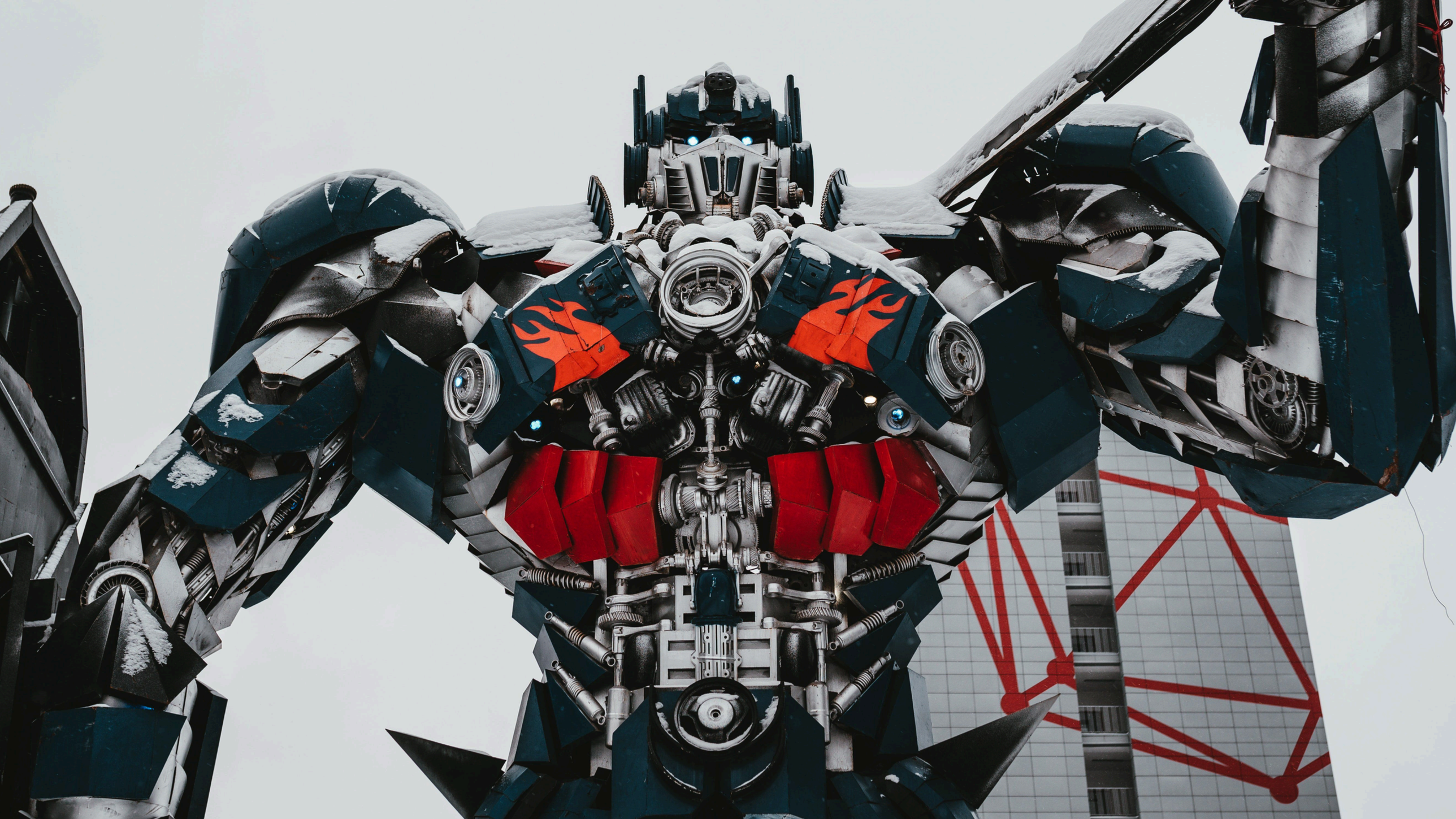
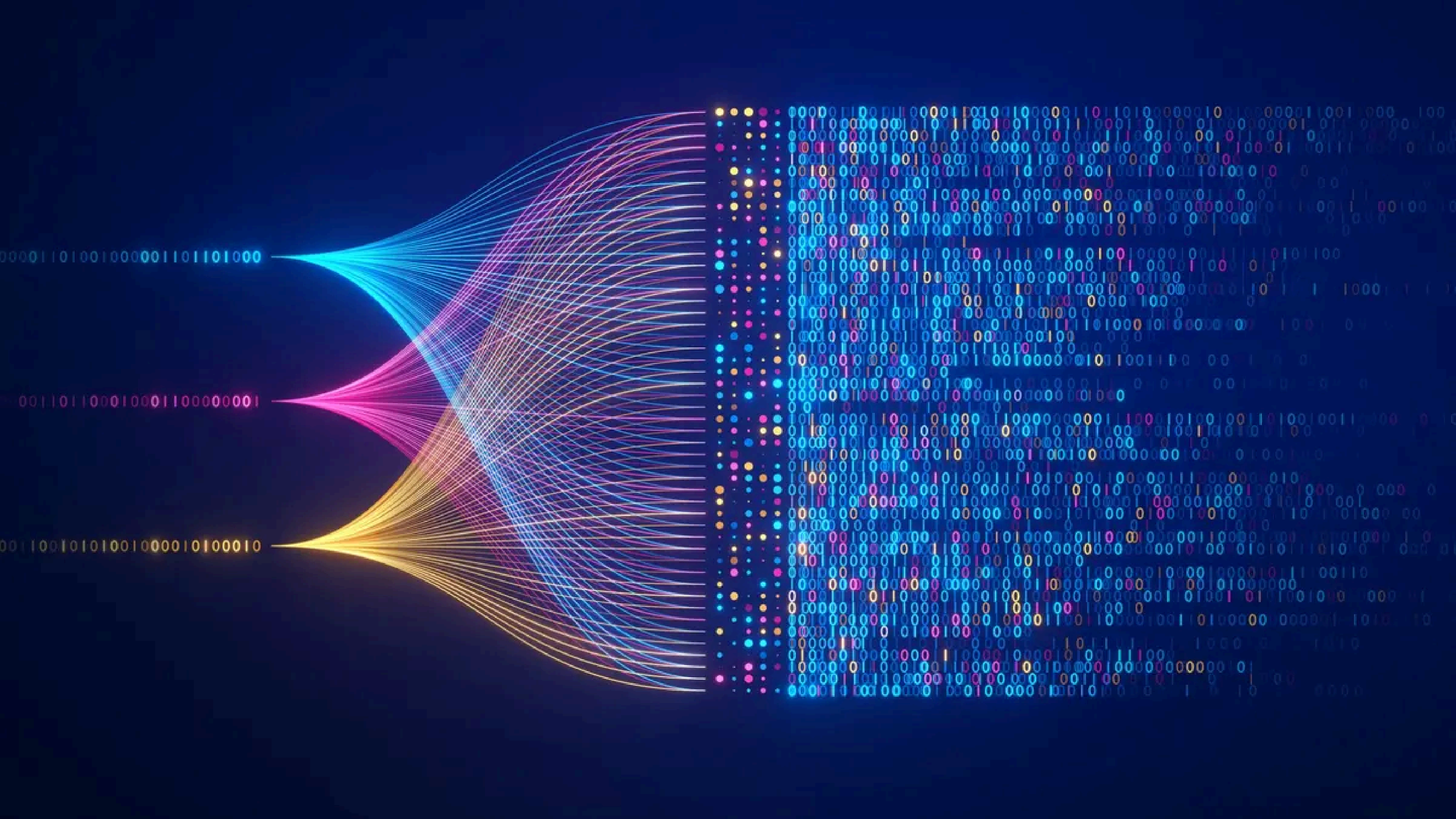
