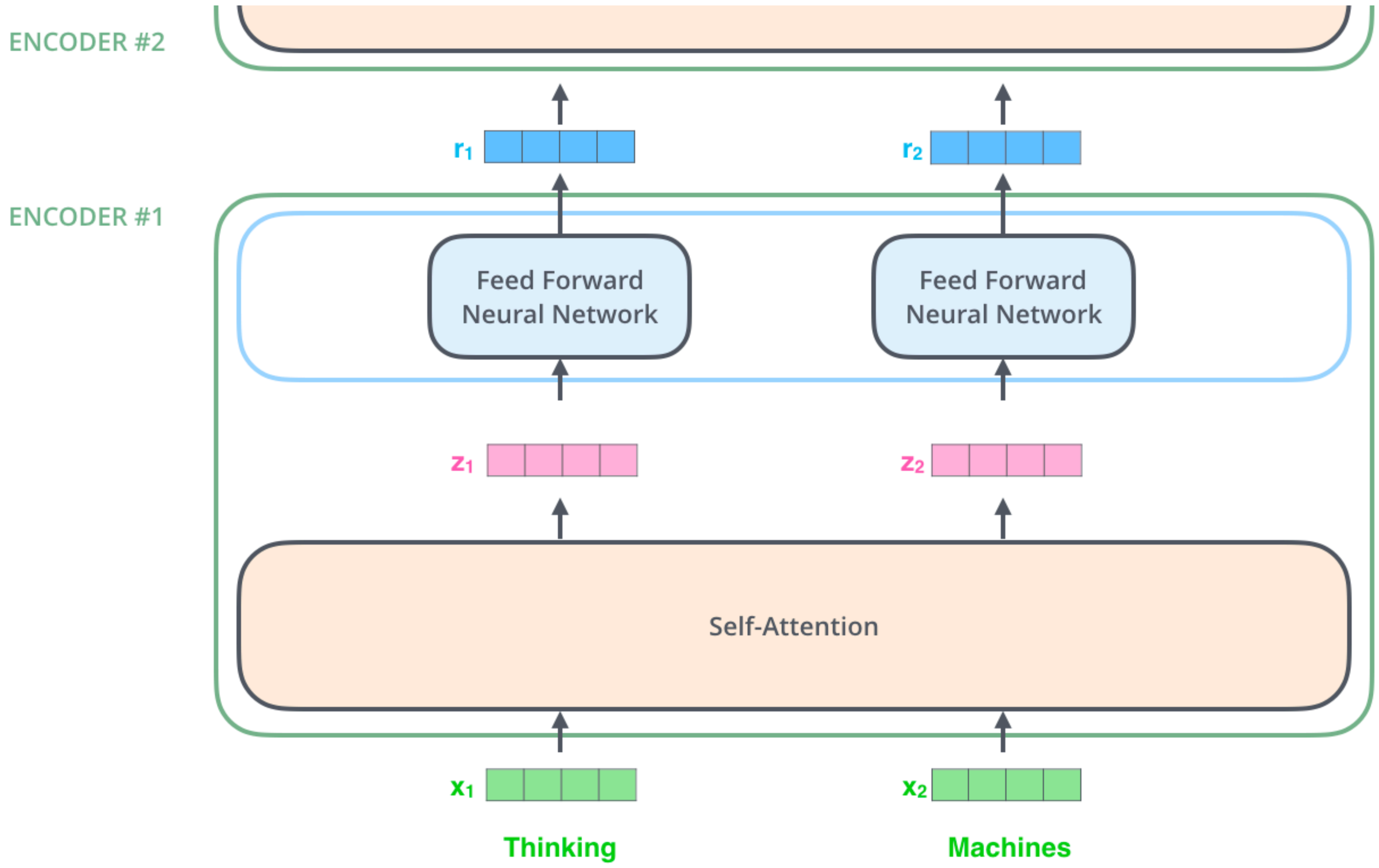


next page




Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

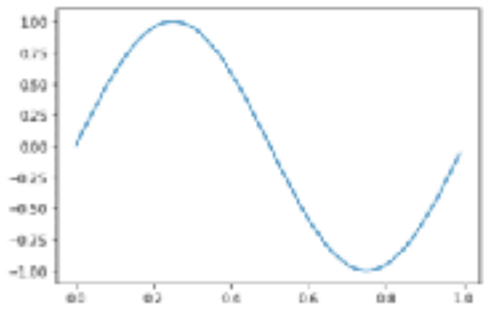
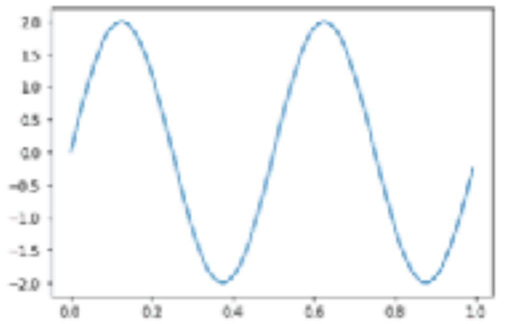
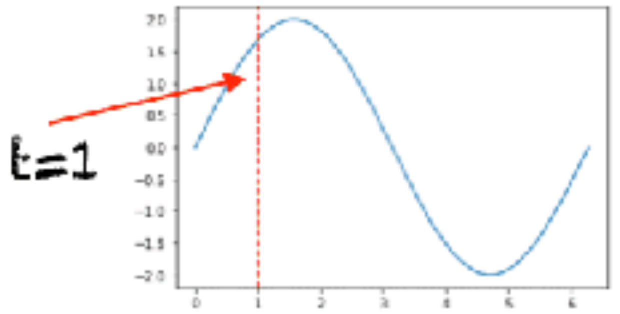
next page


ENCODER #2
r1
r2
ENCODER #1
Feed Forward
Neural Network
Feed Forward
Neural Network
z1
z2
Self-Attention
x1
x2
Thinking
Machines

Position ENCODING

Sequence Length = L Varies
2i=0,…..,L

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Equation	Graph	Frequency	Wavelength
$\sin(2\pi t)$		1	1
$\sin(2 * 2\pi t)$		2	1/2
$\sin(t)$		$1/2\pi$	2π
$\sin(ct)$	Depends on c	$c/2\pi$	$2\pi/c$

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Position ENCODING

Sequence Length = L Varies
2i=0,…..,d_model

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, k	i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

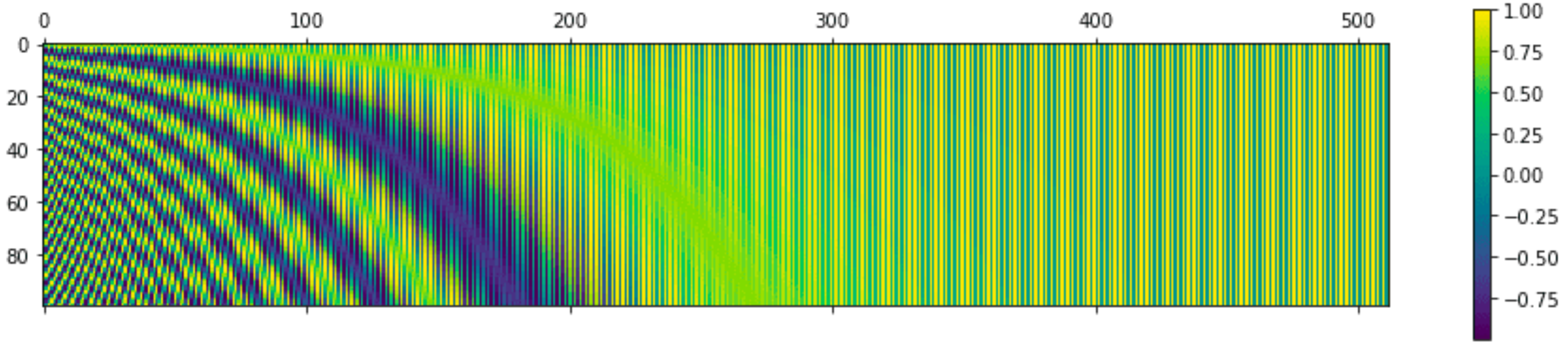
Positional Encoding Matrix for the sequence 'I am a robot'

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Position ENCODING

Sequence Length = L Varies
2i=0,…..,d_model

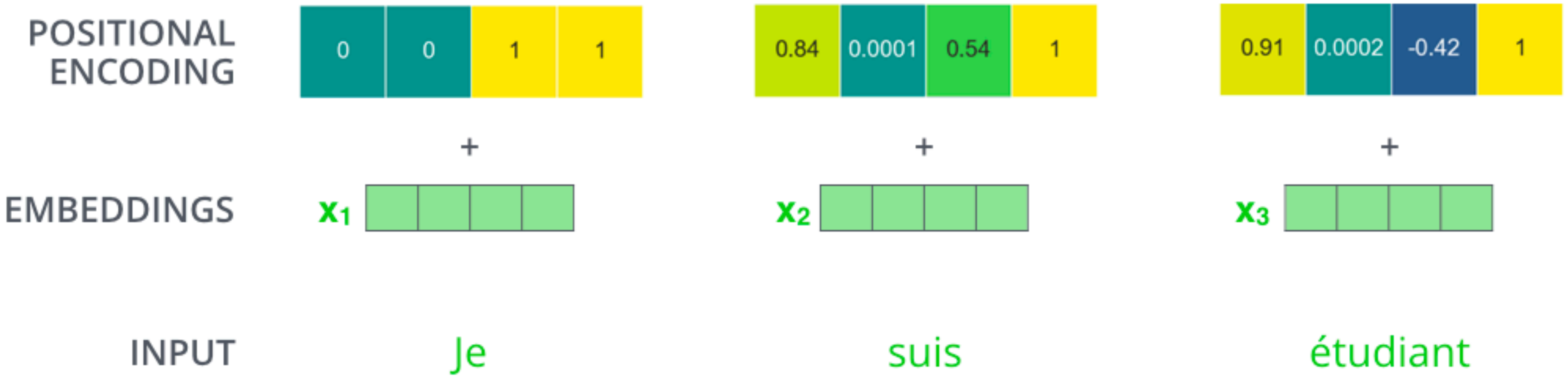
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



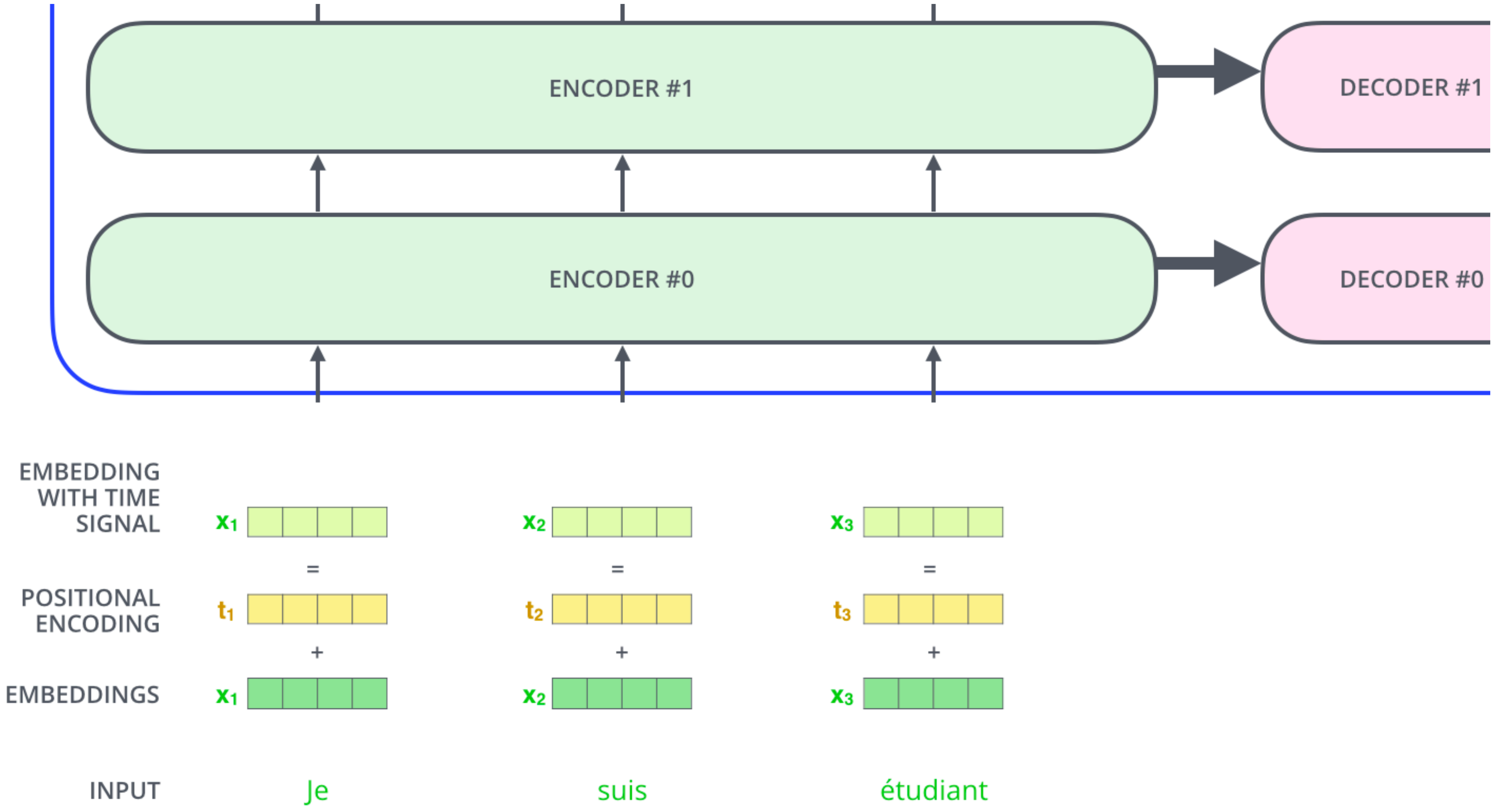
The positional encoding matrix for n=10,000, d=512, sequence length=100

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

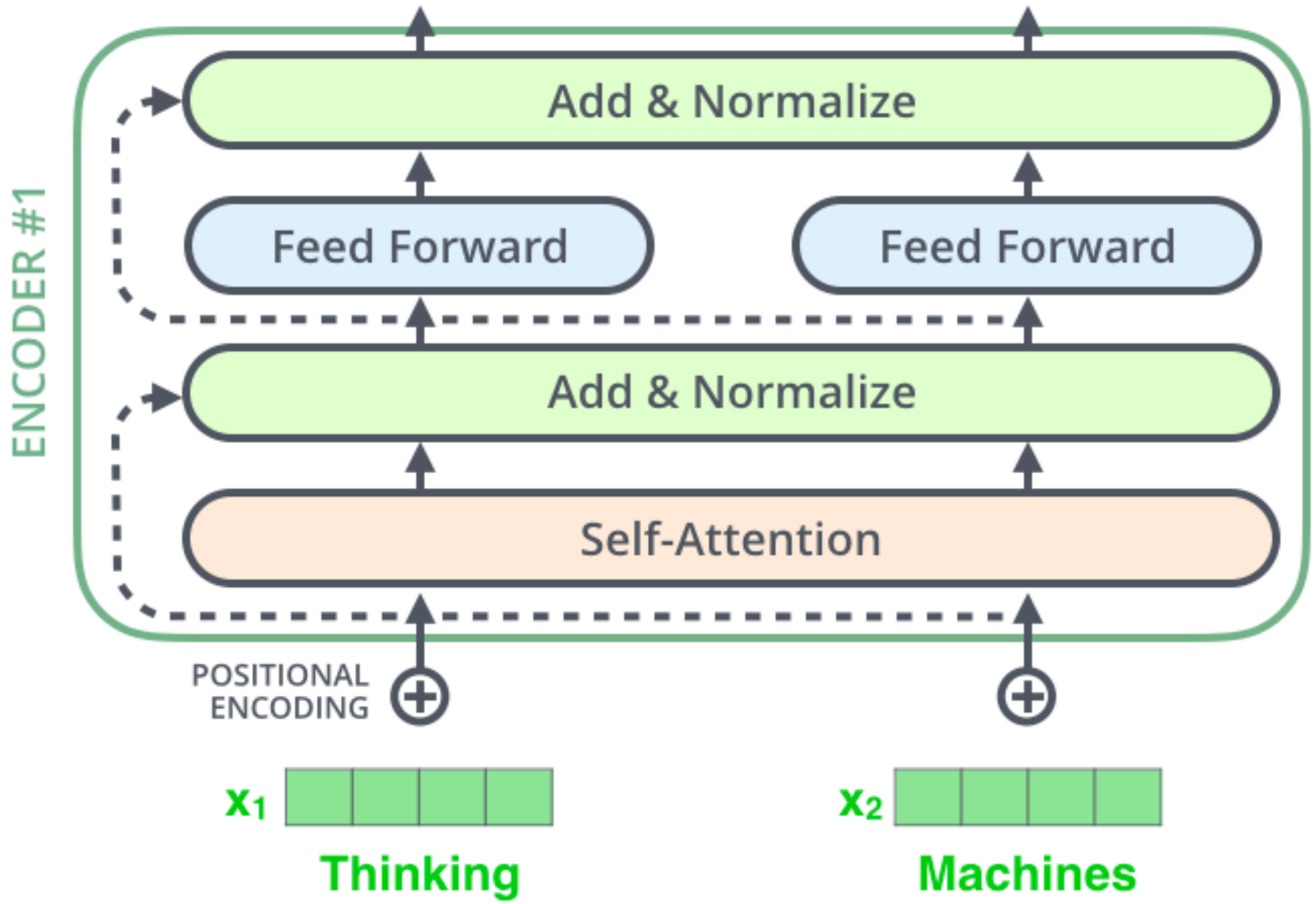
Position ENCODING



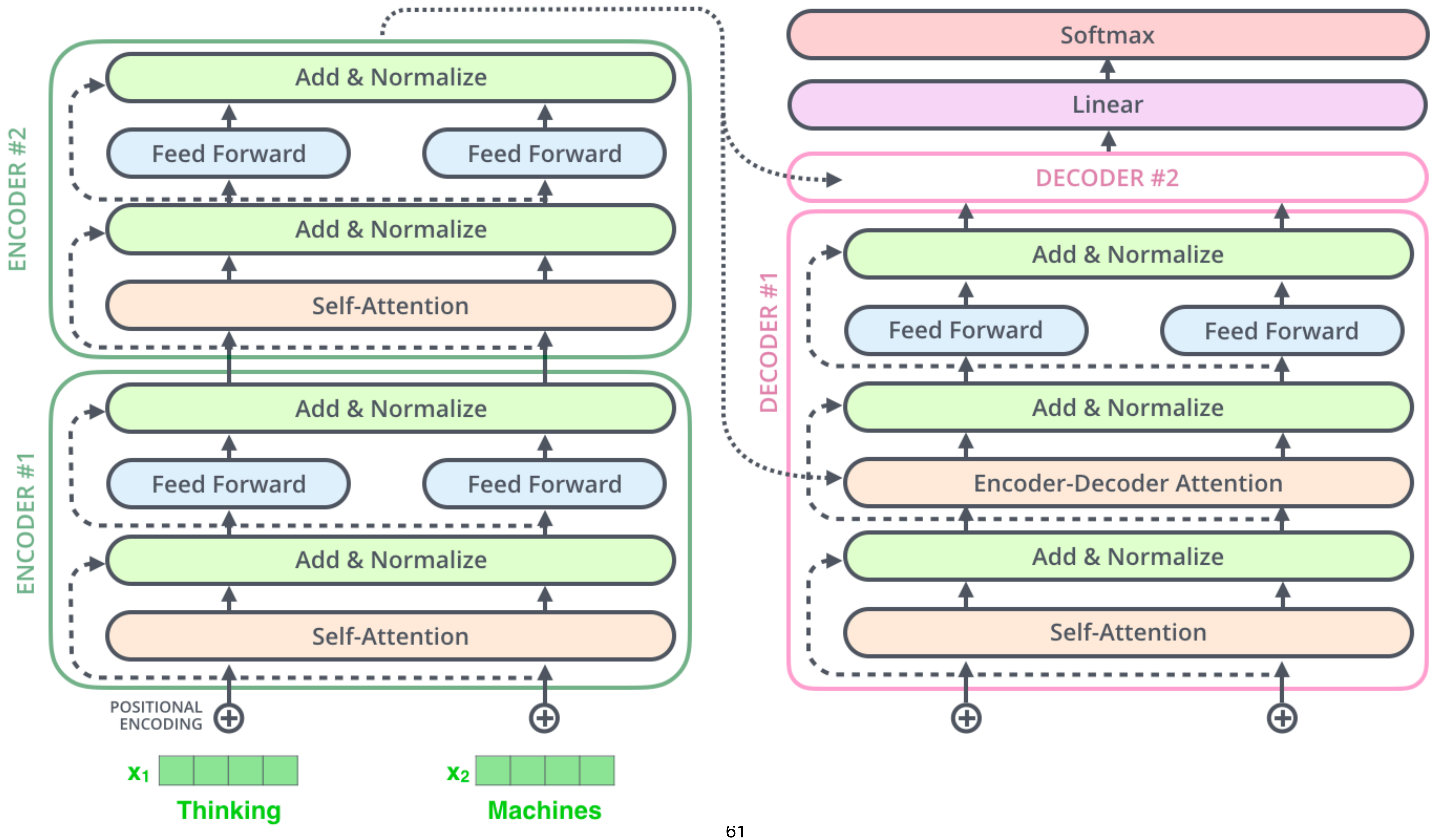
Position ENCODING



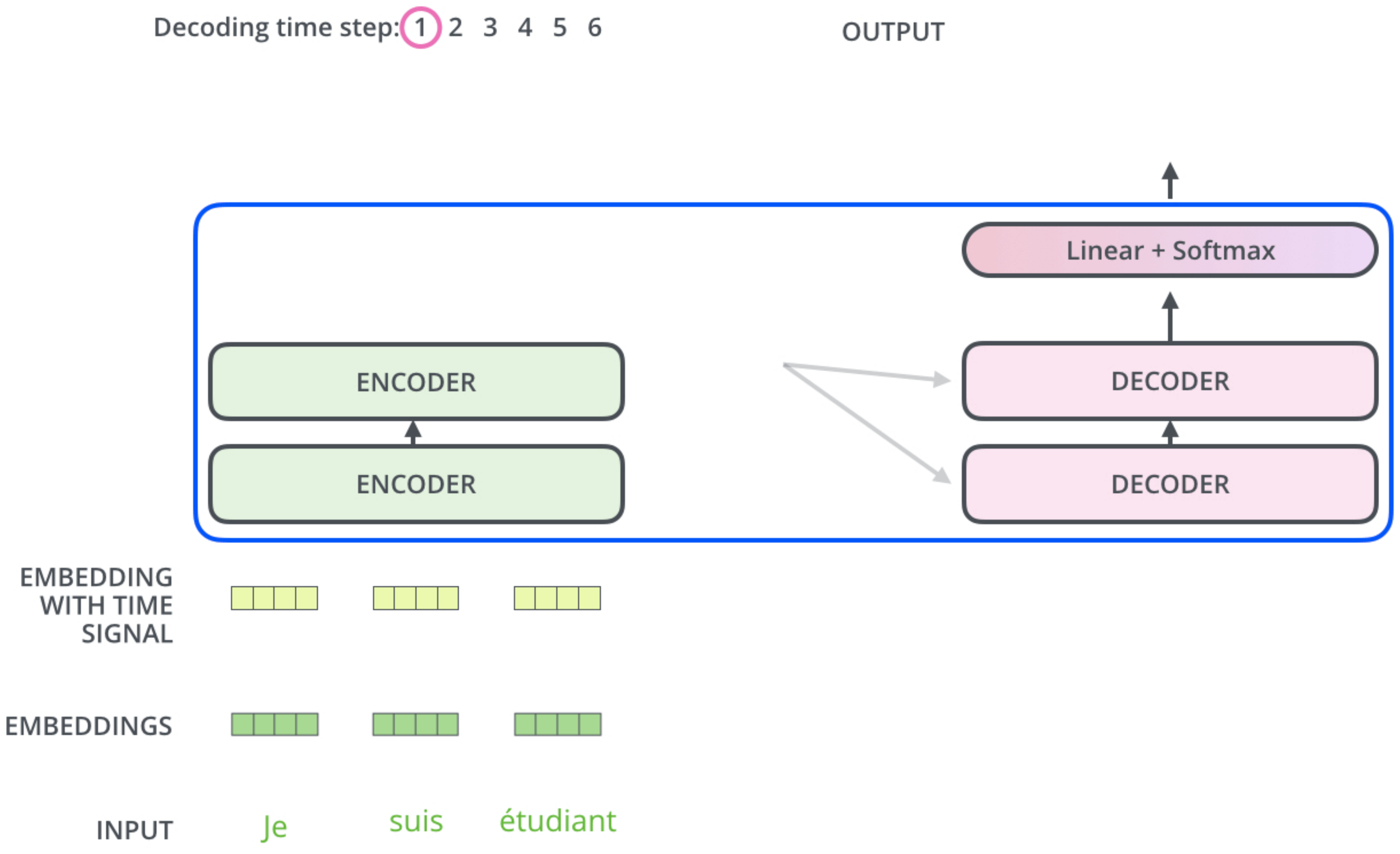
Residual Net



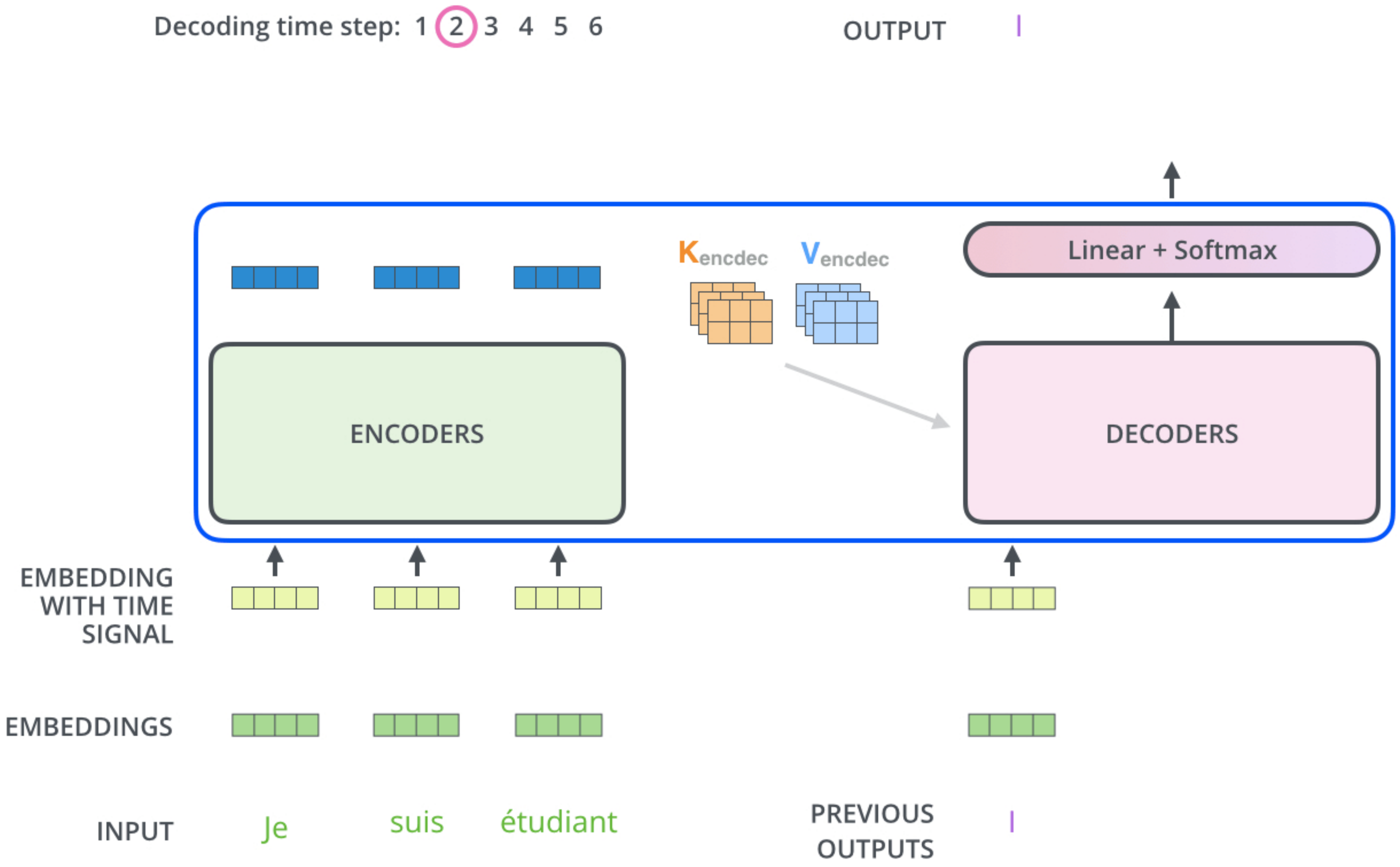
Stack 6 Encoders & Decoders



Cross Attention

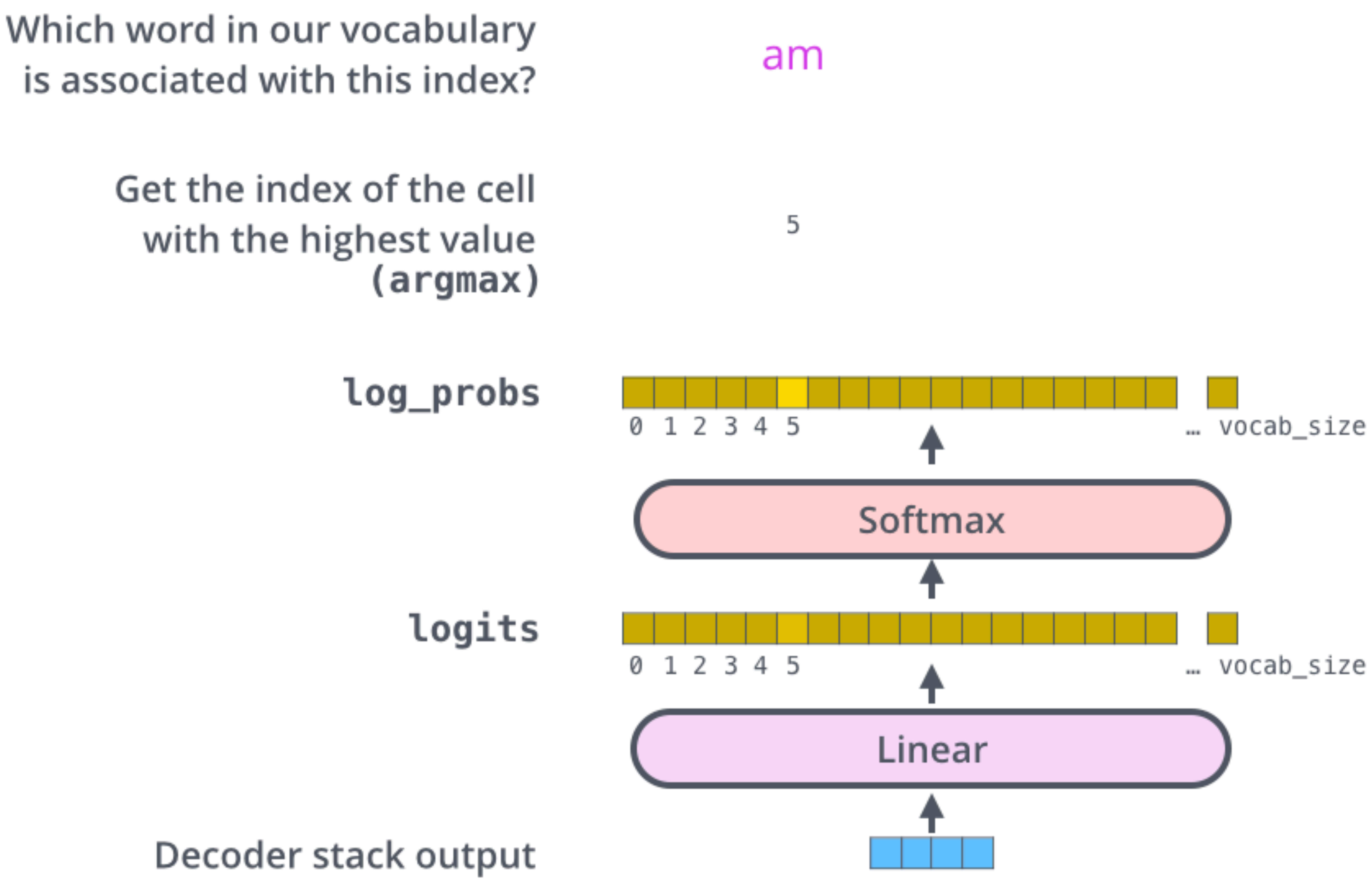


Decoder



Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Softmax


Which word in our vocabulary is associated with this index?
am
Get the index of the cell with the highest value (argmax)
5
log_probs
0 1 2 3 4 5 … vocab_size
Softmax
logits
0 1 2 3 4 5 … vocab_size
Linear
Decoder stack output

End of Lecture 2

✅ Seriously? Where is the Physics?

Fast Simulation for Particle Reconstruction with GNN and Slot Attention

Conditional Generative Modelling of Reconstructed Particles at Collider Experiments

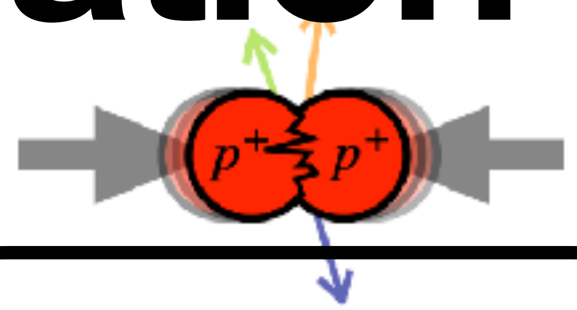
Francesco Armando Di Bello[1], Etienne Dreyer[2], Sanmay Ganguly[3], Eilam Gross[2], Lukas Heinrich[4], Marumi Kado[4,5], Nilotpal Kakati[2], Jonathan Shlomi[2], Nathalie Soybelman[2]

[1] University of Genova
[2] Weizmann Institue of Science
[3] ICEPP, University of Tokyo
[4] Technical University of Munich
[5] Sapienza University of Rome