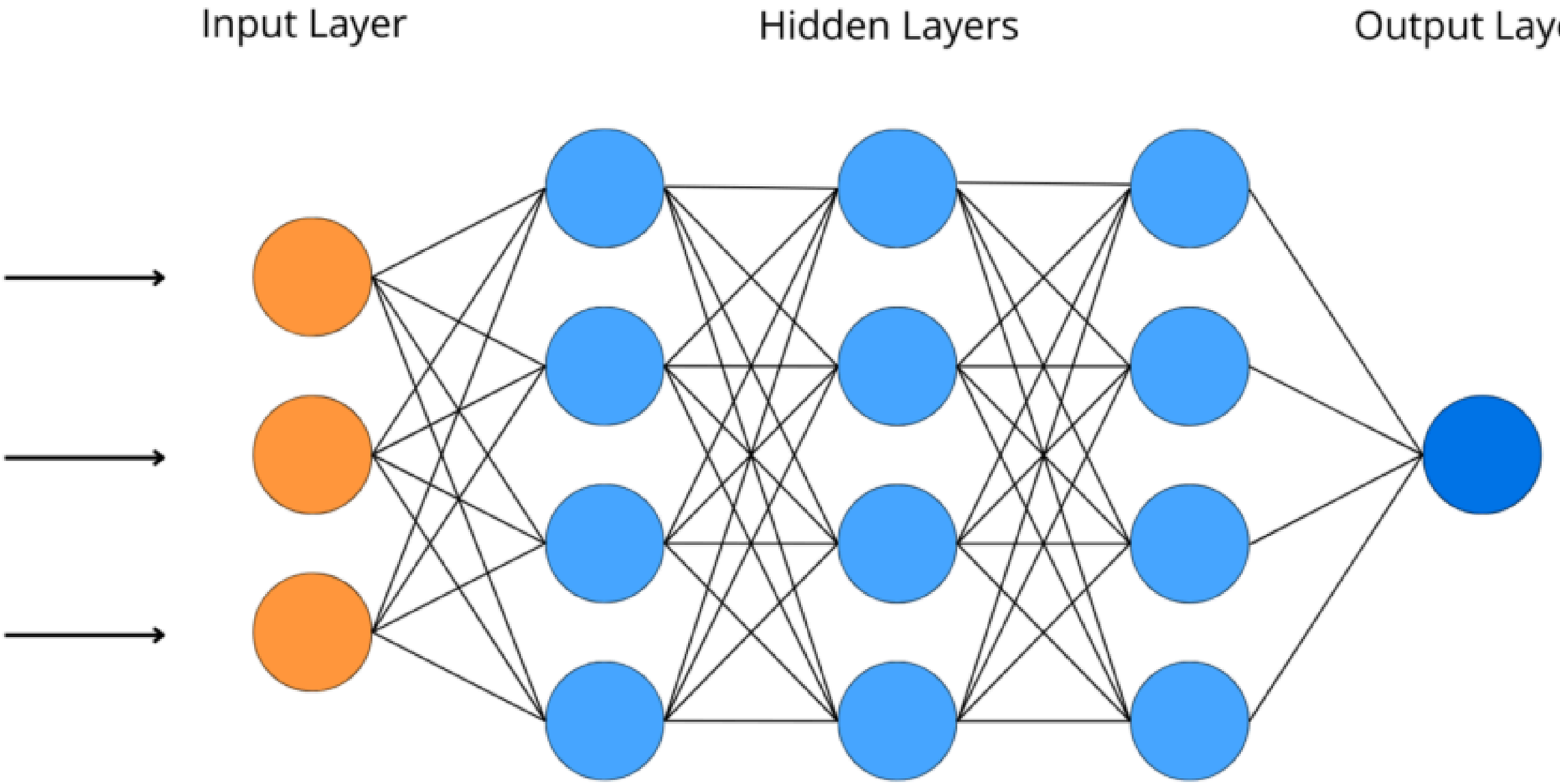
Attention Is All You Need