Motivation

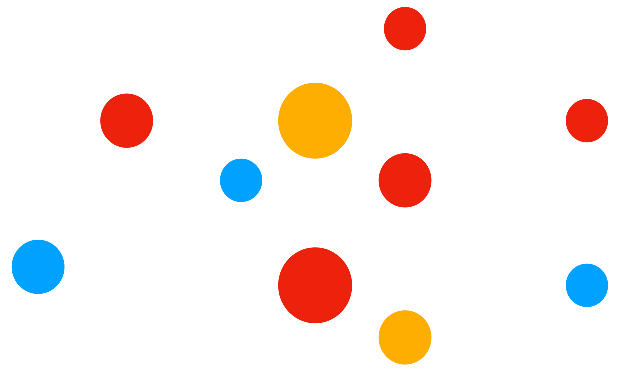
Data taking — 1000 events / sec

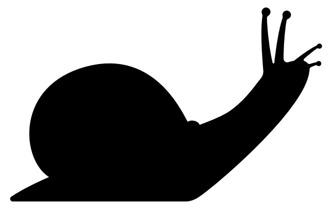
Simulation — 10 min / event

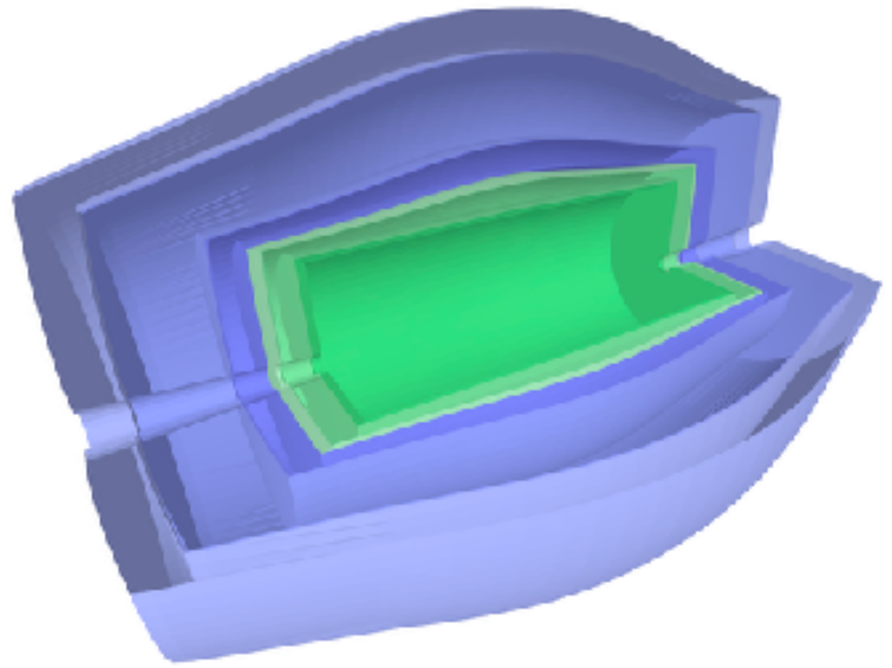


Collision Simulation

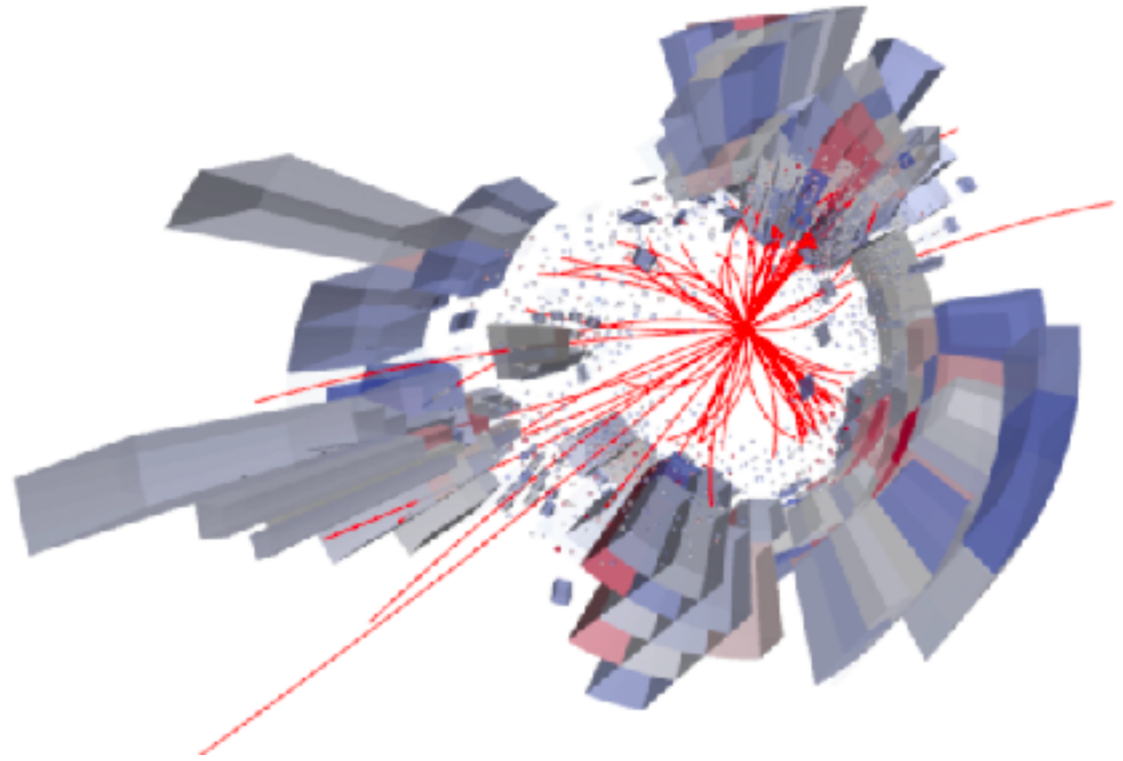
Final state particles





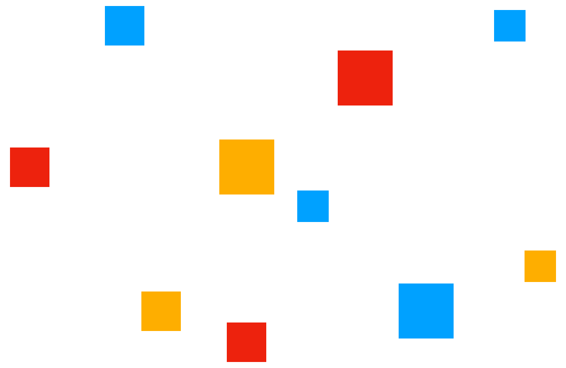


Detector Simulation

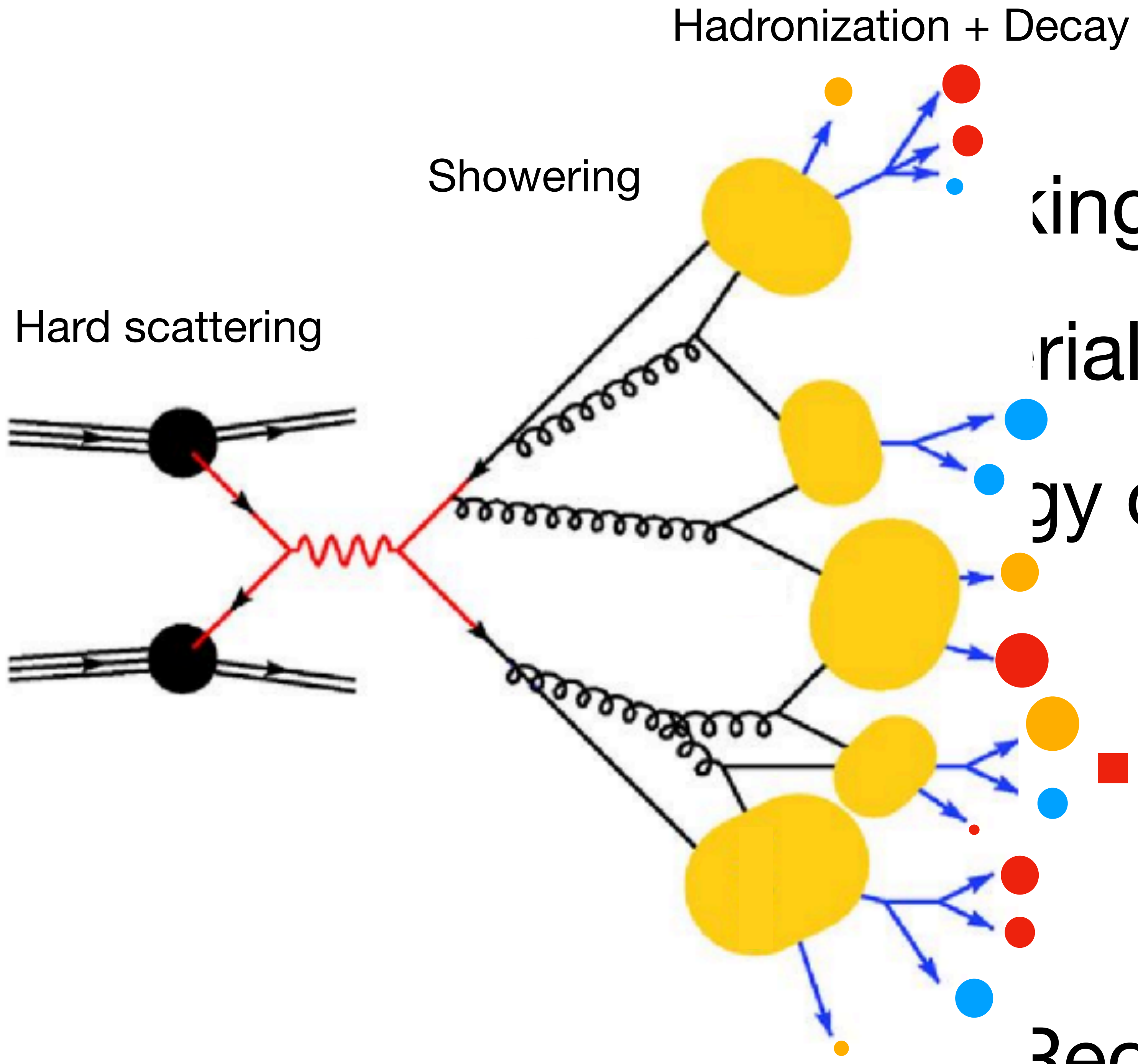


Detector Readout

Reconstruction



Reconstructed objects



king

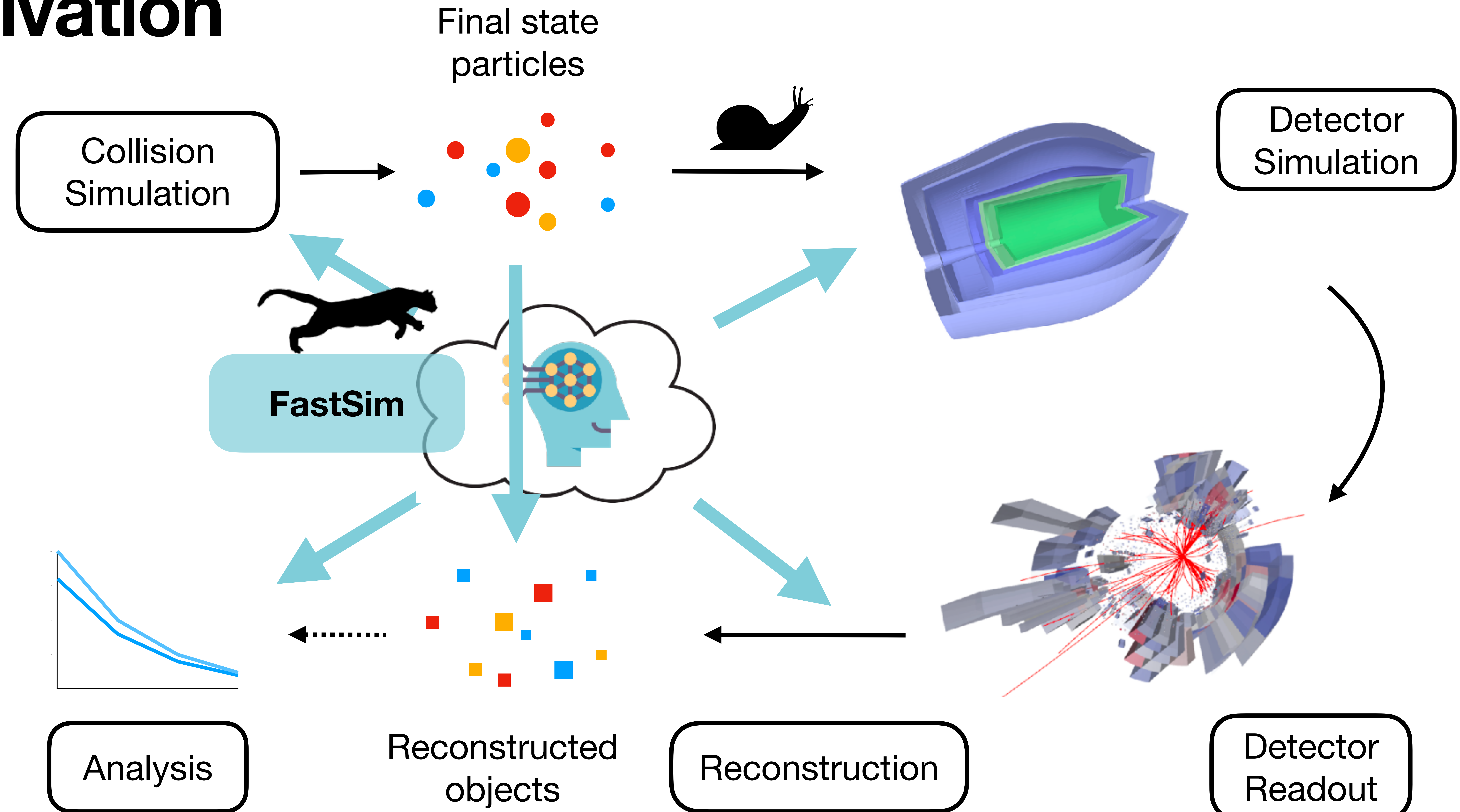
rial interactions

gy deposits in calorimeters

Analysis

N. Soybelman

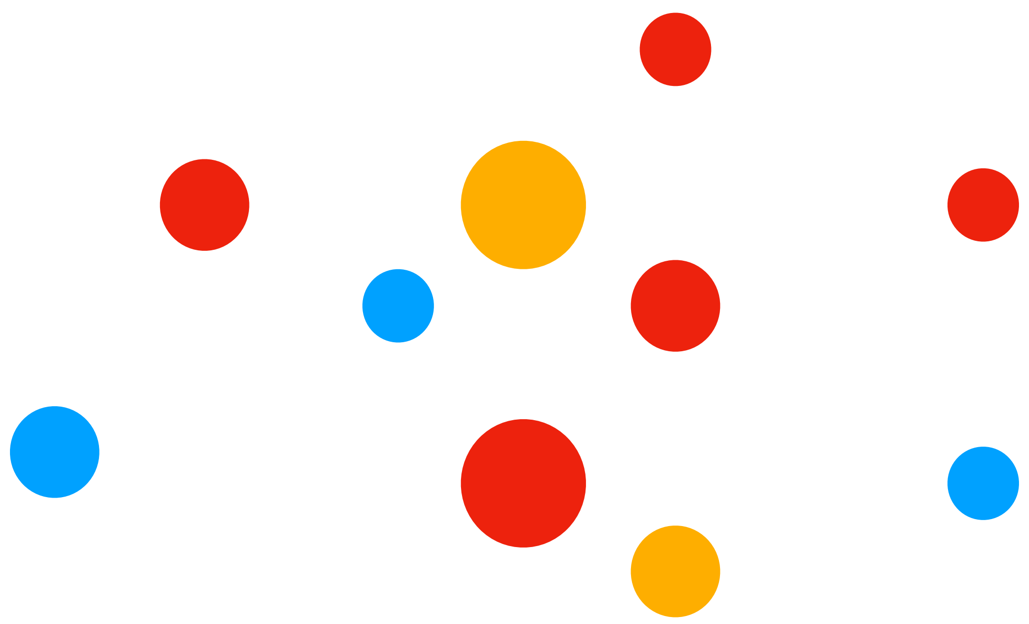
Motivation



N. Soybelman

Problem to solve

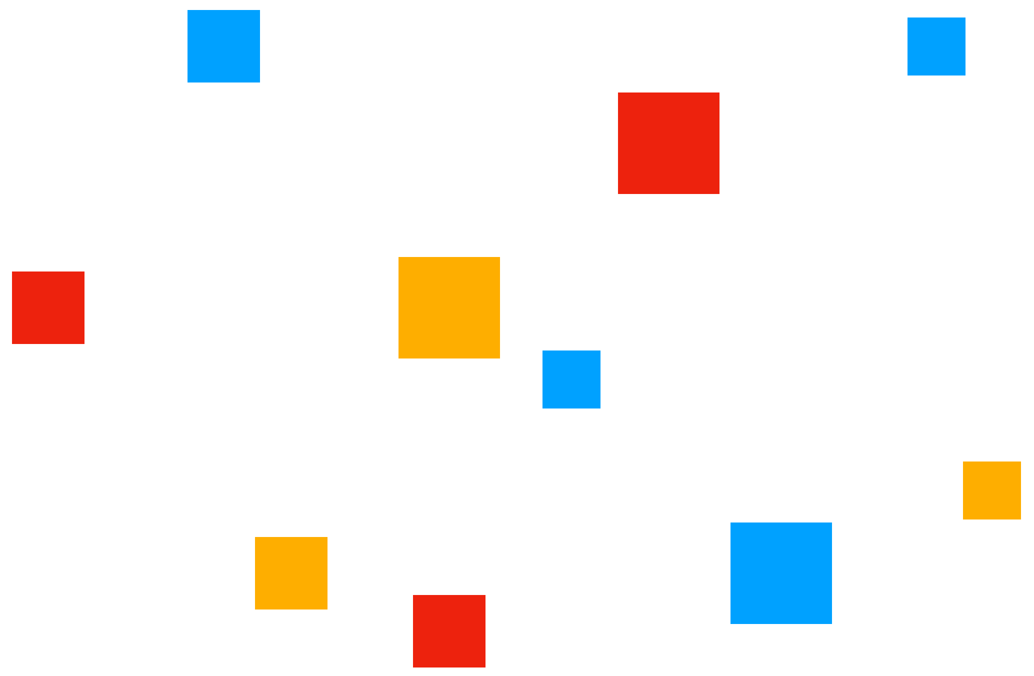
Final state particles



?

Set to Set

Reconstructed objects



Aim: $R \sim q_\theta(R \mid T)$

2 stage Network:

$$R \sim q_{\theta_2}(R \mid N_R, T)\ q_{\theta_1}(N_R \mid T)$$

Goals

$p(R) = \int \mathrm{d}T\ p(R|T)p(T))$

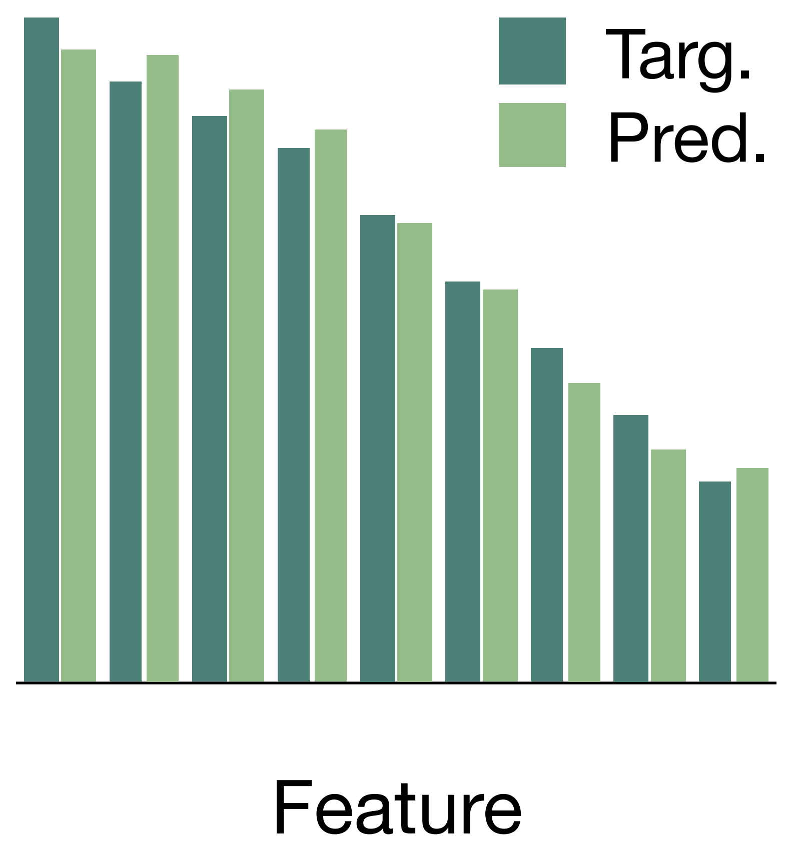
$(d_0, z_0, q/p_T, \theta, \phi)$

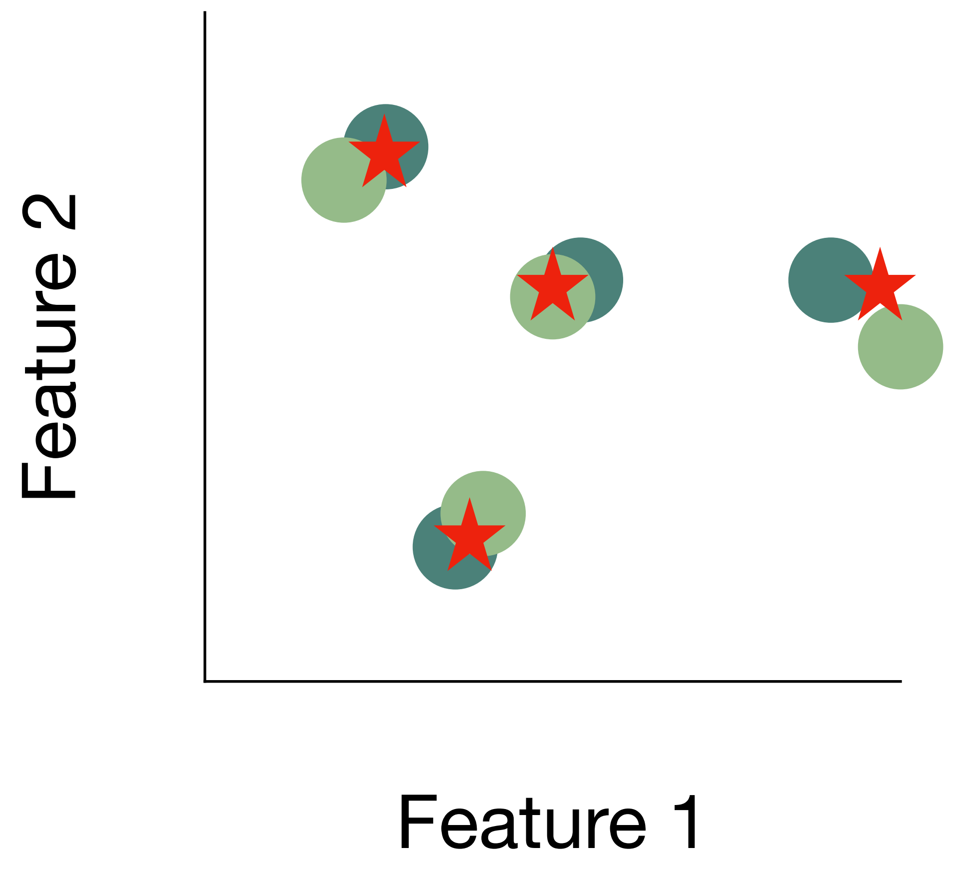
$P(f_R | f_T)$

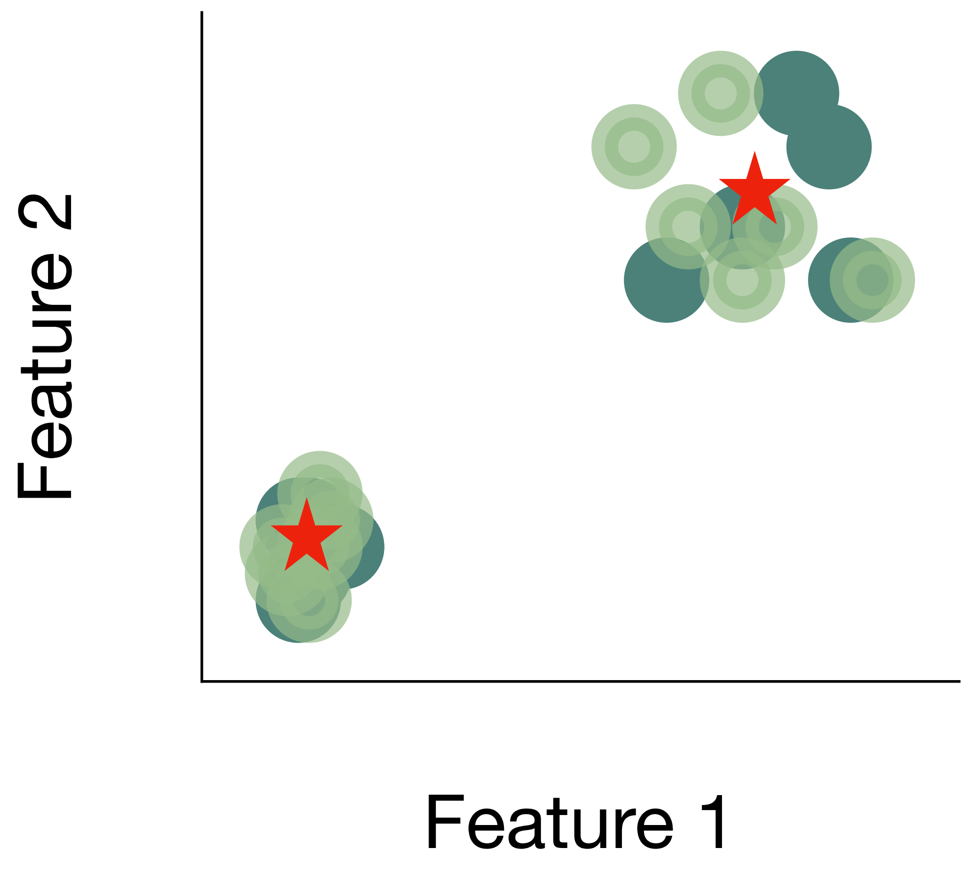
Marginal distributions

Reconstruct constituents

Resolution



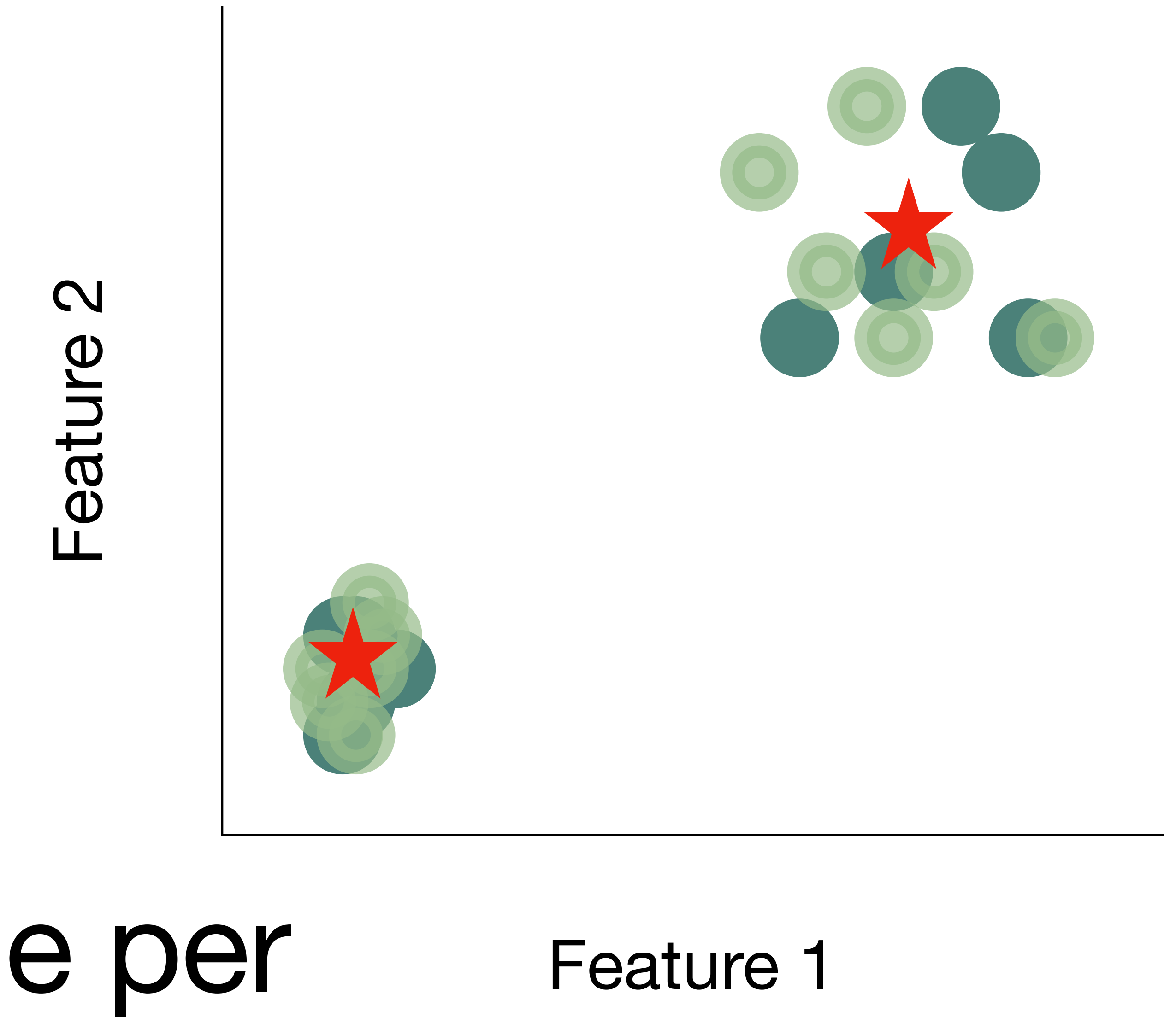




N. Soybelman

Goals: RESOLUTION

How to obtain the correct resolution?