M3hdi Rahmani

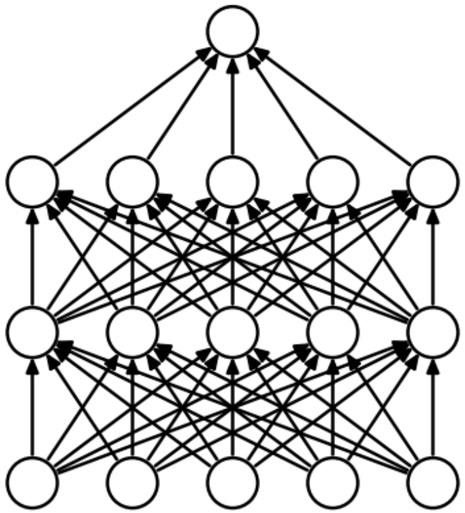




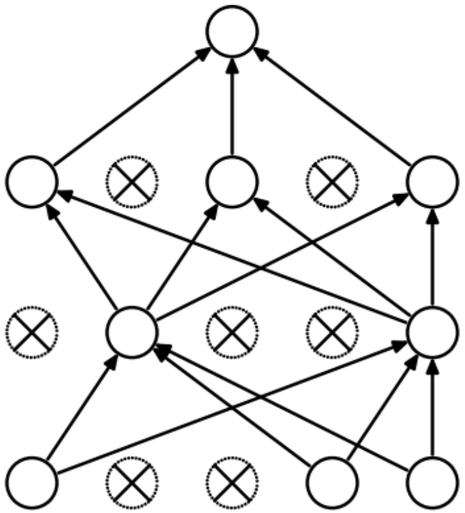
ChatGPT'S Neural Network Architecture



Dropout



(a) Standard Neural Net

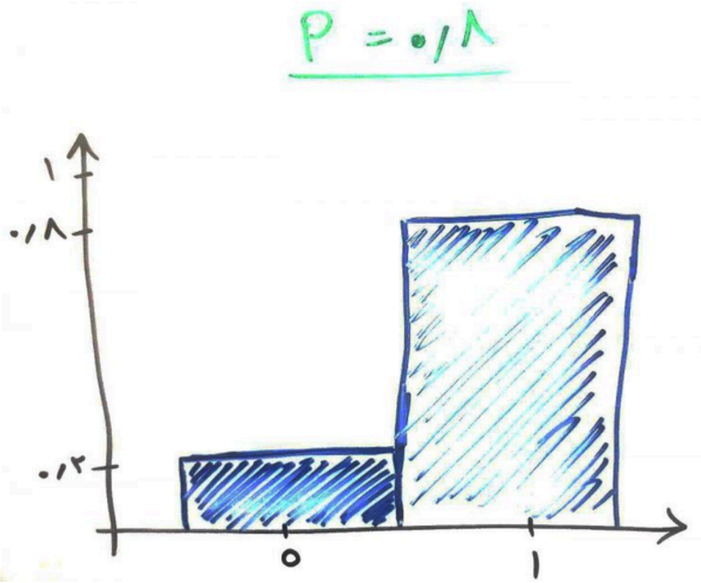


(b) After applying dropout.

Left: A standard neural net with **2 hidden layers**.

Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Bernoulli distribution

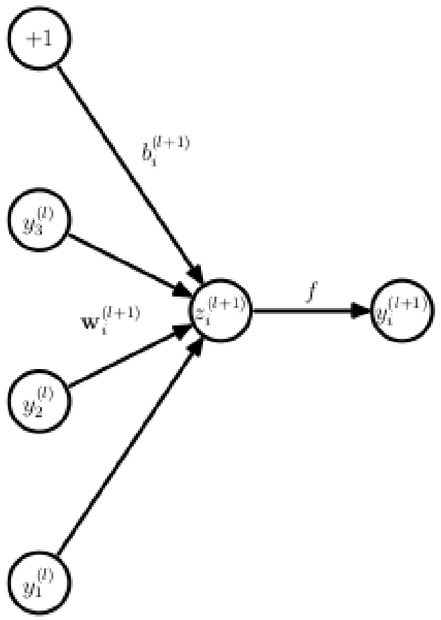


$$P(x \text{ "success"}) = \frac{n!}{x!\,(n-x)!}\,P^{x}\,(1-P)^{(n-x)}$$

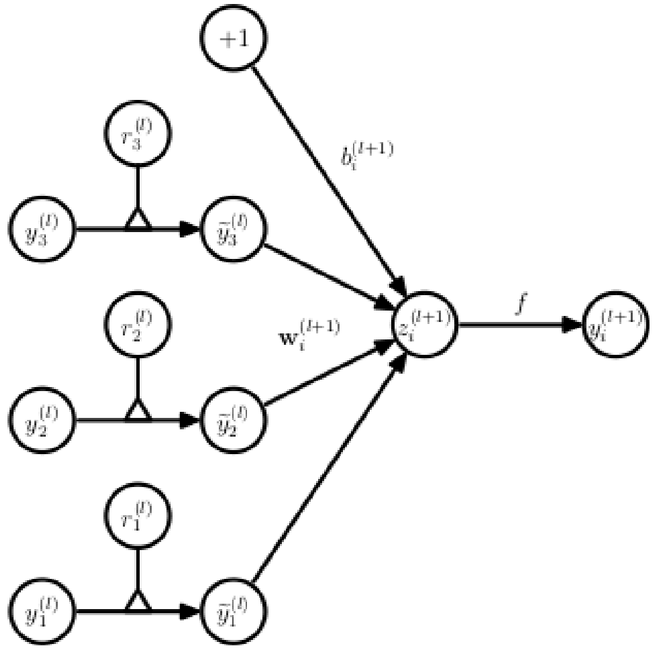
تعداد مشاهدات

تعداد موفقیت مورد نظر

احتمال موفقیت



(a) Standard network

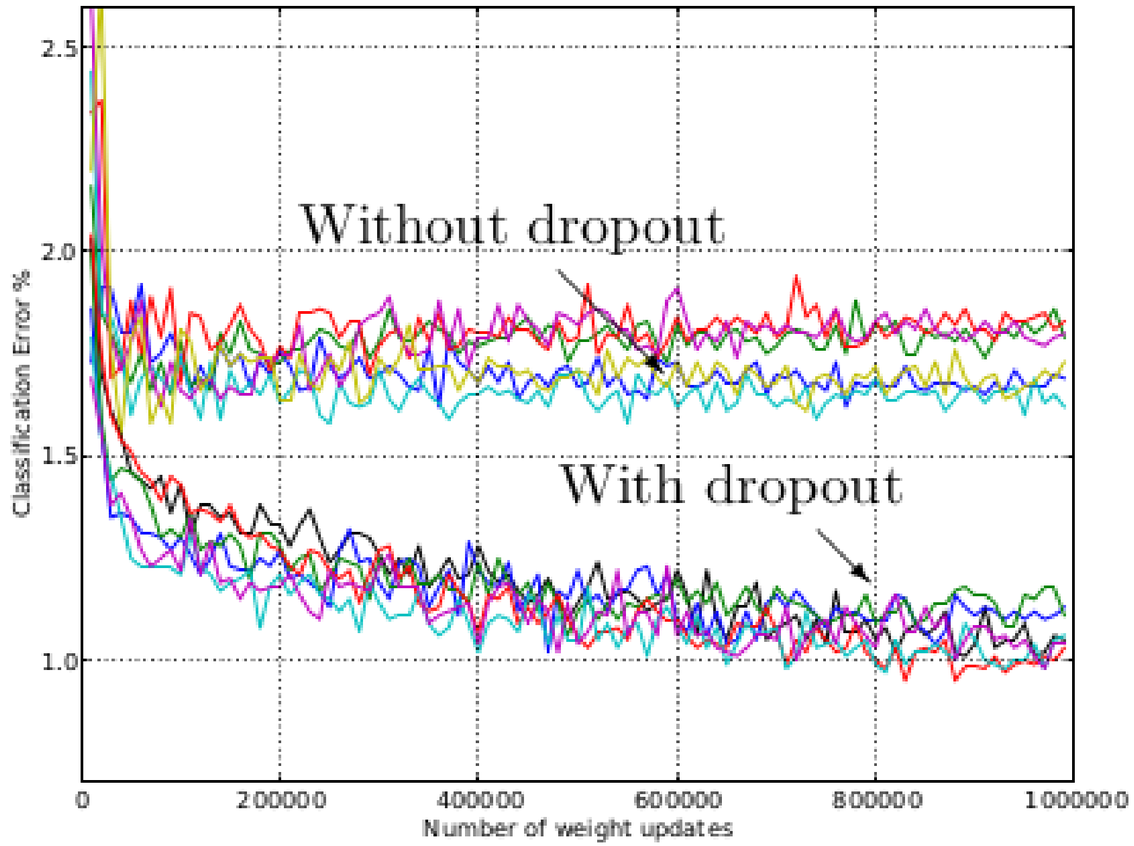


(b) Dropout network

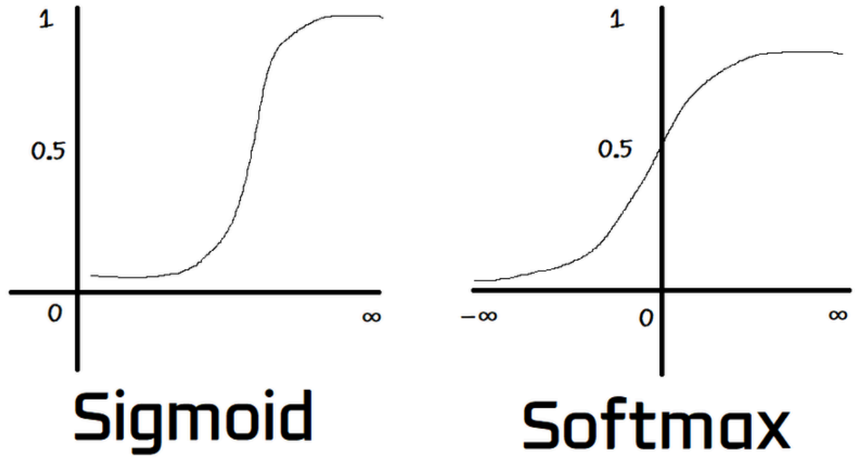
$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^{l} + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

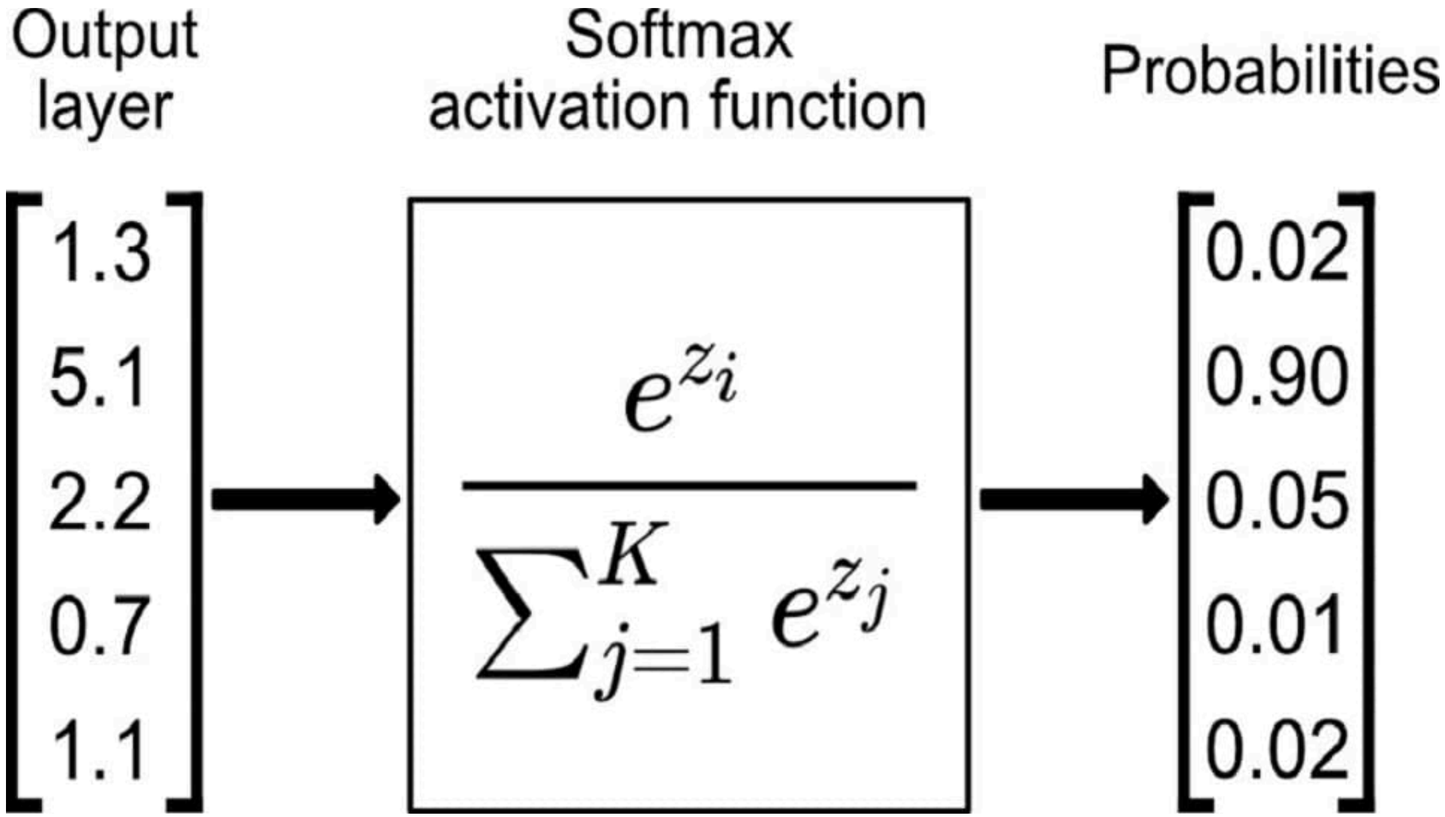
f (x) = 1/ (1 + exp(−x))