Resolution depends on features

⟶ difficult to learn smearing from one reconstructed sample per truth event

⟶ need in principle very large dataset

Solution

Introduce ***replicas:***

Generate many reconstructions per truth event,

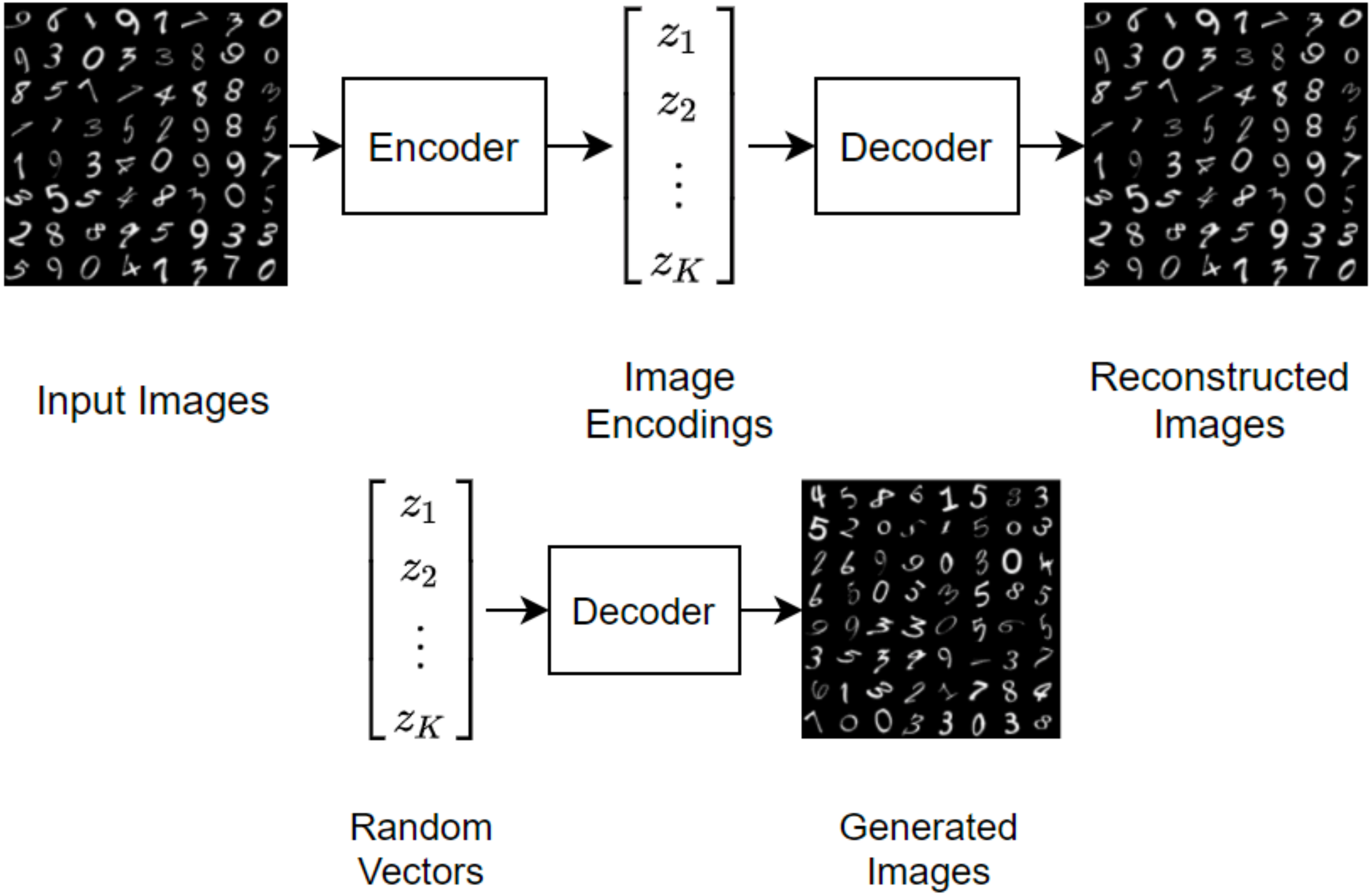
i.e. replicas for the SAME truth event

N. Soybelman

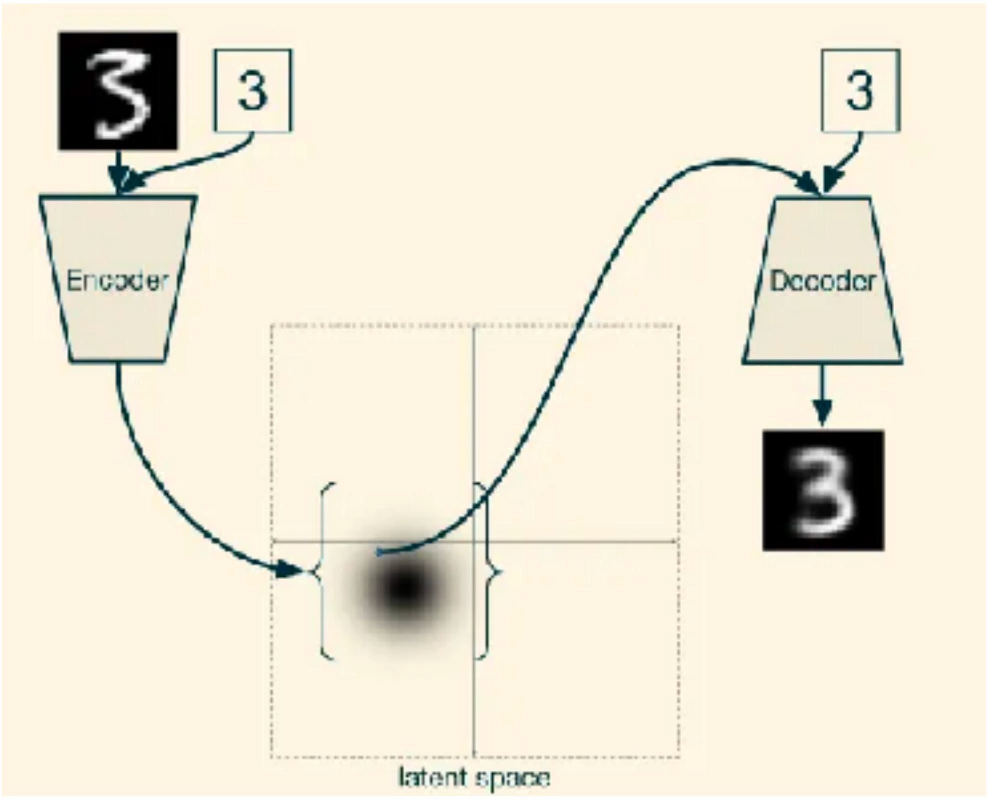
Dataset

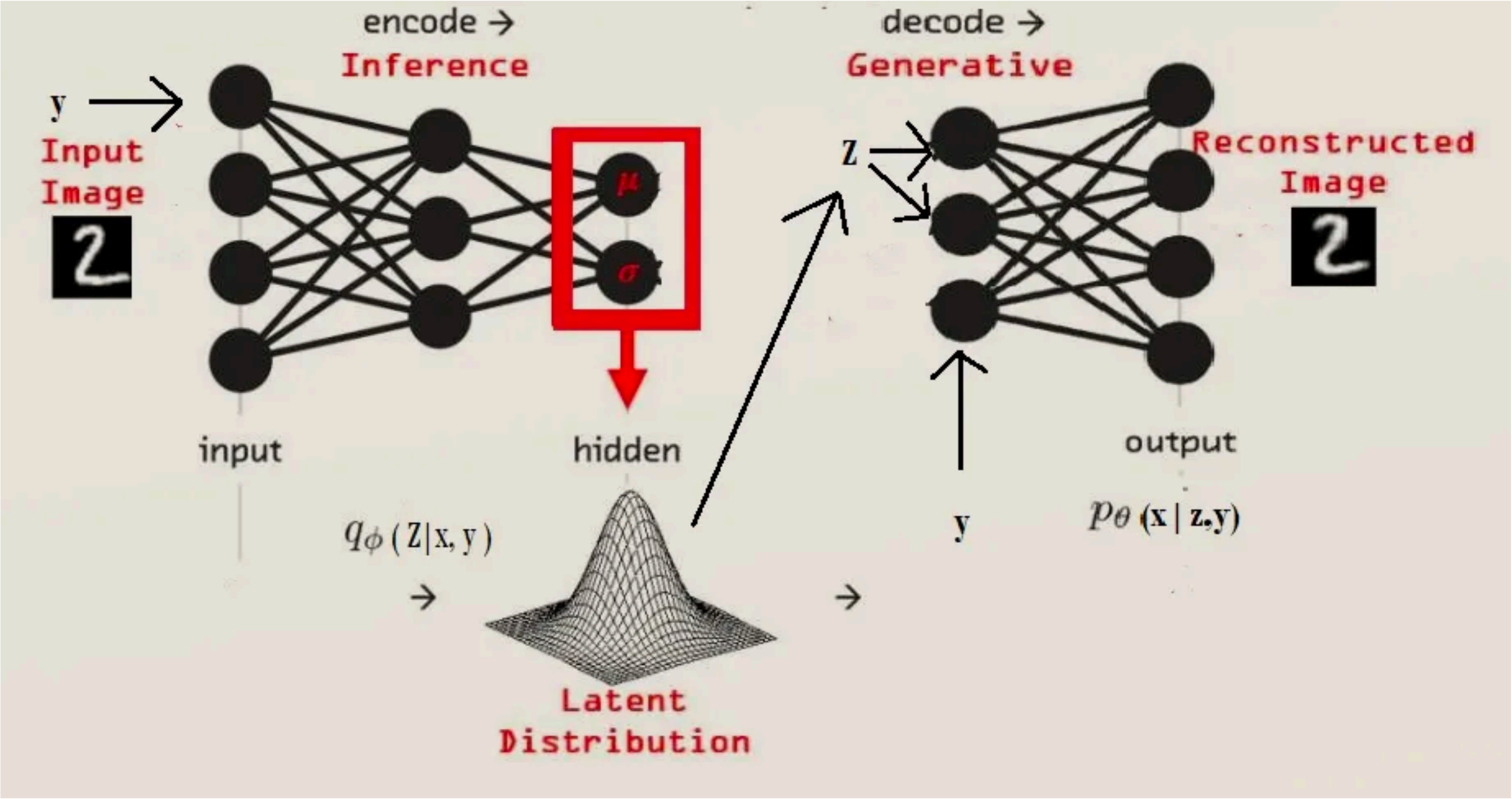
- SET of CHARGED particles within a single jet
- Detector Simulation GEANT based COCOA (tomorrow)
- 1-12 charged particles/jet
- Toy example: Smeared tracks as targets
- Reconstruction efficiency, no fakes $\longrightarrow n_{reco} \leq n_{truth}$
- 100 replicas per event (train on 25 for speed)

Variational Auto Encoder



cVAE


3
Encoder
3
Decoder
latent space


encode →
Inference
decode →
Generative
y
Input
Image
μ
σ
Z
Reconstructed
Image
input
hidden
output
y
q_ϕ (Z|x, y)
p_θ (x | z,y)
Latent
Distribution

Variational Auto Encoder as Baseline

VAE in a NUTSHELL

- Start by picking a prior to Z, p(Z)~N(0,1)
  The details are in the conditional P(X|Z)

- The decoder learns the distribution of x|z, it learns two functions

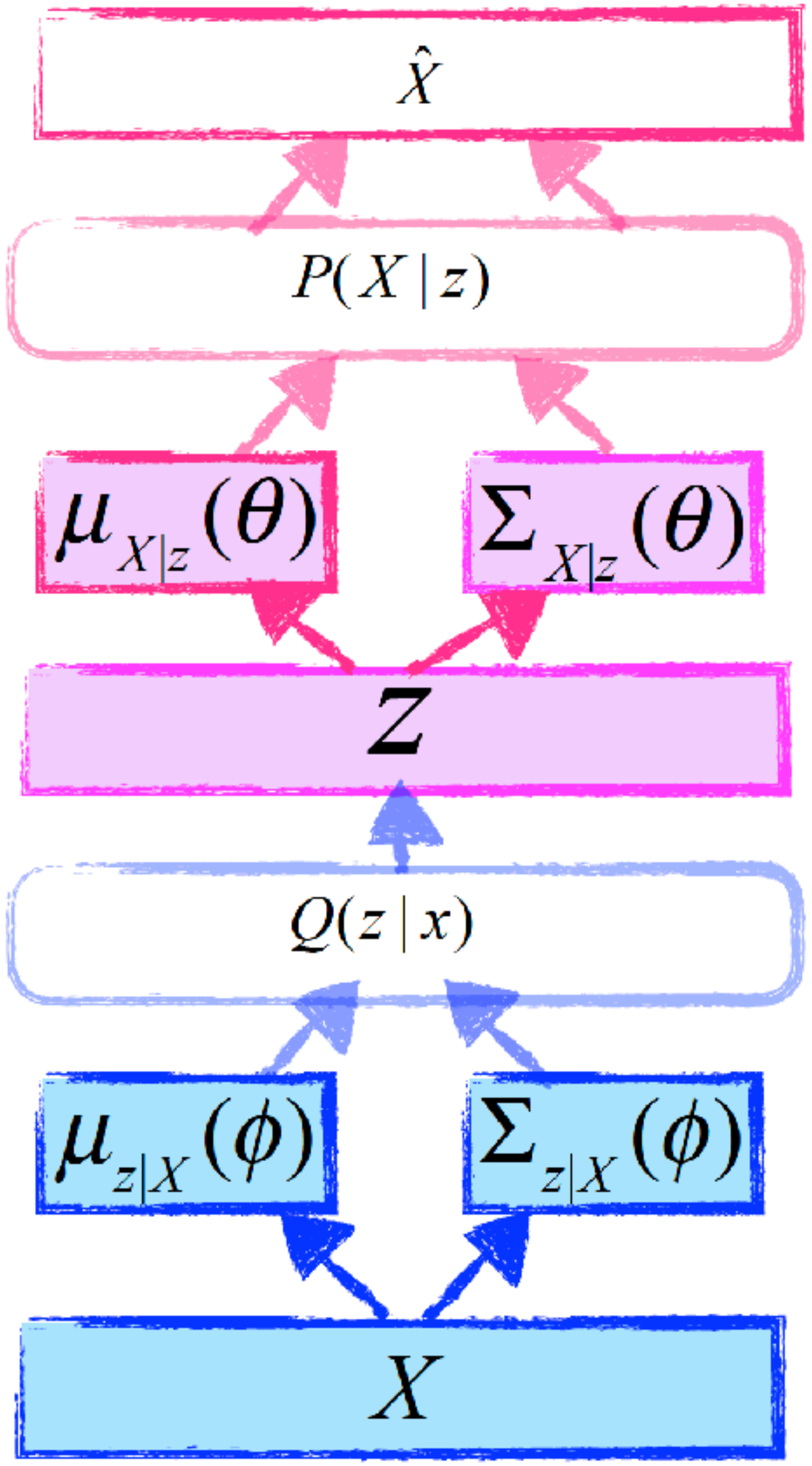
  $\mu_{X|z}(\theta) \quad \Sigma_{X|z}(\theta)$

Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

- To ensure that the latent space contains z that lead to a DATA-like X the encoder learns an **approximate** distribution of **P(Z|X); Q(Z|X)** by learning two functions

  $\mu_{z|X}(\phi) \quad \Sigma_{z|X}(\phi)$

  from which Z|x is sampled



VAE

Maximize Likelihood of original input X

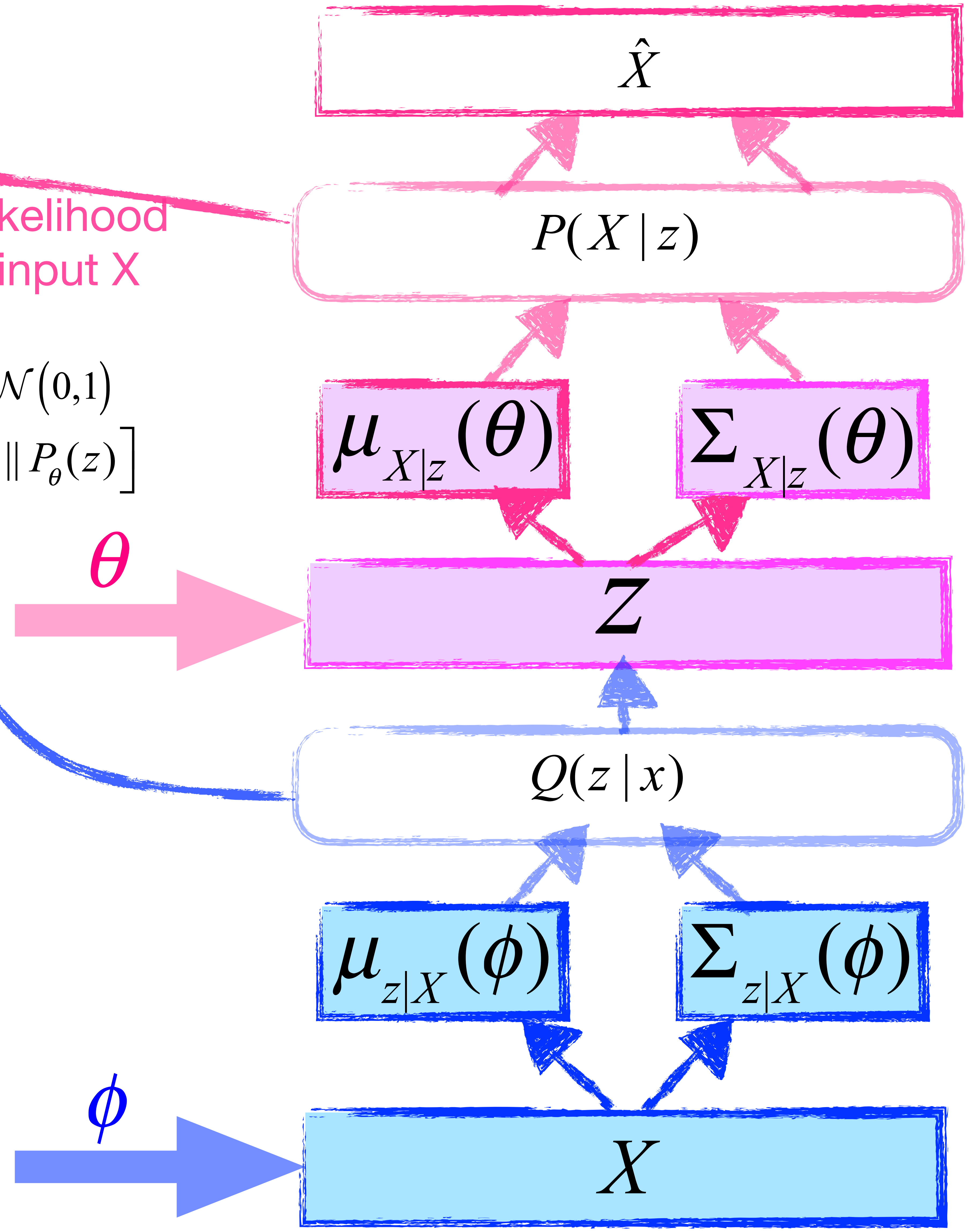
Putting it all together: maximizing the likelihood lower bound

$$P(z) = \mathcal{N}(0,1)$$

$$\log P_\theta(X) \le E_{z \sim Q_\phi(z|X)}[\log(P_\theta(X \mid z)] - D_{KL}\left[Q_\phi(z \mid X) \parallel P_\theta(z)\right]$$

Make posterior distribution of z close to prior

For every minibatch of input data: compute this forward pass, and then backprop!



We update our model by continuously update the decoder and encoder parameters, Φ and ϑ

VAE

Maximize Likelihood
of original input X

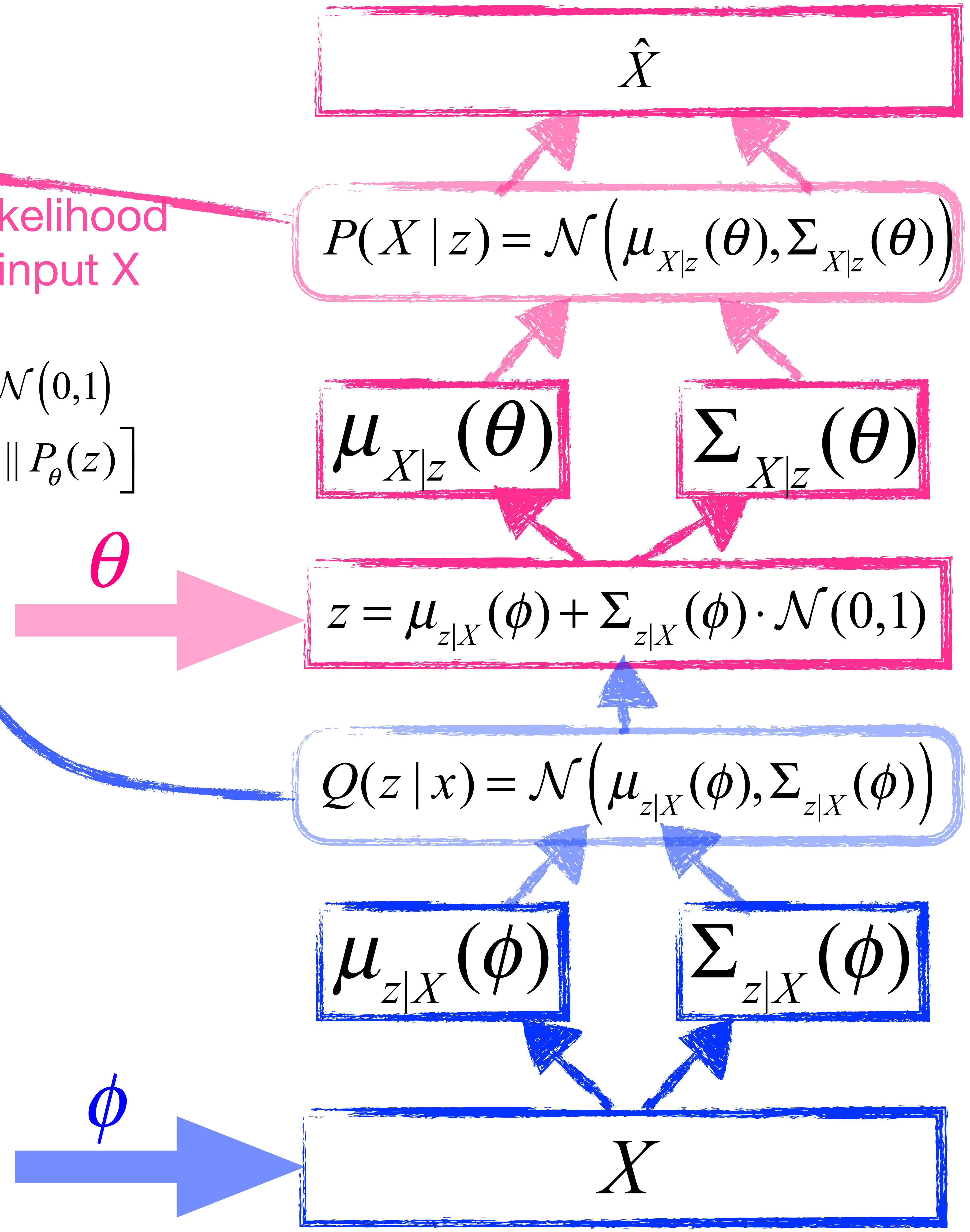
Putting it all together: maximizing the
likelihood lower bound

$P(z)=\mathcal{N}(0,1)$

$$\log P_\theta(X) \le E_{z\sim Q_\phi(z|X)}[\log(P_\theta(X|z)] - D_{KL}\left[Q_\phi(z|X)\,\|\,P_\theta(z)\right]$$

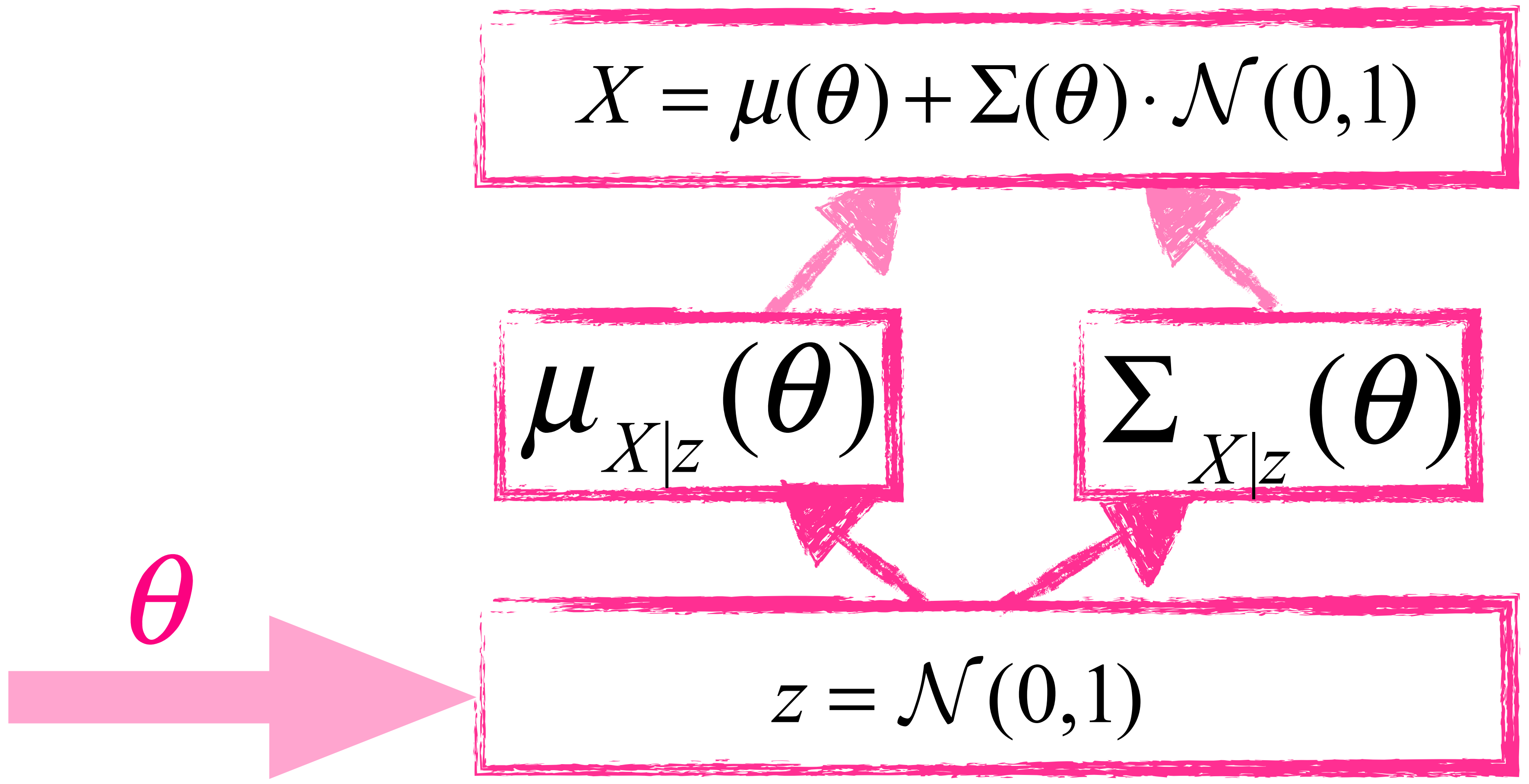
Make posterior
distribution of z
close to prior

For every minibatch of input
data: compute this forward
pass, and then backprop!