Test error for different architectures



softmax

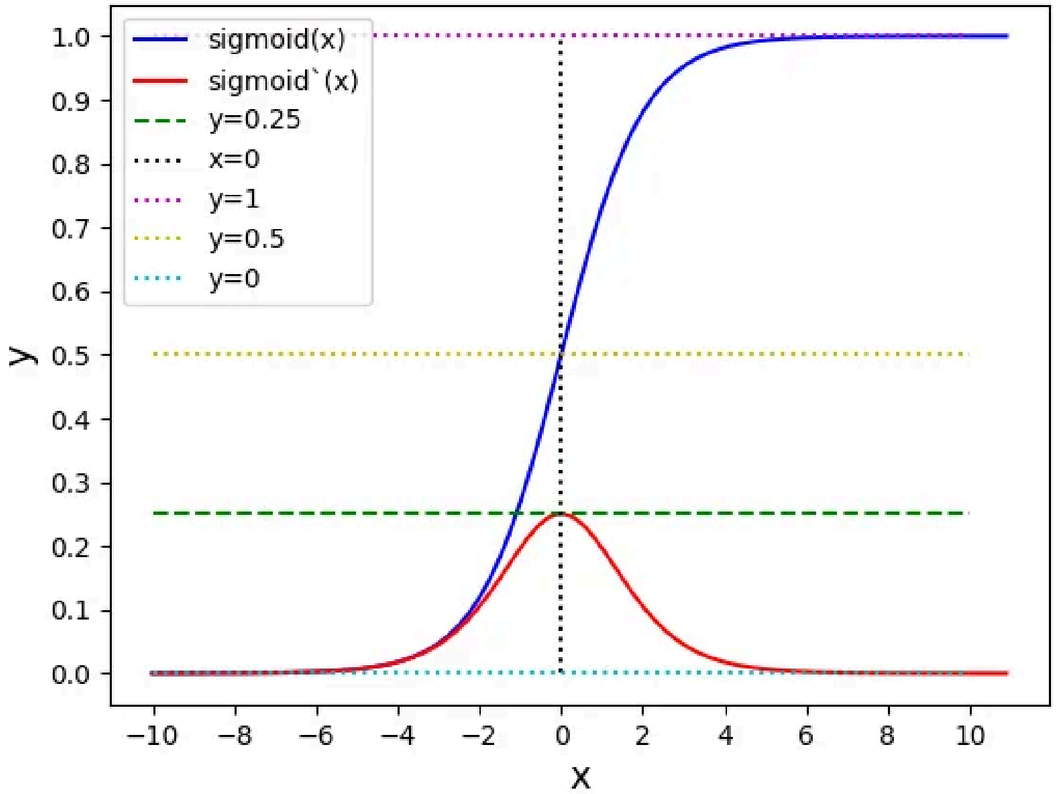

1
0.5
0
∞
Sigmoid
1
0.5
−∞
0
∞
Softmax


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

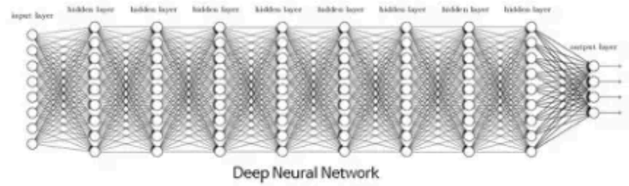
Vanishing Gradient

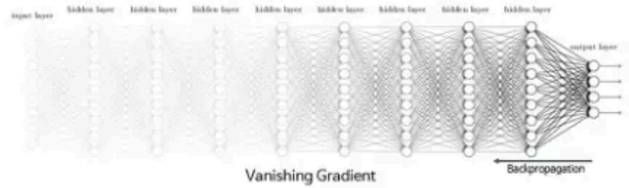
$$W_{new} = W_{old} - \eta * \frac{\partial C}{\partial w}$$

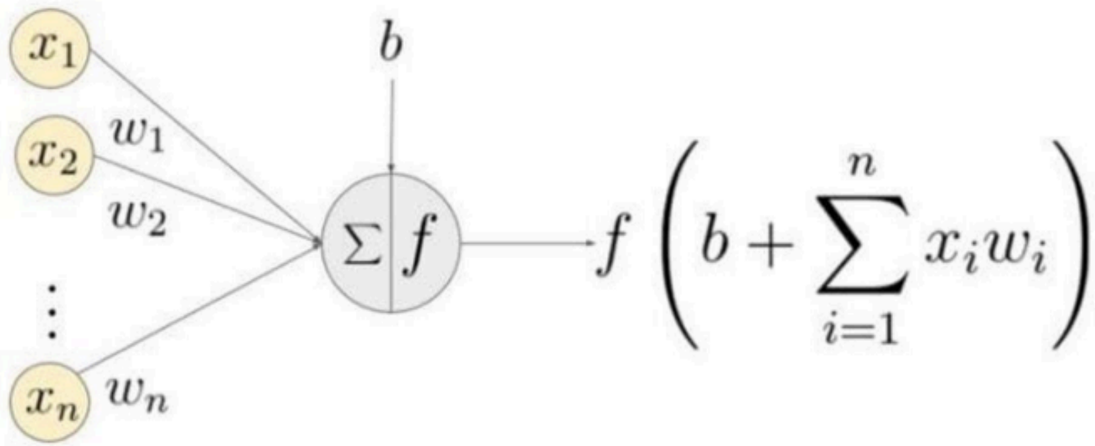
learning rate



Sigmoid: Vanishing Gradient Problem







Method to overcome the problem

01 - Activation function

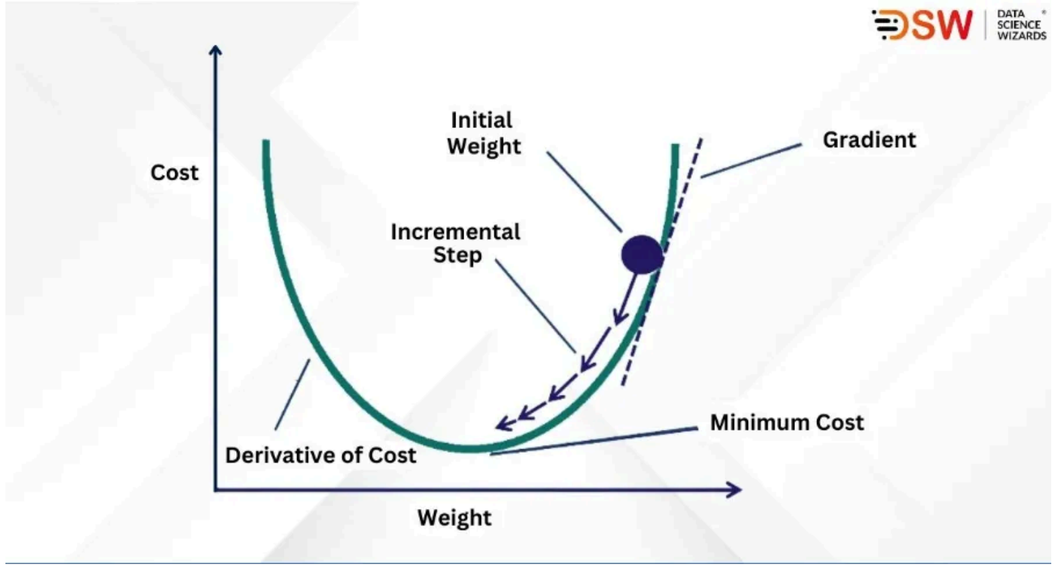
ReLu **activation functions** keep **linearity** for regions where sigmoid and TanH are saturated, thus responding better to gradient vanishing / exploding. You can also use different types like Leaky-ReLu, Randomized ReLu, etc.

02 - Cross Entropy Loss Functions

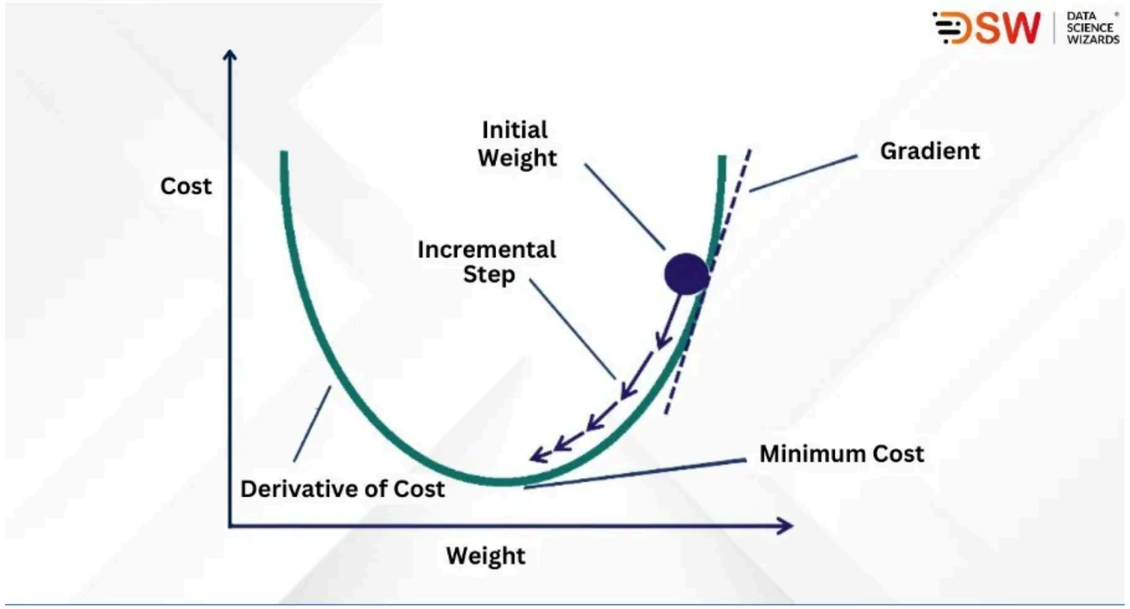
$$L = -\frac{1}{N}\left[\sum_{j=1}^{N}\left[t_j \log(p_j) + (1-t_j)\log(1-p_j)\right]\right]$$

for N data points where t_i is the truth value taking a value 0 or 1 and p_i is the Softmax probability for the i^{th} data point.

03 - Gradient Descent



Gradient Descent Explained with an Example



مقداردهی اولیه

$X_0 = 3$

$Learning\ rate = 0.01$

$$\frac{dy}{dx} = \frac{d}{dx}(x+5)^2 = 2 * (x+5)$$

تکرار اول

$$X_1 = X_0 - (learning\ rate) * (\frac{dy}{dx})$$

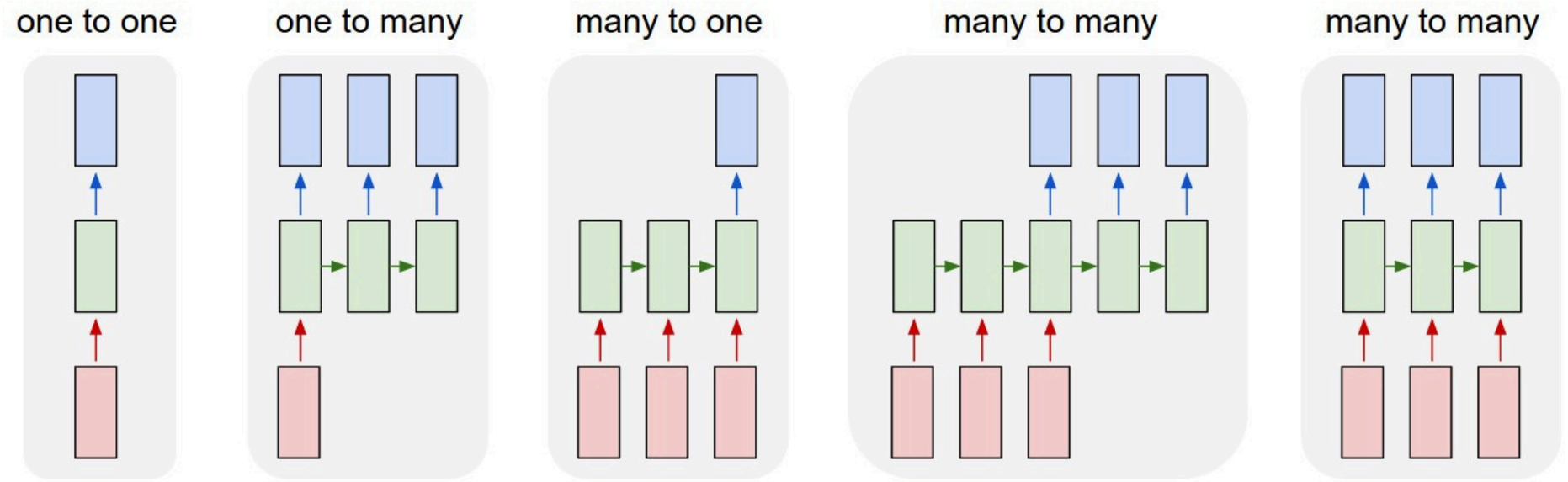
$$X_1 = 3 - (0.01) * \big(2 * (3+5)\big) = 2.84$$