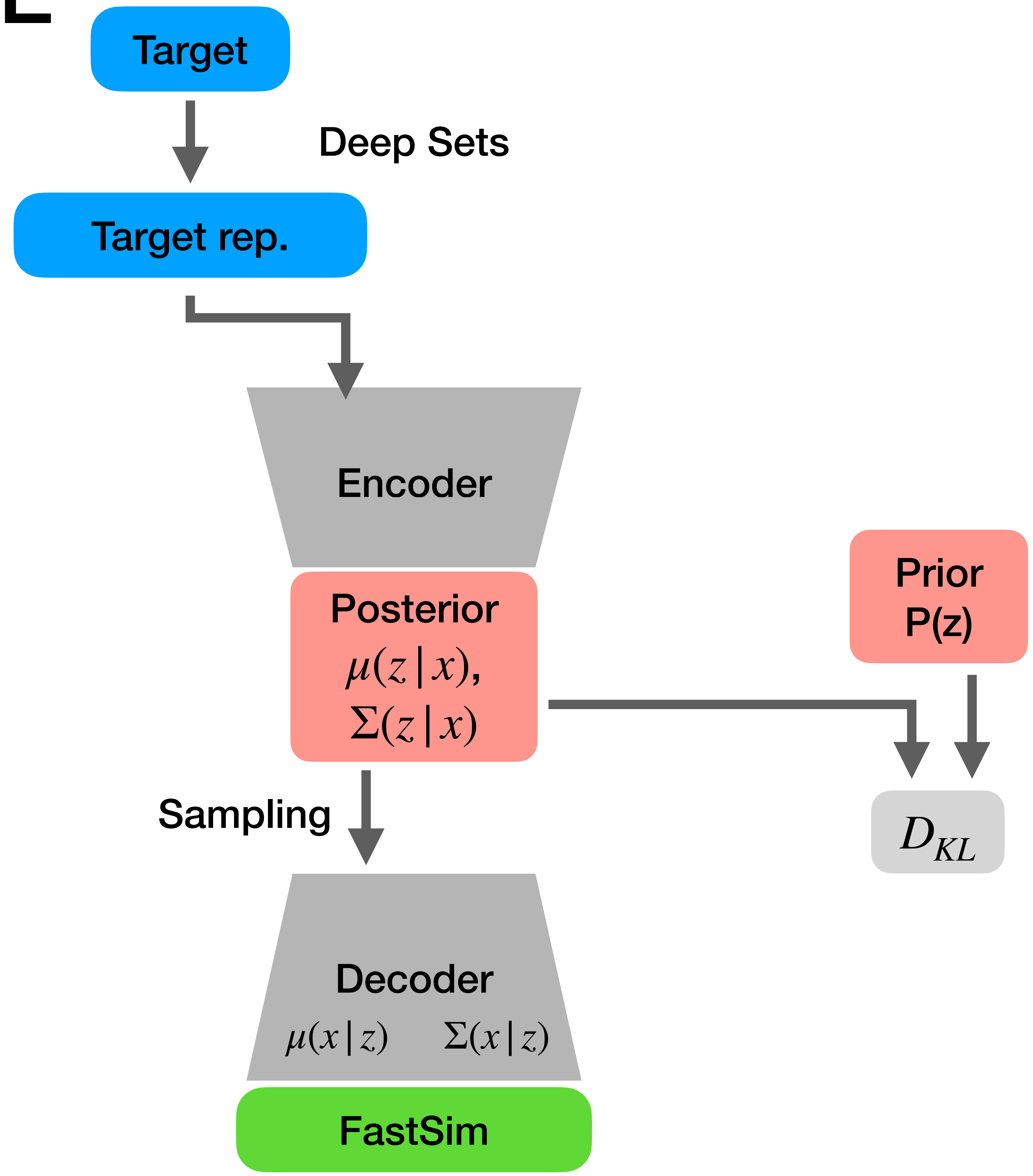


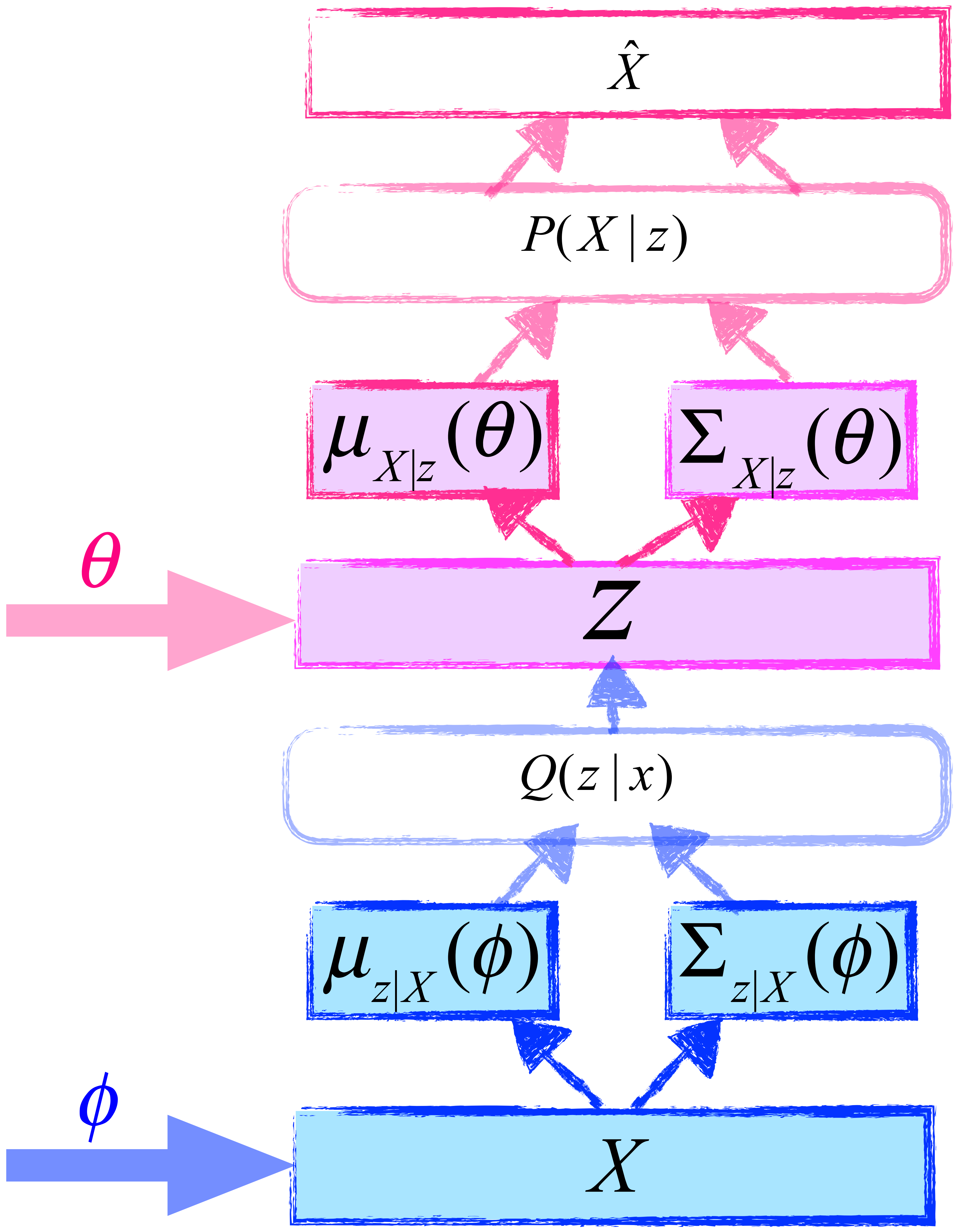
We update our model by continuously update the decoder and encoder parameters, Φ and ϑ

VAE

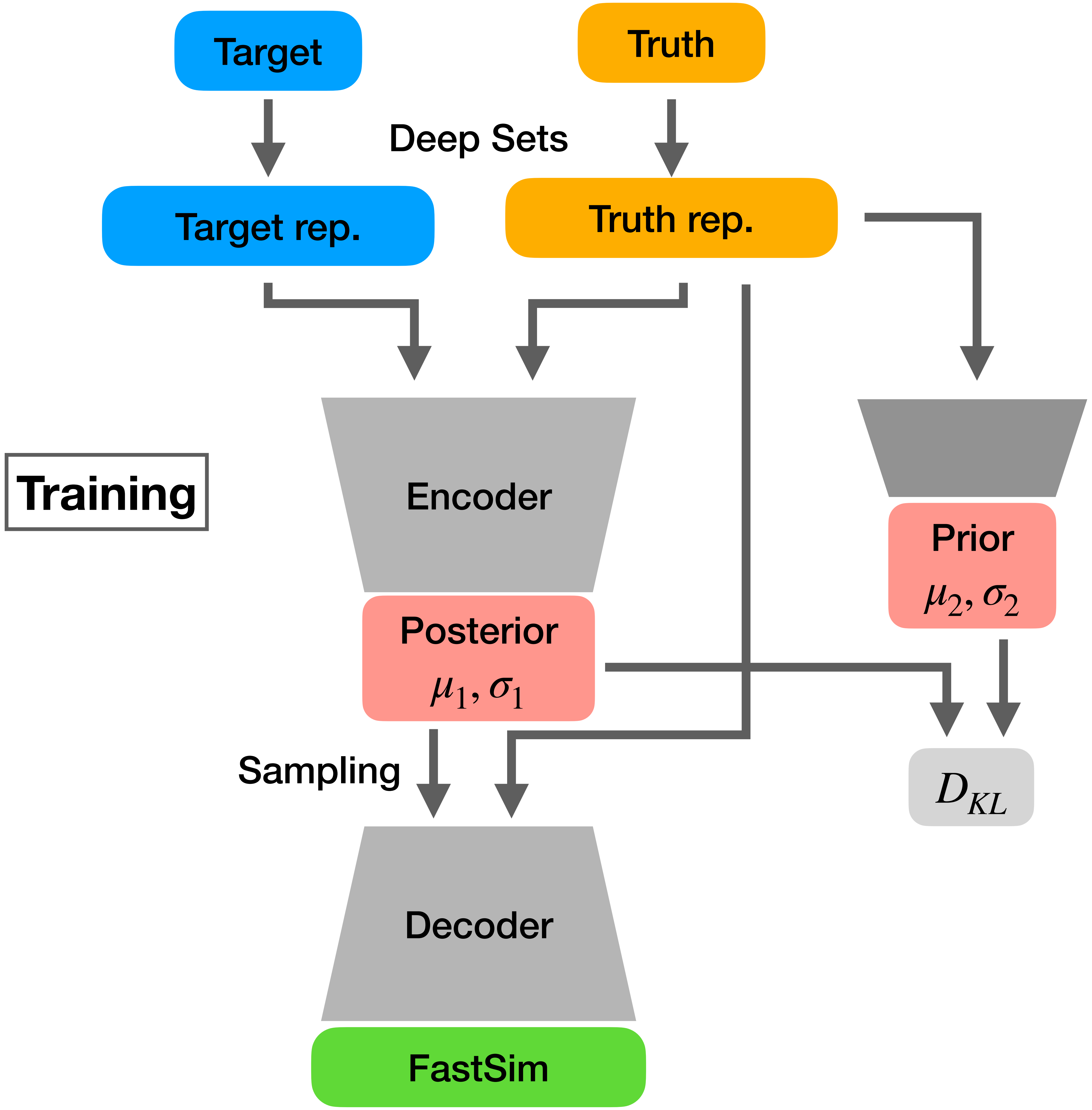

$X = \mu(\theta) + \Sigma(\theta) \cdot \mathcal{N}(0,1)$
$\mu_{X|z}(\theta)$
$\Sigma_{X|z}(\theta)$
θ
$z = \mathcal{N}(0,1)$

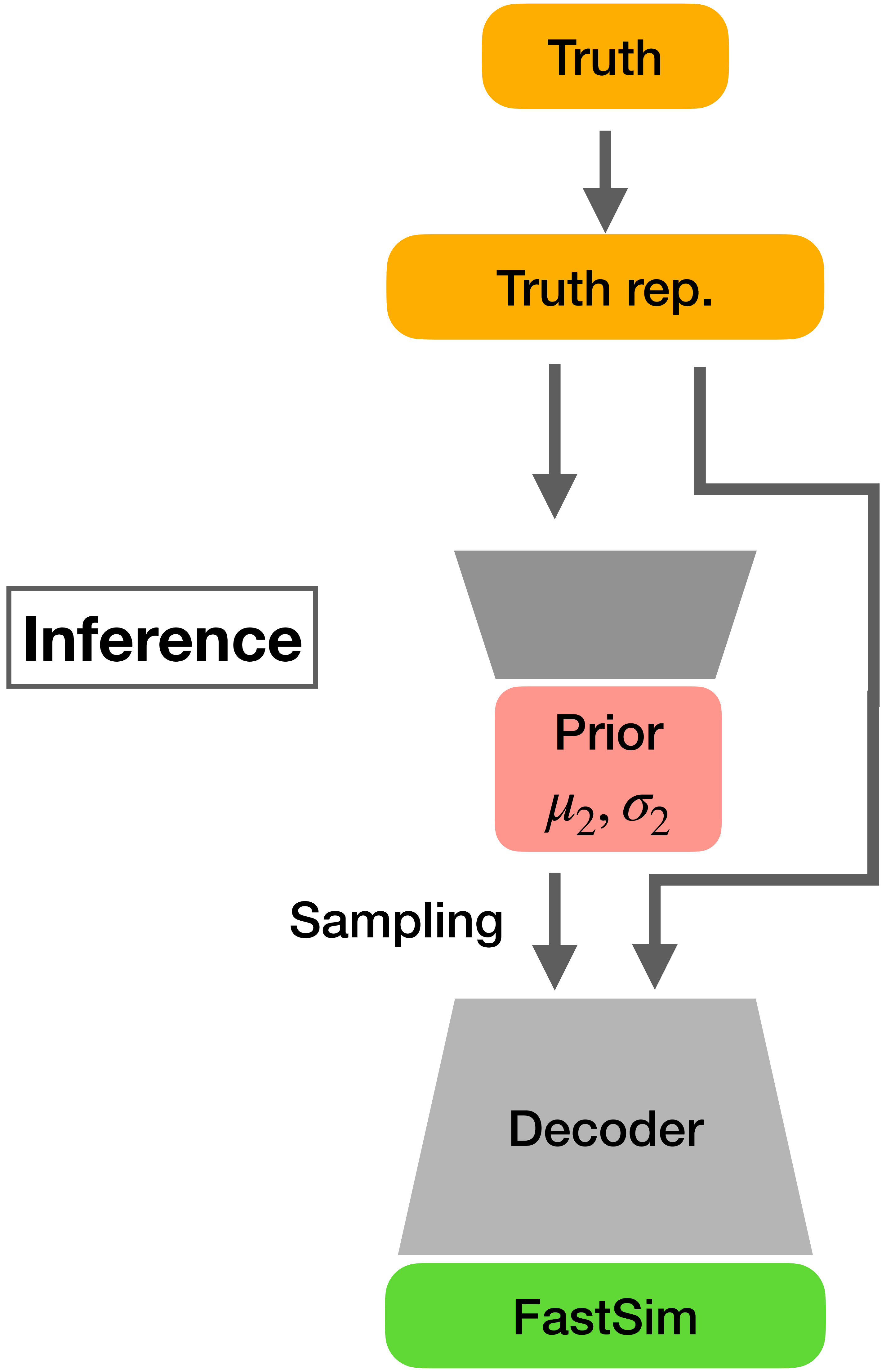
VAE


Target
Deep Sets
Target rep.
Encoder
Posterior
$\mu(z|x)$,
$\Sigma(z|x)$
Prior
P(z)
Sampling
D_{KL}
Decoder
$\mu(x|z)$ $\Sigma(x|z)$
FastSim


$\hat{X}$
$P(X|z)$
$\mu_{X|z}(\theta)$
$\Sigma_{X|z}(\theta)$
θ
Z
$Q(z|x)$
$\mu_{z|X}(\phi)$
$\Sigma_{z|X}(\phi)$
ϕ
X

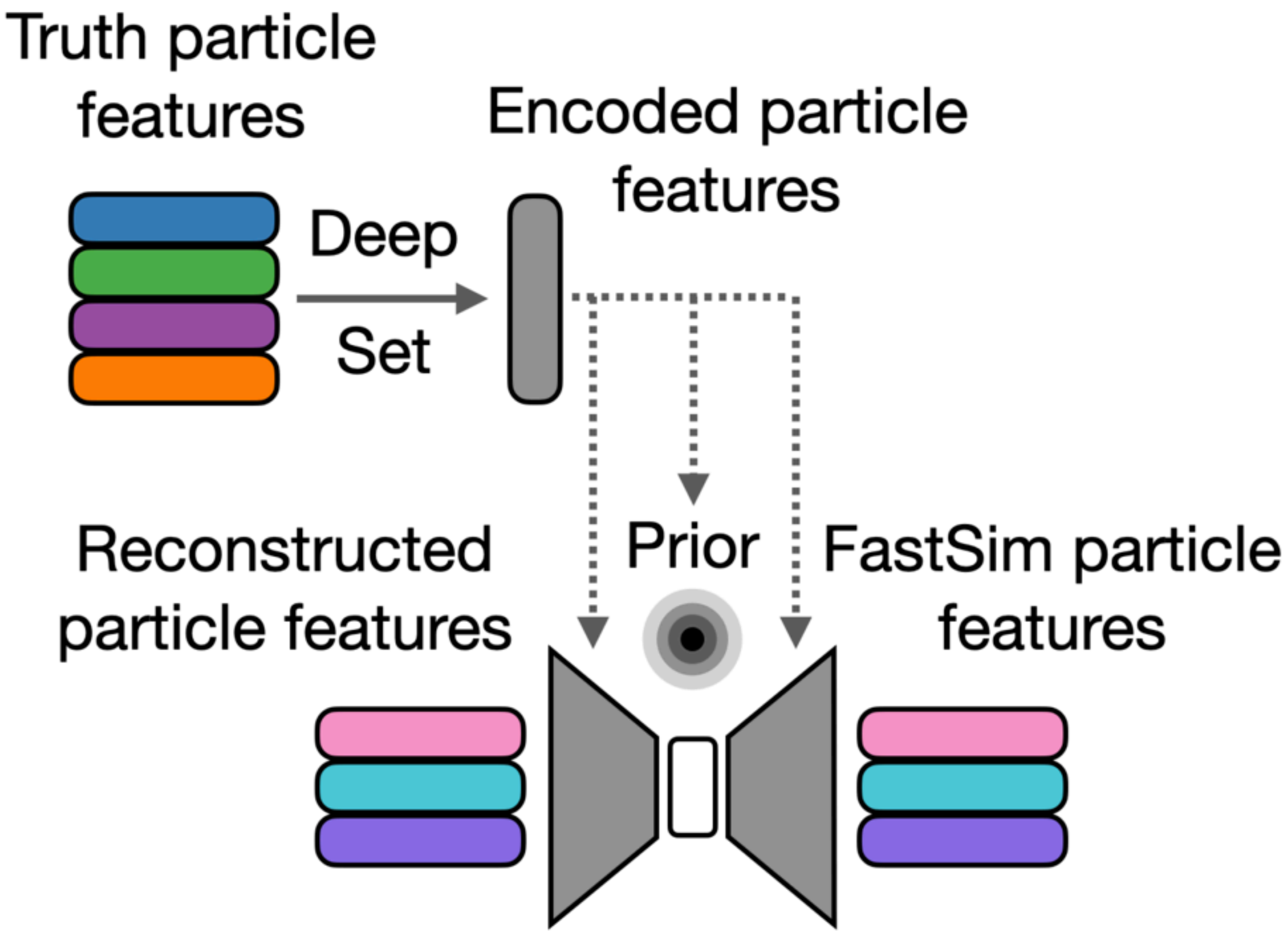
cVAE Architecture



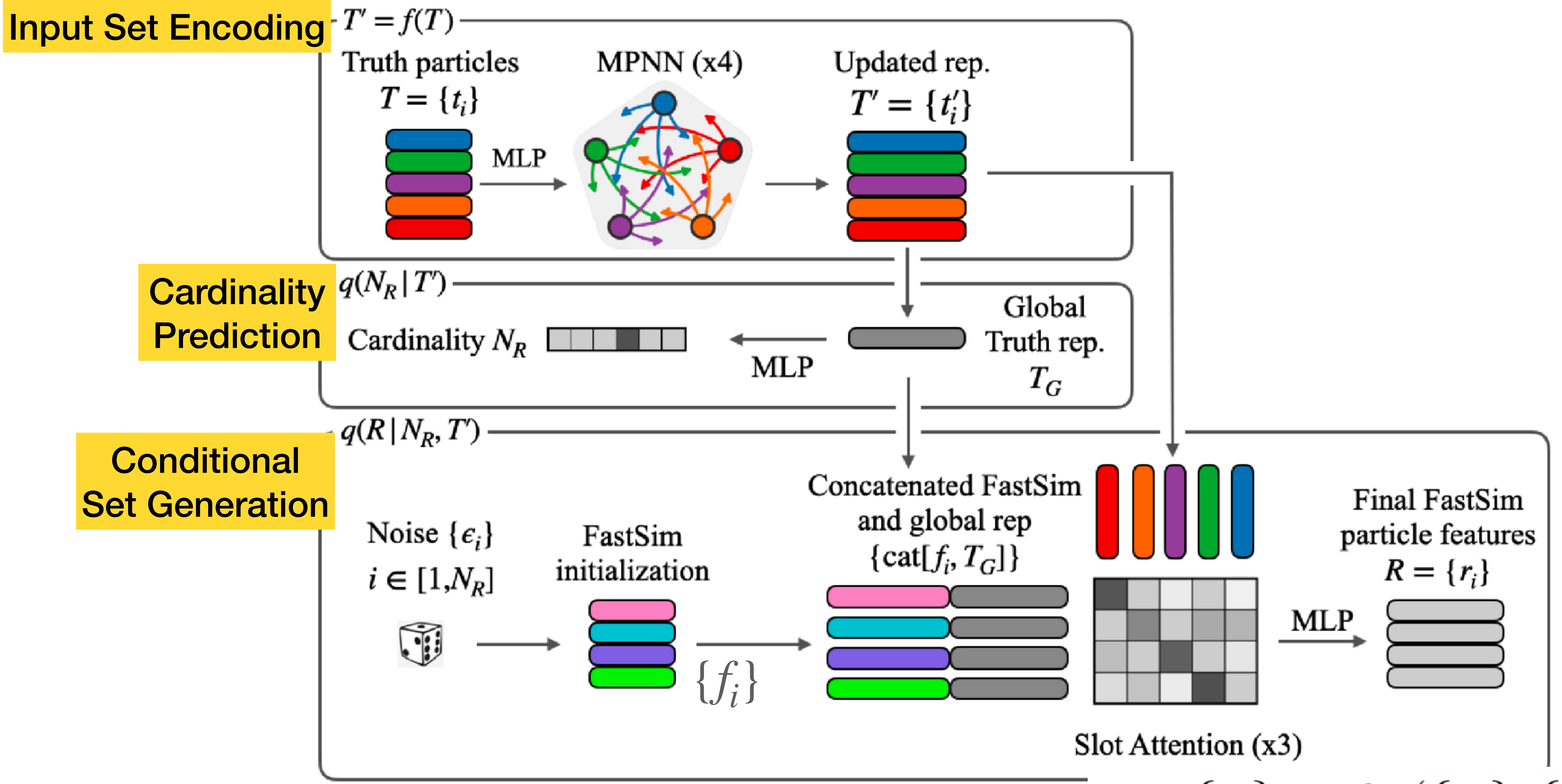


N. Soybelman

cVAE Architecture



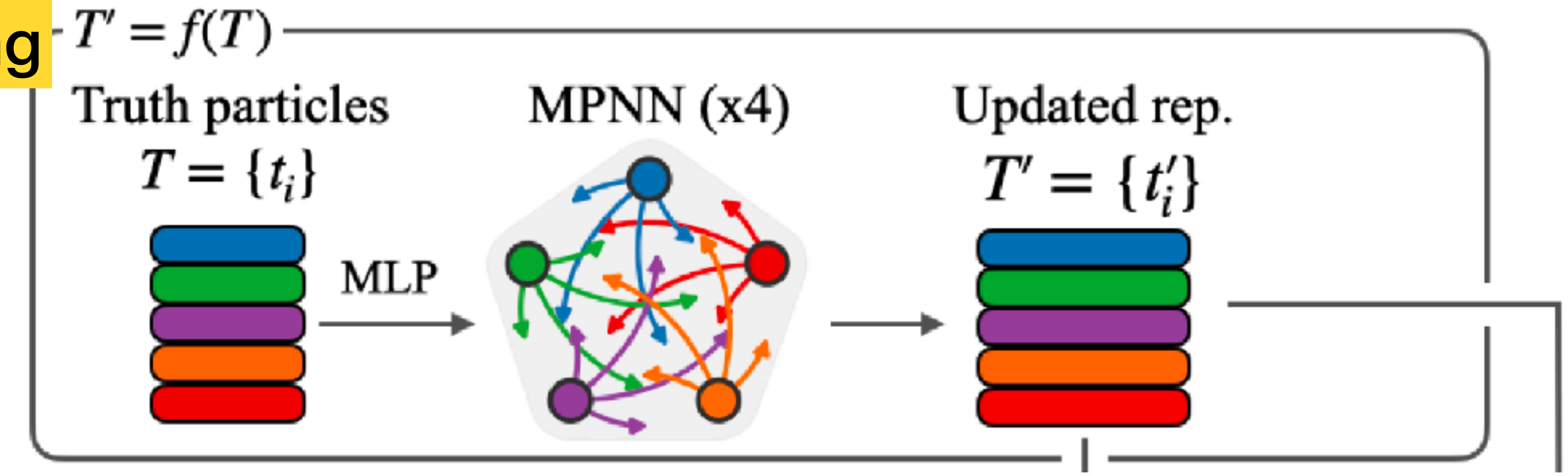
GNN Architecture



$$R = \{r_i\} = f_{\theta_2}(\{\varepsilon_i\}, \{t_i'\})$$

GNN Architecture

Input Set Encoding

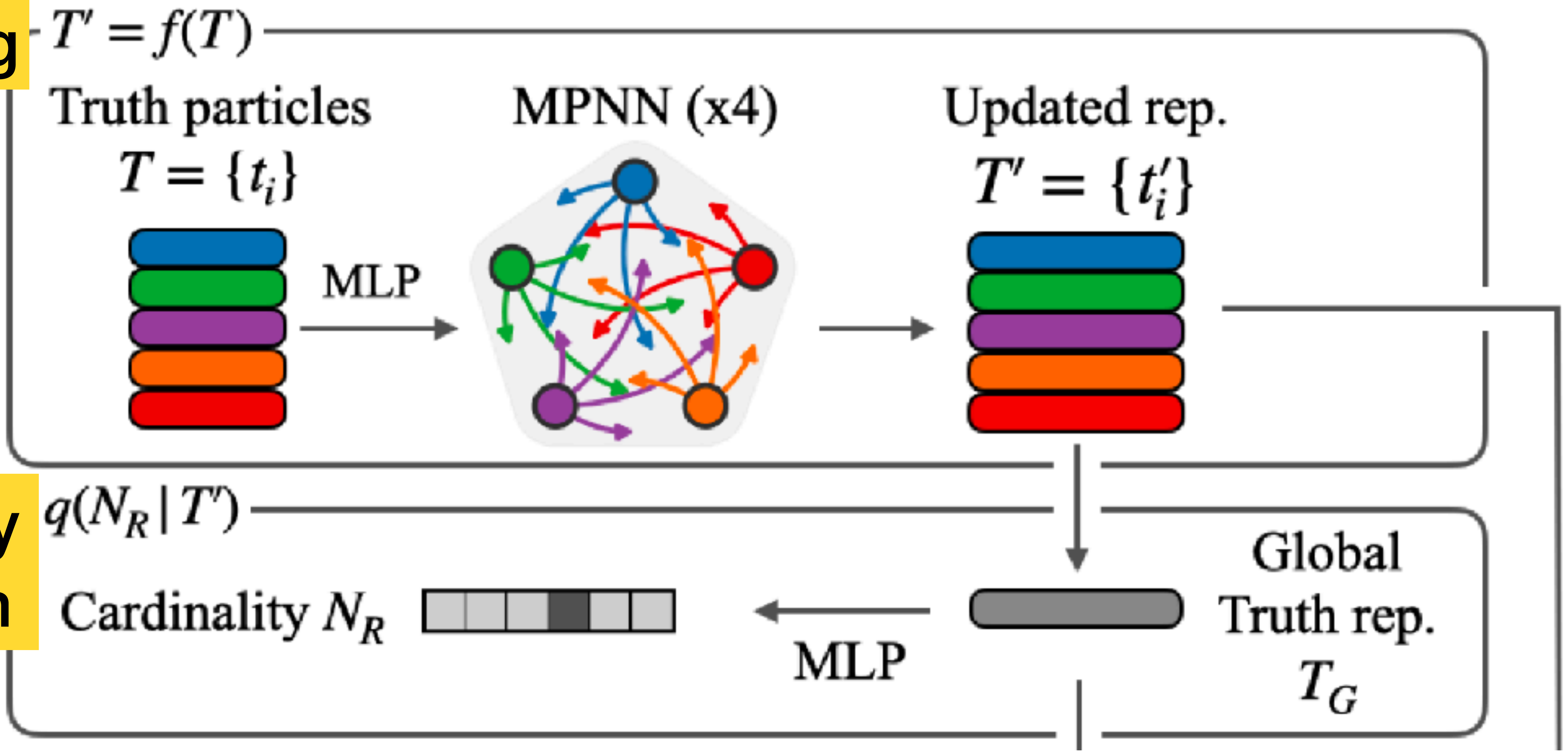


$$\{f_i\} = \{p_{T,i}, \eta_i, \phi_i\}$$

GNN Architecture

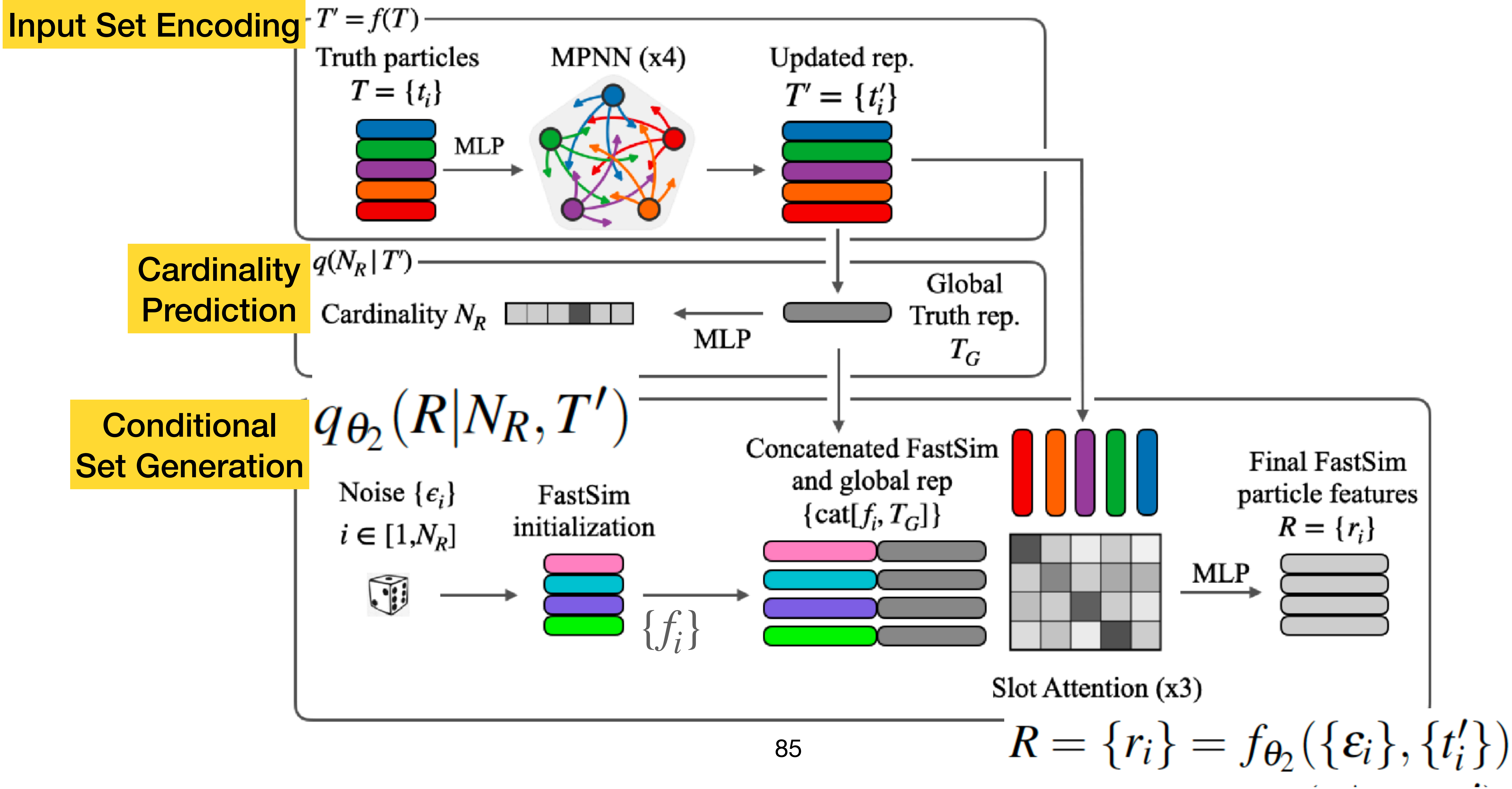
Input Set Encoding

Cardinality Prediction



$$q_{\theta_1}(N_R|T)$$

GNN Architecture

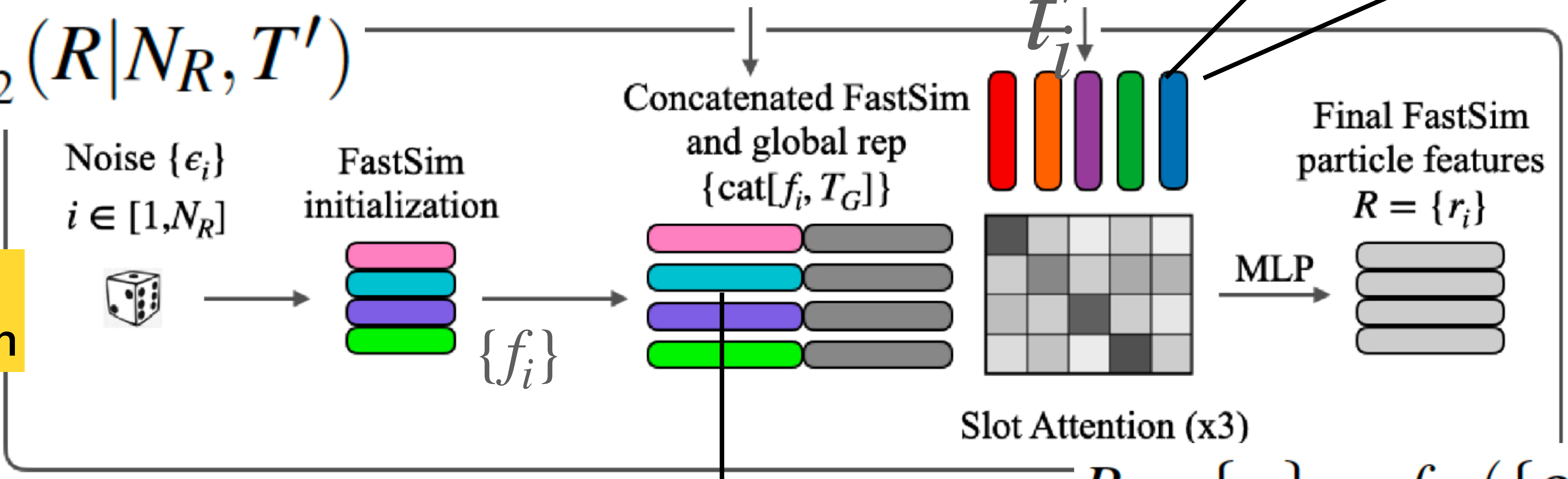


GNN Architecture:Attention on the Attention

$k = F_k(t_i', t_i), \quad v = F_v(t_i', t_i)$

$q_{\theta_2}(R|N_R, T')$

Conditional Set Generation



$R = \{r_i\} = f_{\theta_2}(\{\varepsilon_i\}, \{t_i'\})$

$q = F_q(f_i, T_G).$

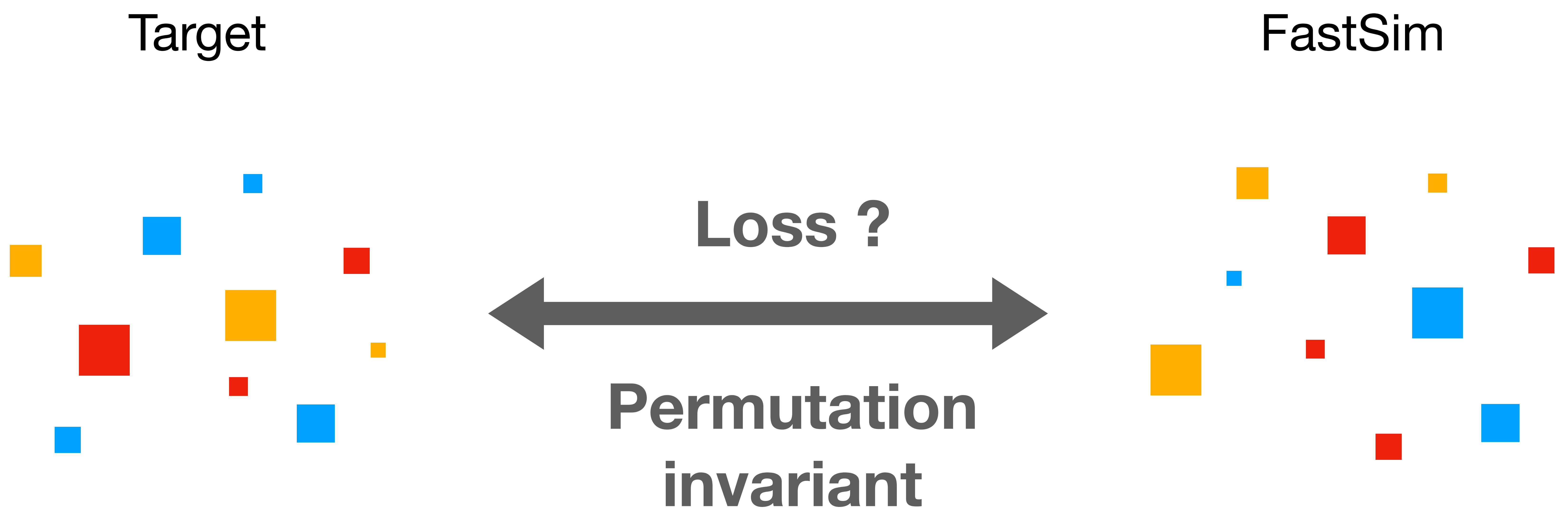
$T_G = agg\{t_i'\}$

$$\boxed{A = \mathtt{Softmax}\left(\frac{1}{\sqrt{D}} k \cdot q^T\right)}$$

MLP (GRU)

$$f_i \longrightarrow f_i \; + \; \mathtt{MLP}(\mathtt{LayerNorm}(\mathtt{GRU}(v * A, f_i)))$$

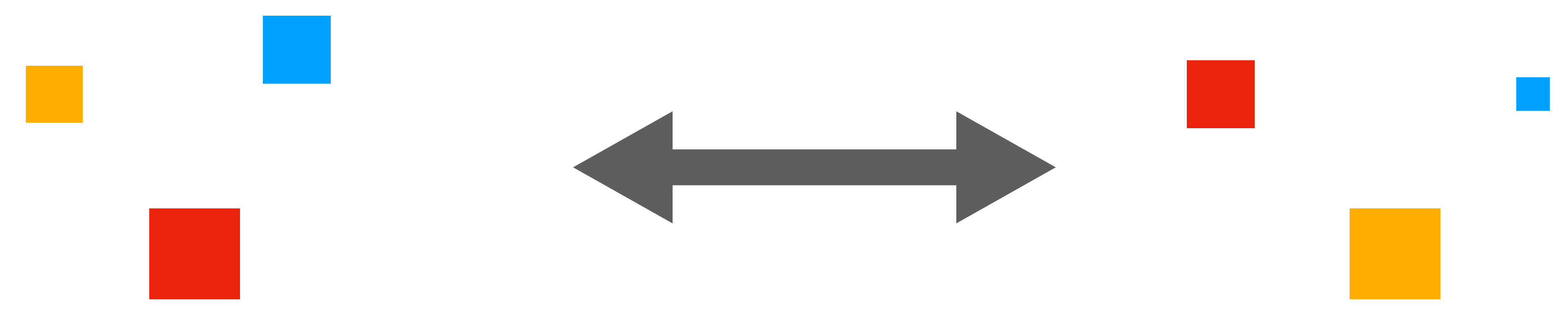
Set to Set problem



N. Soybelman

Hungarian matching

Target FastSim



$$\min \sum_{i}^{n} \sum_{j}^{m} C_{ij} X_{ij} \quad \text{where} \quad X_{ij} = \begin{cases} 1 & \text{if row } i \text{ is assigned to column } j \\ 0 & \text{otherwise} \end{cases}$$

$$\boxed{C_{ij} = (p_{Ti} - p_{Tj})^2 + (\eta_i - \eta_j)^2 + (1 - \cos(\phi_i - \phi_j))}$$

Target

FastSim

	0.5	0.8	0.1
	0.3	0.7	0.6
	0.3	0.4	1.5

Invariant under exchange of columns/rows

N. Soybelman

“Double Hungarian”

Replicas—> Set of Sets

Construct a a samplebased

similarity measure between the two distributions

$p(R\,|\,T,N)$ $\qquad$ $q_\phi(R\,|\,T,N)$

$$MMD^2(p,q) = \mathbb{E}_{x,x'\sim p}k(x,x') + \mathbb{E}_{y,y'\sim q}k(y,y') - 2\mathbb{E}_{x\sim p,y\sim q}k(x,y)$$

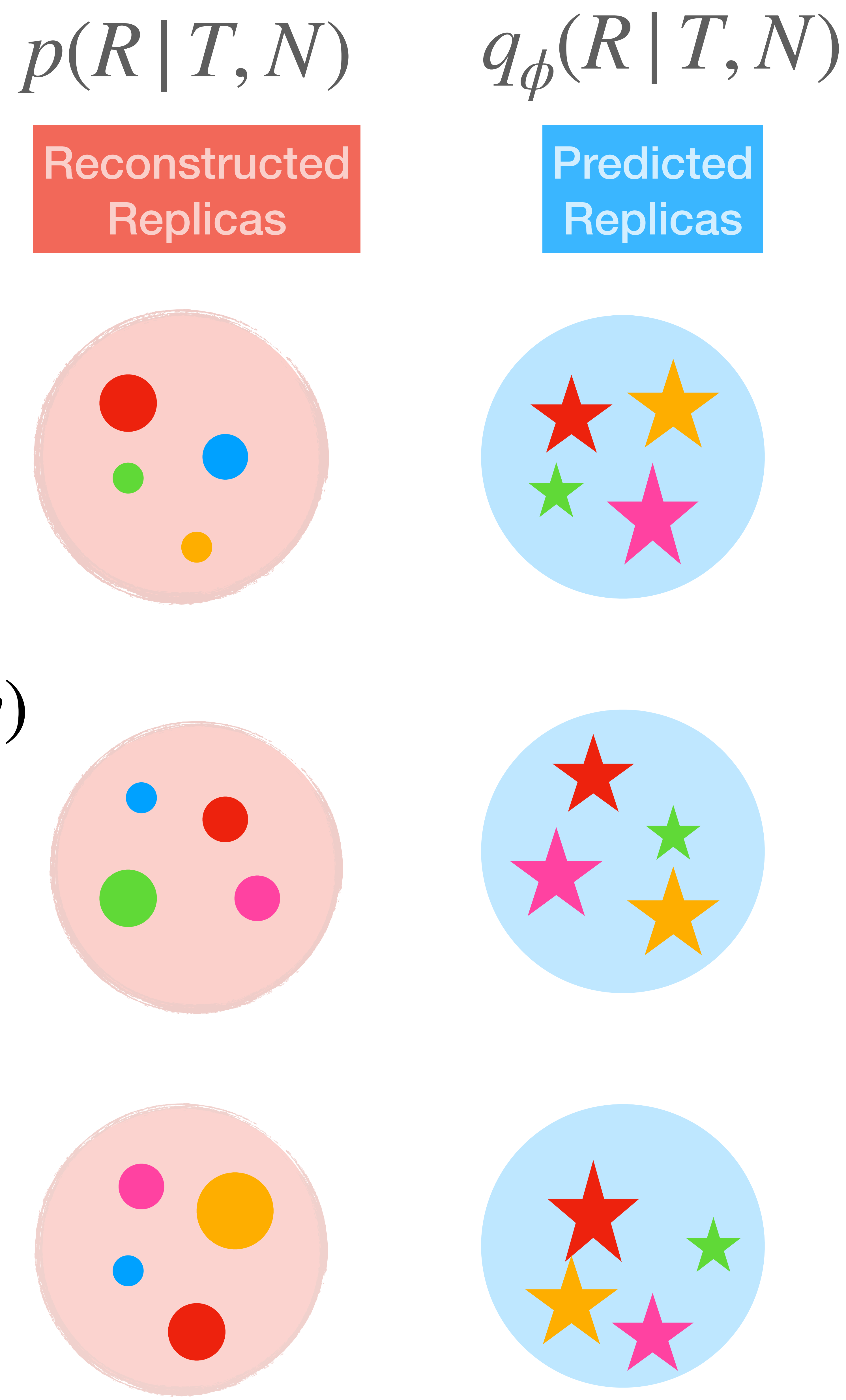
$x_i = (p_{T,i}, \eta_i, \phi_i)$ $\qquad$ $y_i = (p'_{T,i}, \eta'_i, \phi'_i)$

Hungarian $\qquad$ $k(x,y) = ||x-y||^2$

MMD vanishes when p=q

but is time consuming

$$L_{proxy} = min_{x_i,y_j}k(x_i,y_j)$$



“Double Hungarian”

Replicas —> Set of Sets

Construct a a samplebased
similarity measure between the two distributions

$p(R\,|\,T,N)$ $\qquad$ $q_\phi(R\,|\,T,N)$

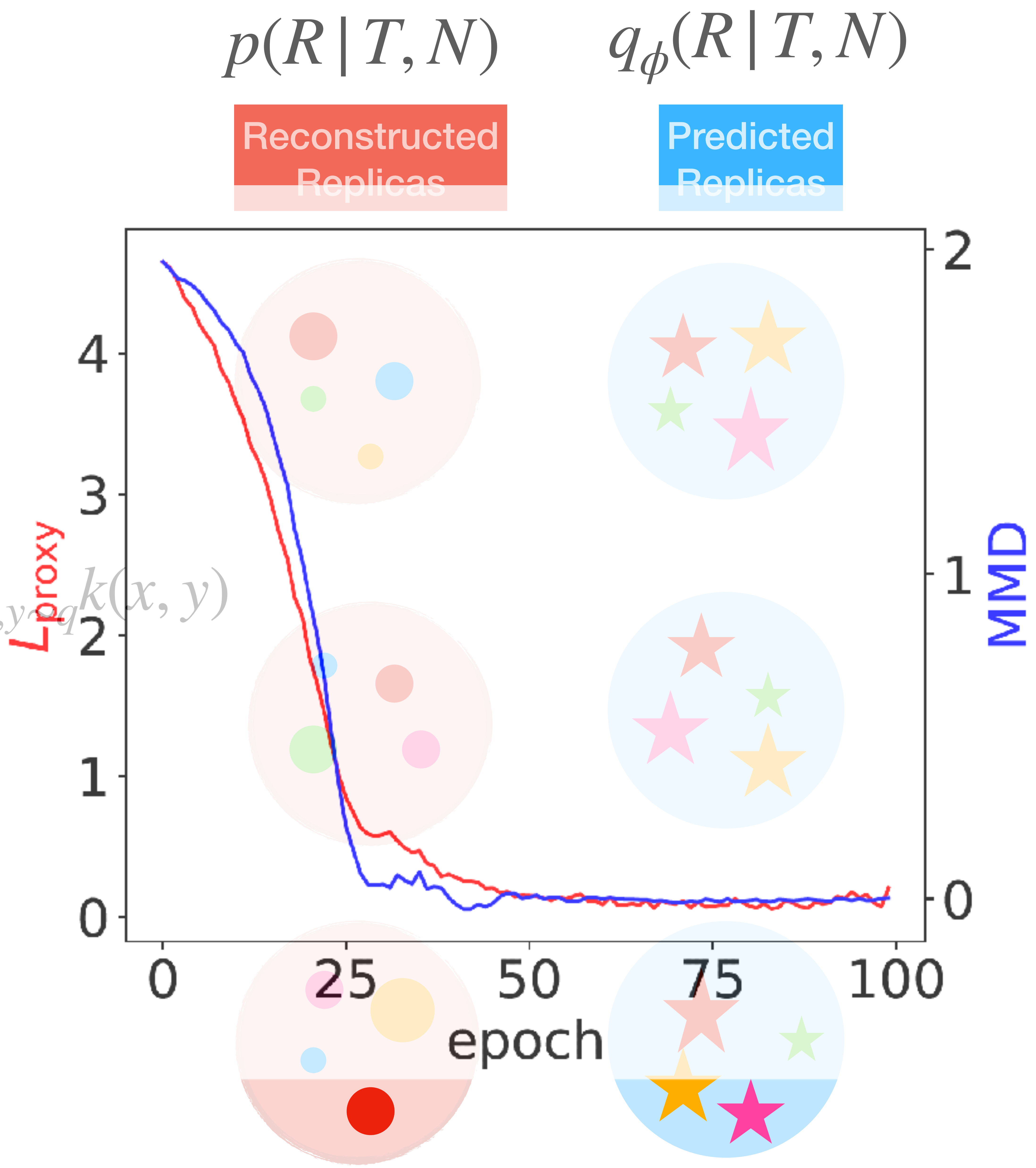
$$MMD^2(p,q) = \mathbb{E}_{x,x'\sim p}k(x,x') + \mathbb{E}_{y,y'\sim q}k(y,y') - 2\mathbb{E}_{x\sim p,y\sim q}k(x,y)$$

$x_i = (p_{T,i}, \eta_i, \phi_i)$ $\qquad$ $y_i = (p'_{T,i}, \eta'_i, \phi'_i)$

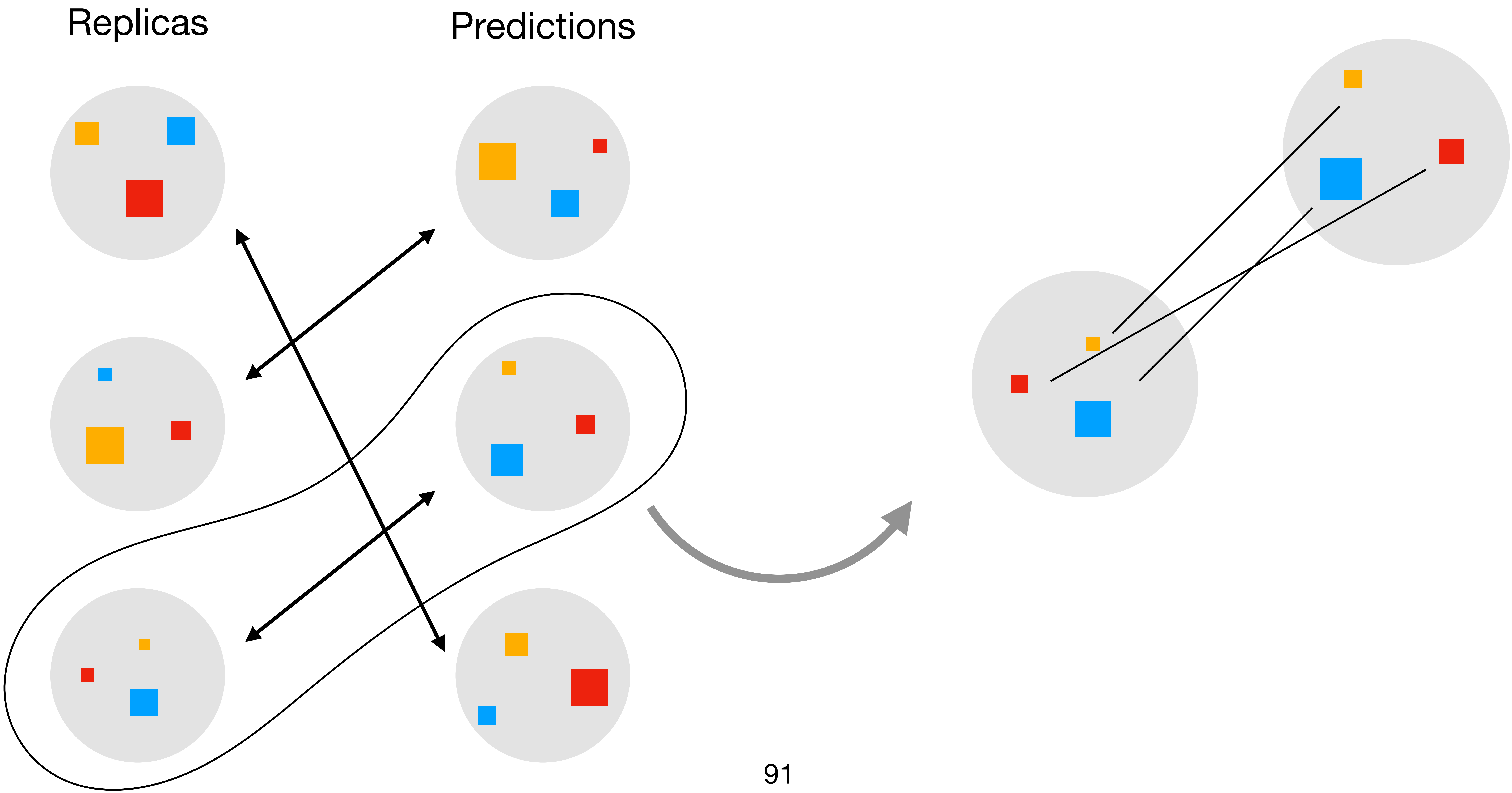
Hungarian $\qquad k(x,y) = ||x-y||^2$

MMD vanishes when p=q
but is time consuming

$$L_{proxy} = min_{x_i,y_j} k(x_i, y_j)$$



“Double Hungarian”

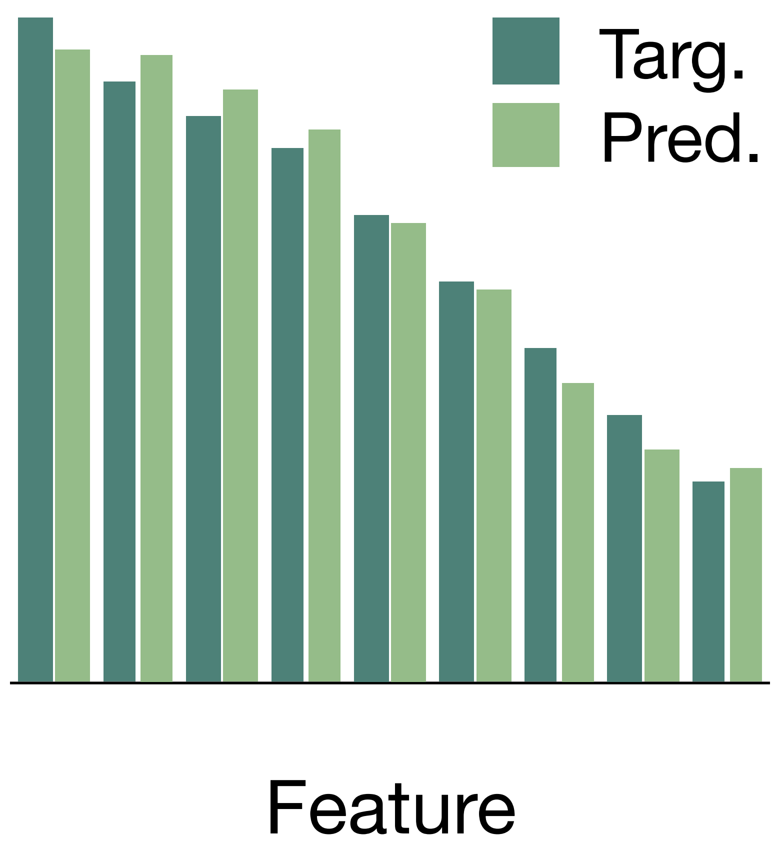


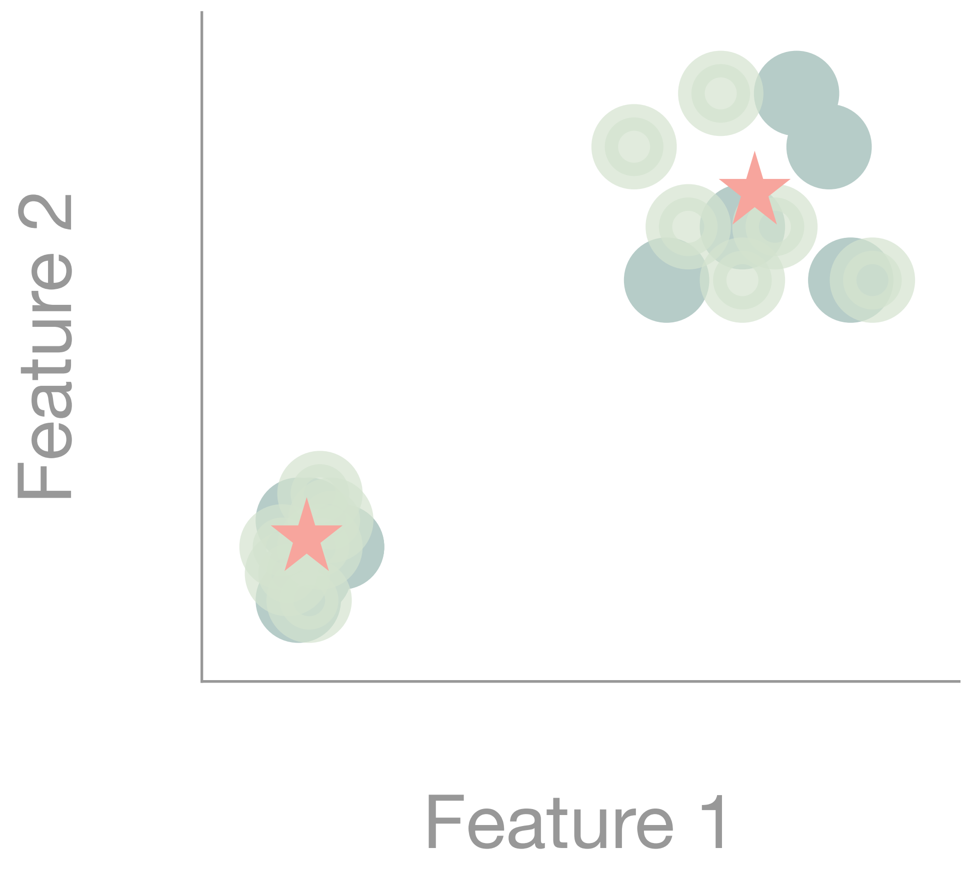
Goals

Marginal distributions

Reconstruct constituents

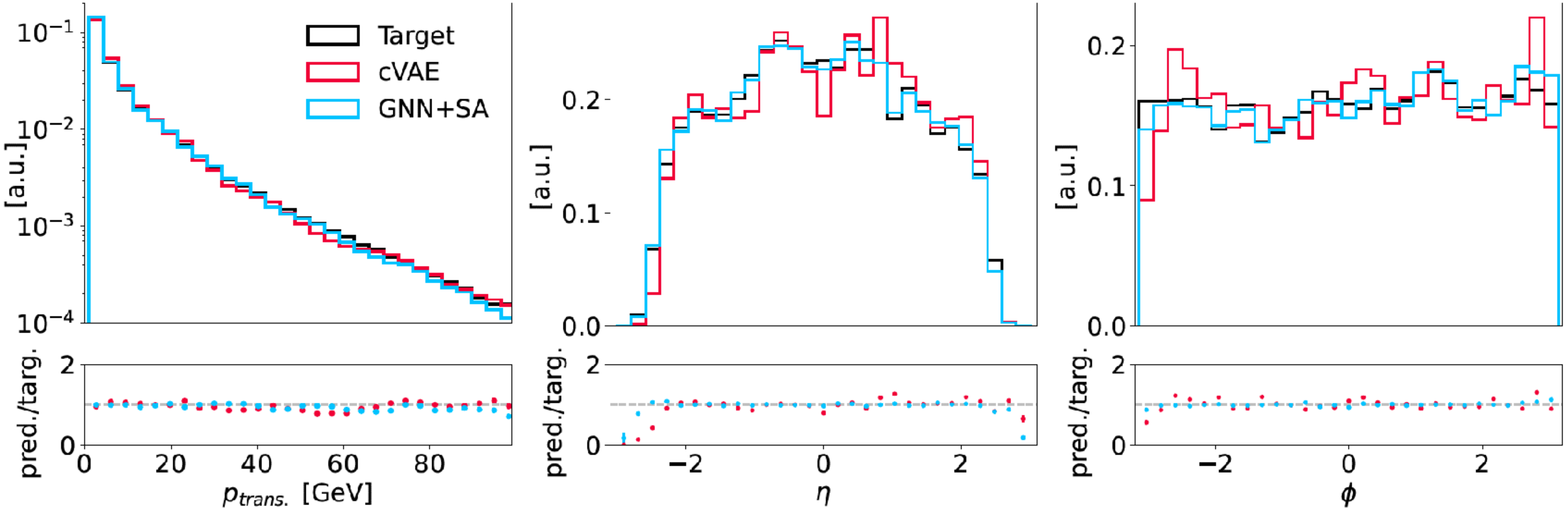
Resolution





N. Soybelman

Marginal distributions



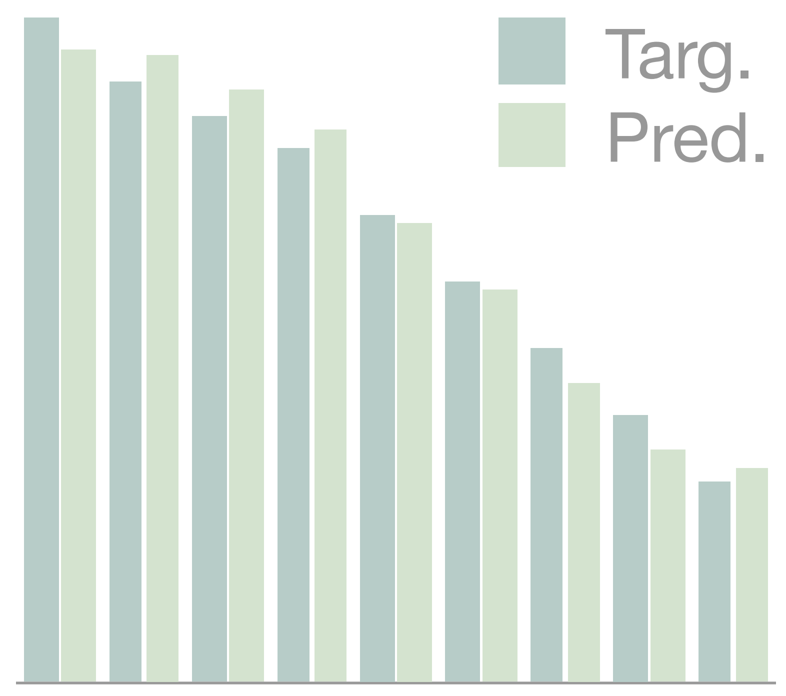
- 1D marginal distributions similarly good for both cVAE and GNN

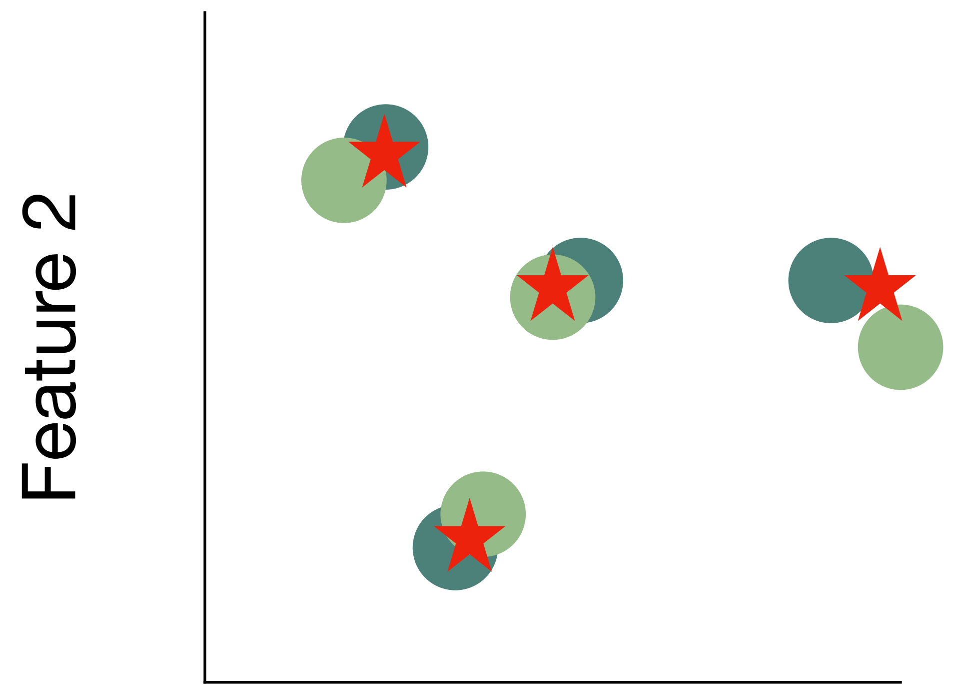
Goals

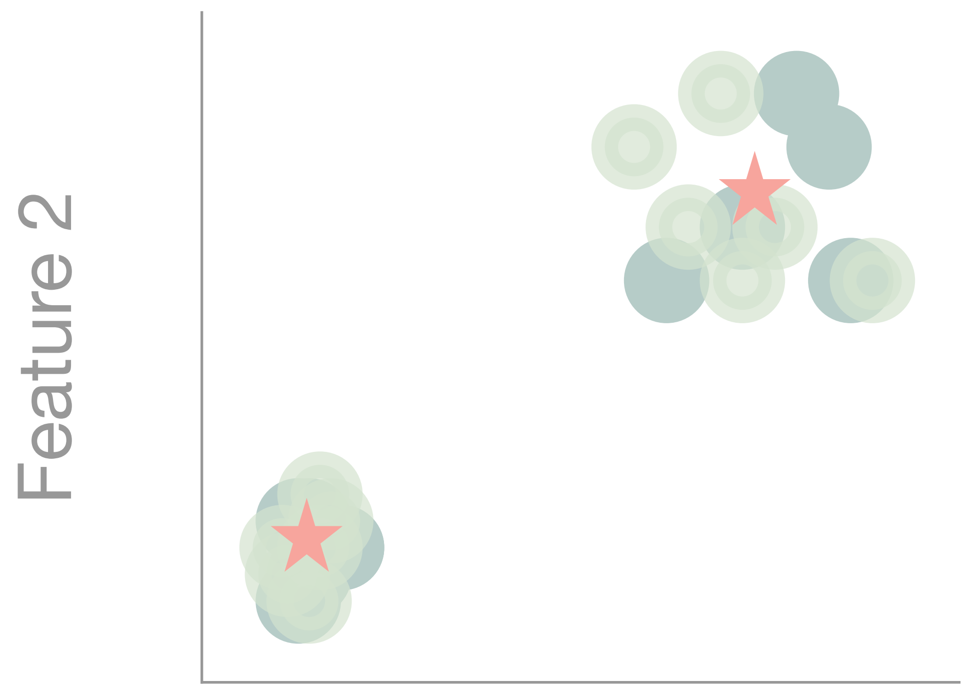
Marginal distributions

Reconstruct constituents

Resolution

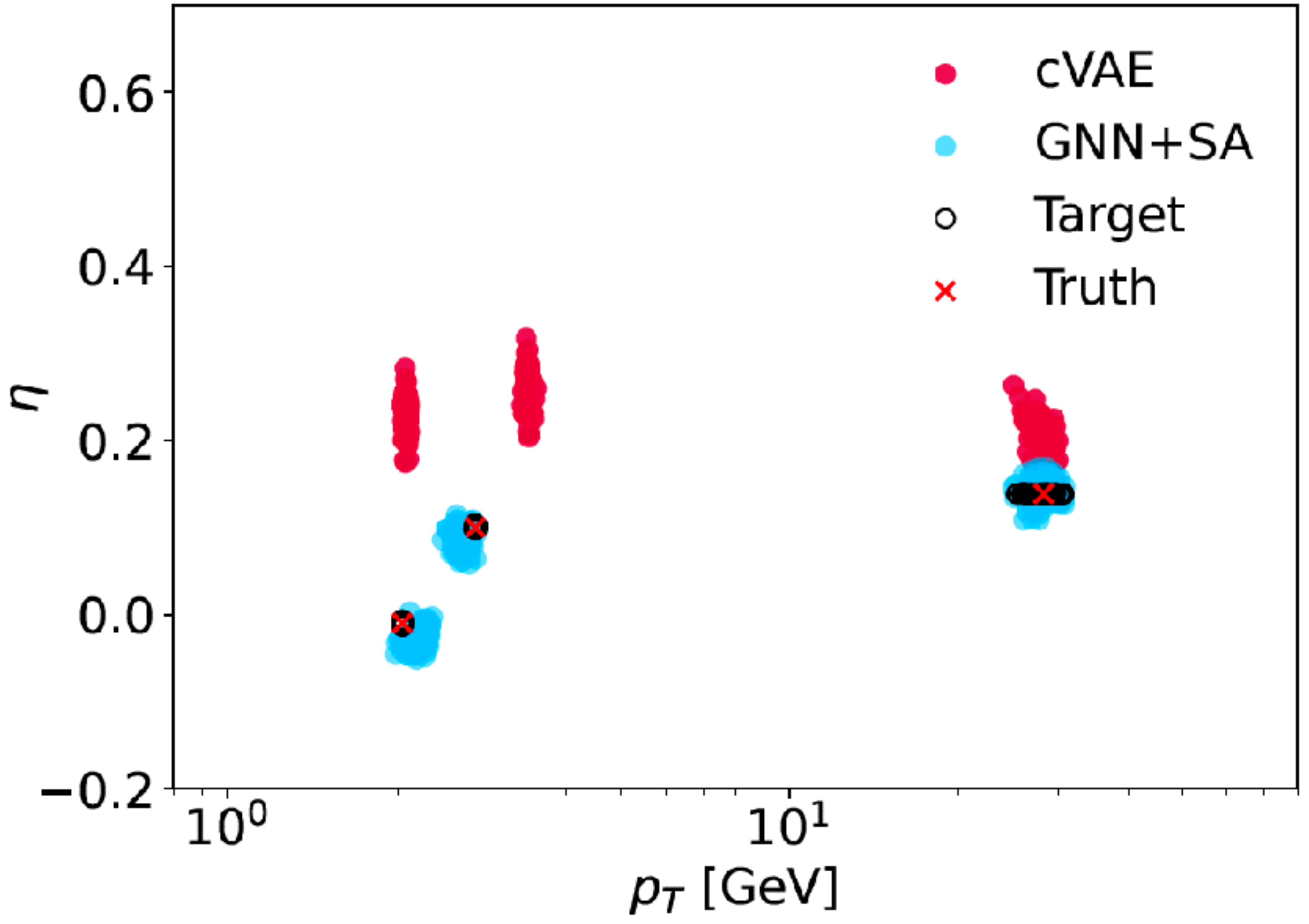


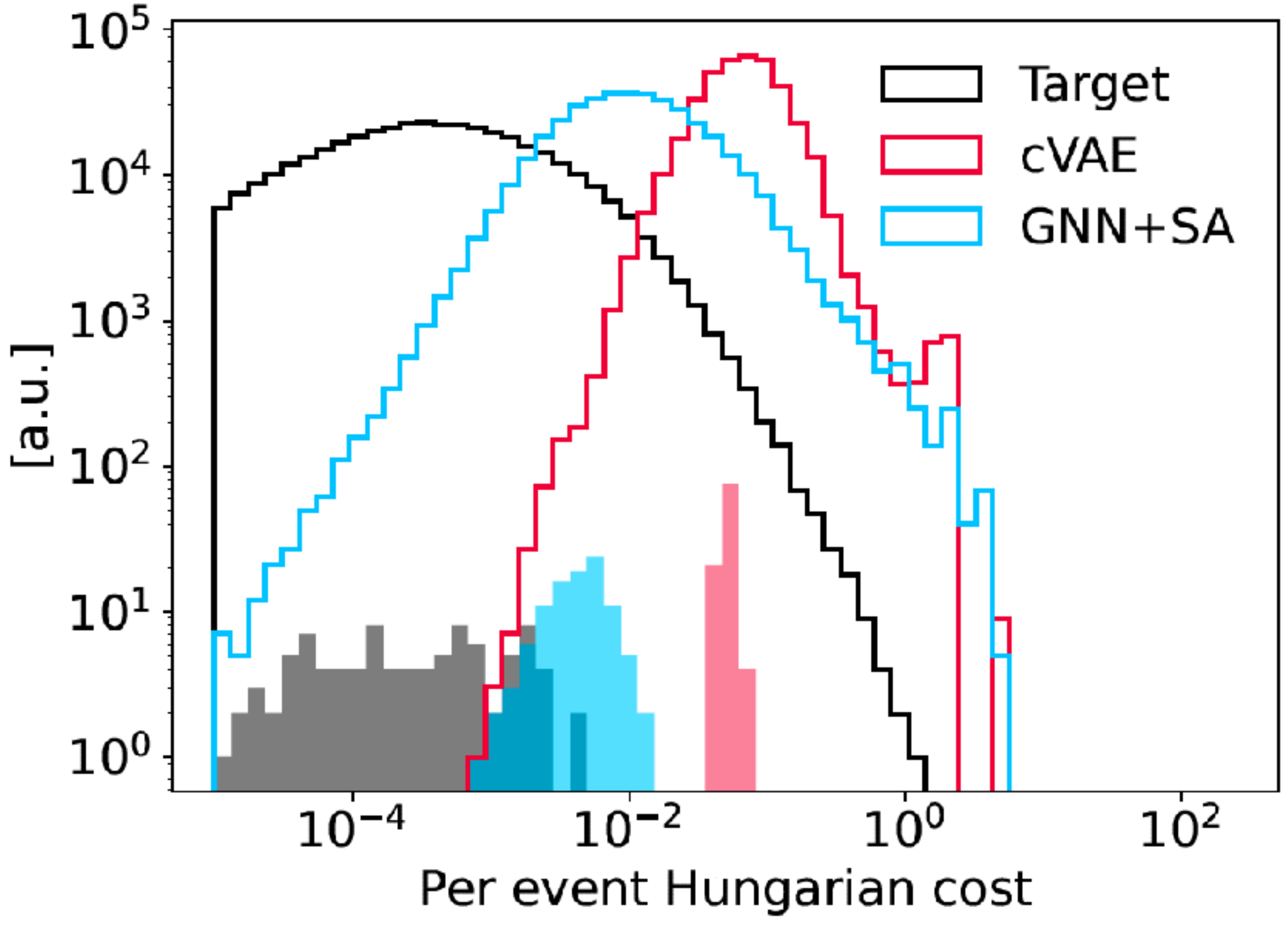




N. Soybelman

Reconstruct Constituents



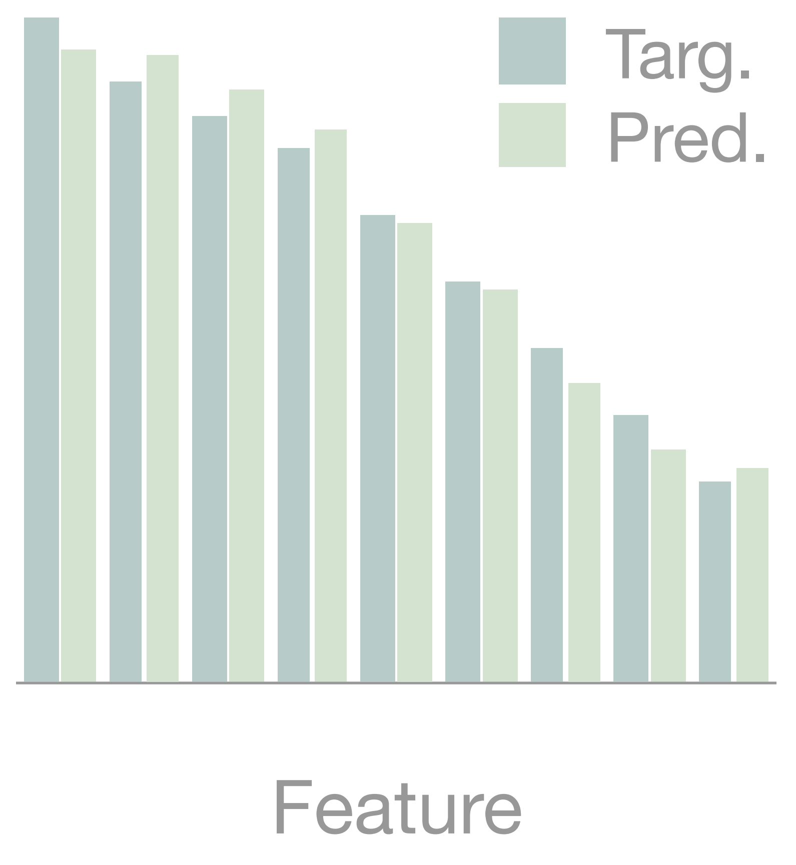


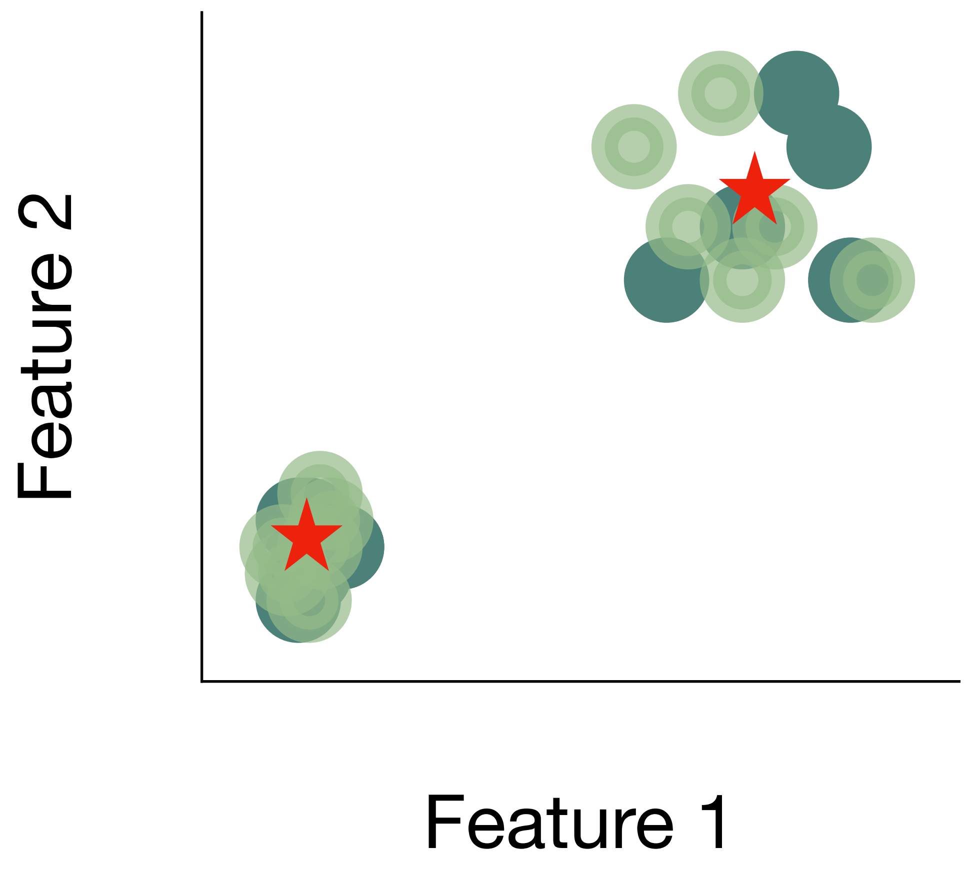
Goals

Marginal distributions

Reconstruct constituents

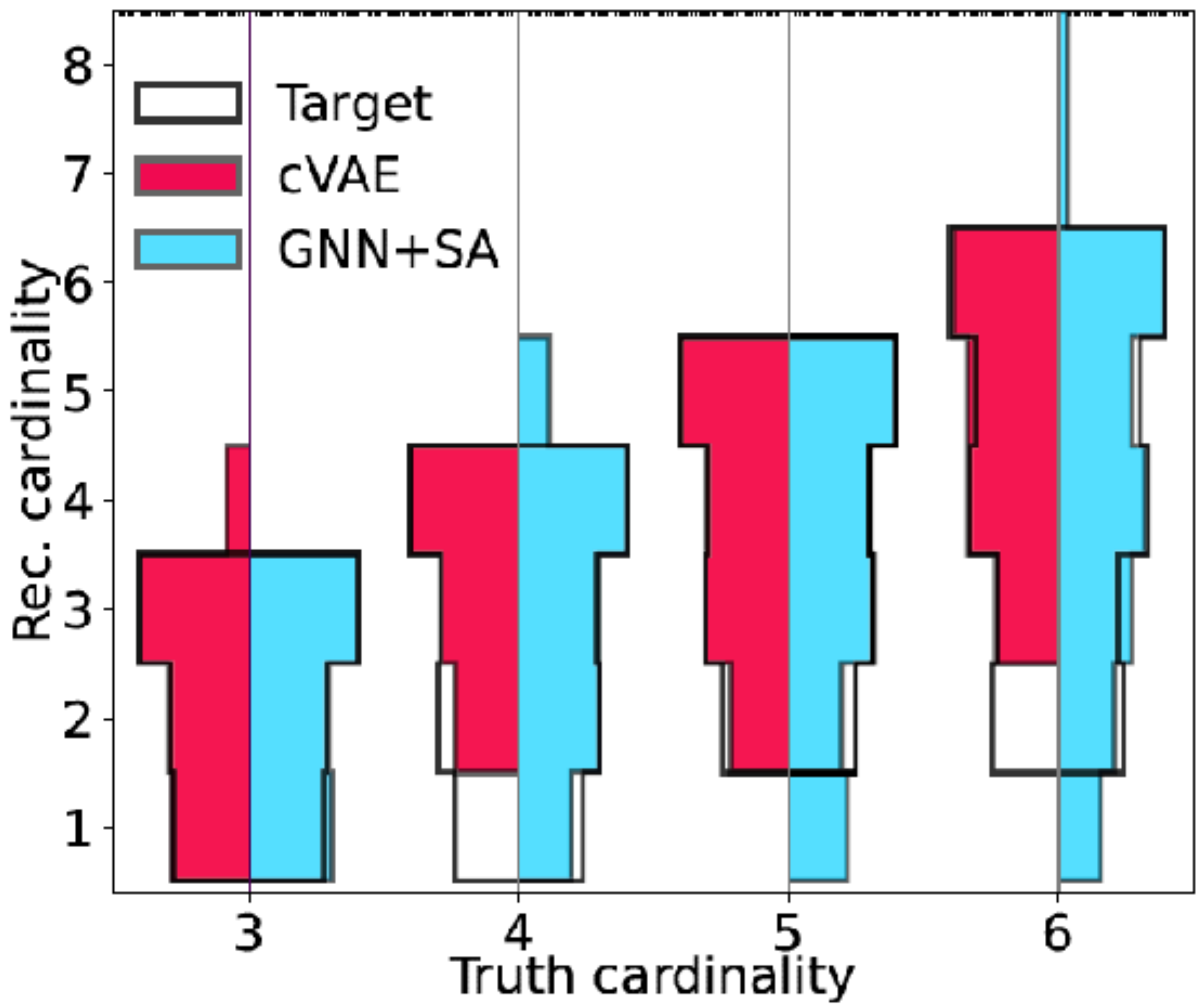
Resolution



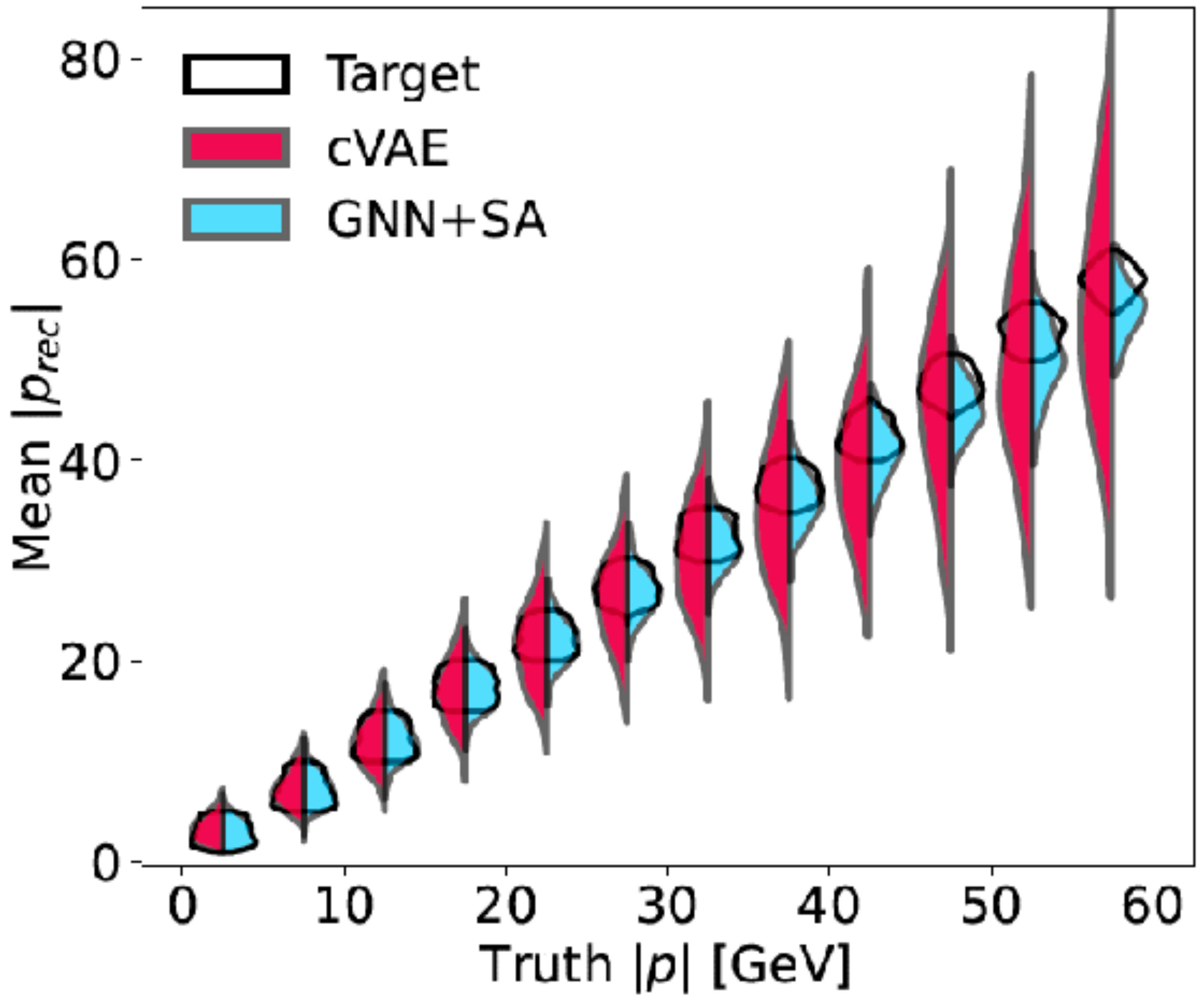


N. Soybelman

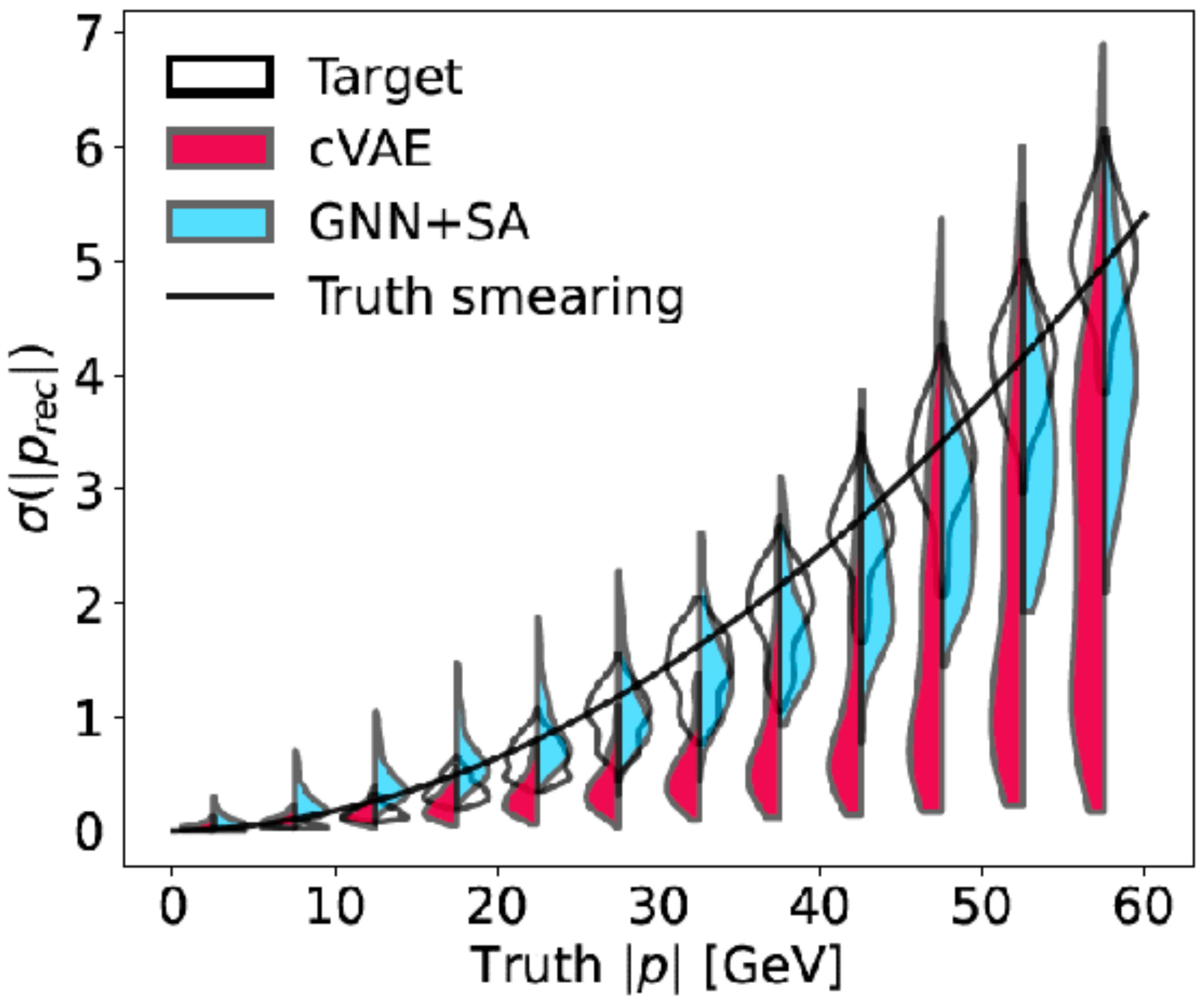
Resolution



(a) $p\left(N_R \middle| N_T\right)$



(b) $p\left(\overline{(|p|_R)} \middle| |p|_T\right)$



(c) $p\left(\sigma(|p|_R) \middle| |p|_T\right)$

Conclusion

- Investigated feasibility of set generation via attention-based GNN architecture, using replicas important to learn the resolution
- Marginal distribution well-modeled by baseline (cVAE) and GNN with Slot att, however, 1D distributions can be deceptive
- The GNN+SA model outperforms the baseline model and better captures key properties of the target distribution. It performs better in predicting mean and variance of constituents

Syllabus

- ✅ Graph Neural Nets
  - ✅Set to Graph
- ✅ Attention is all you need
  - ✅ Transformers, ✅ Slot Attention (SA)
  - ✅ Set Prediction Networks with a Transformer and SA (TSPN-SA)
- ✅ Constrained Variational Auto Encoder (cVAE)
- Particle Flow
  (Reconstructing Particles in Jets using TSPN-SA,
  Hyper-Graph PFlow [HGPflow])
- ✅ Simulation of PF Objects (Using TSPN-SA, cVAE)