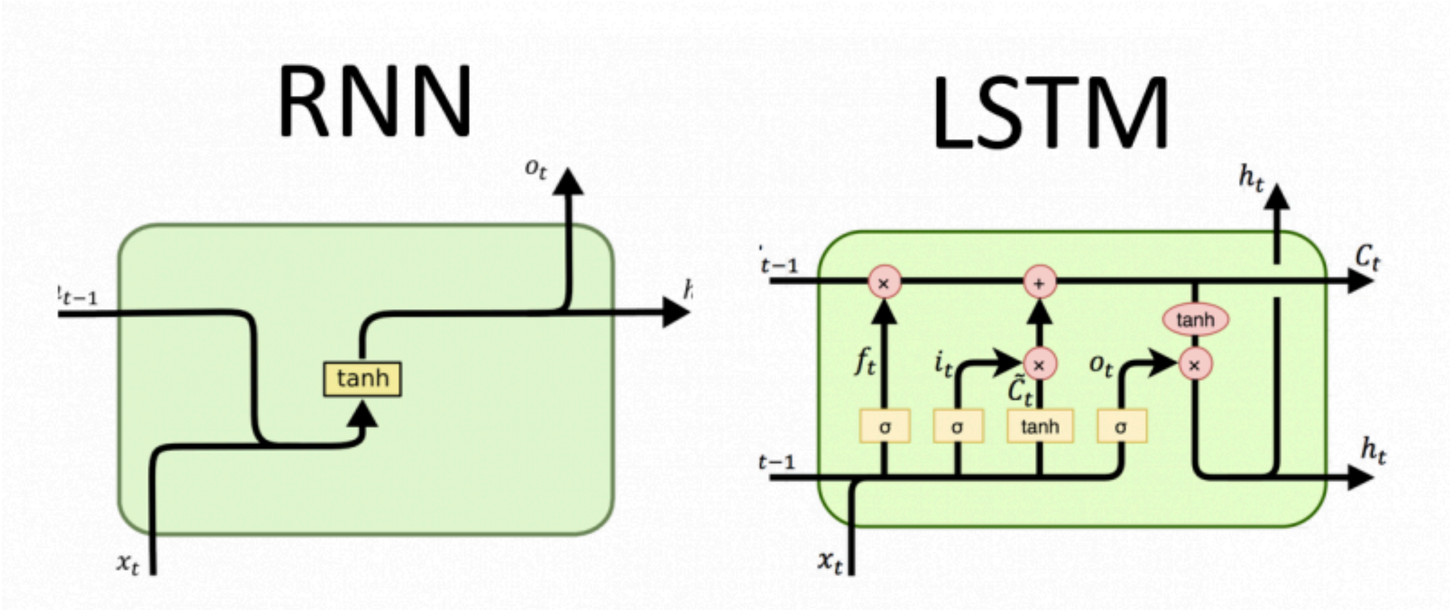
تکرار دوم

$$X_2 = X_1 - (learning\ rate) * (\frac{dy}{dx})$$

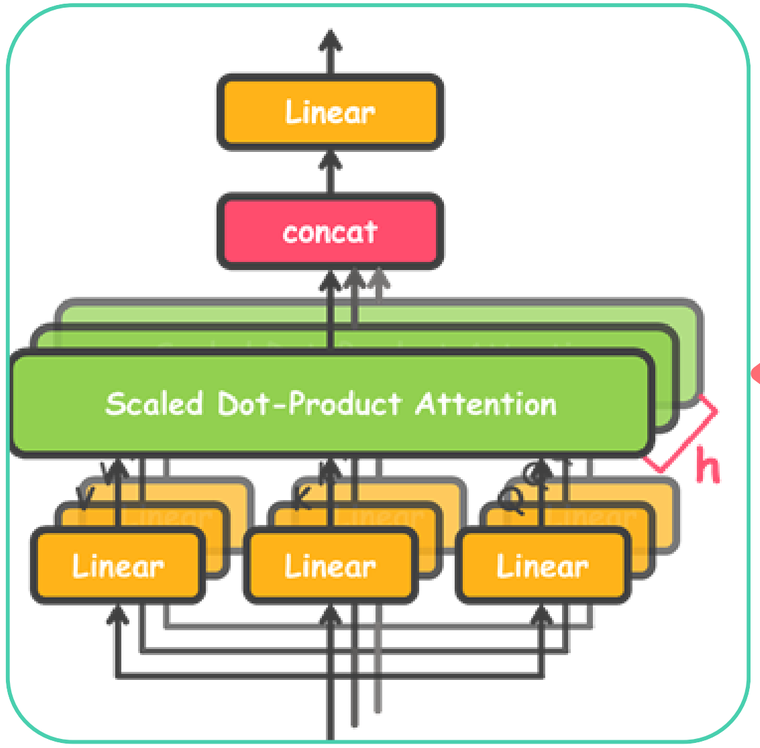
$$X_2 = 2.84 - (0.01) * \big(2 * (2.84+5)\big) = 2.6832$$

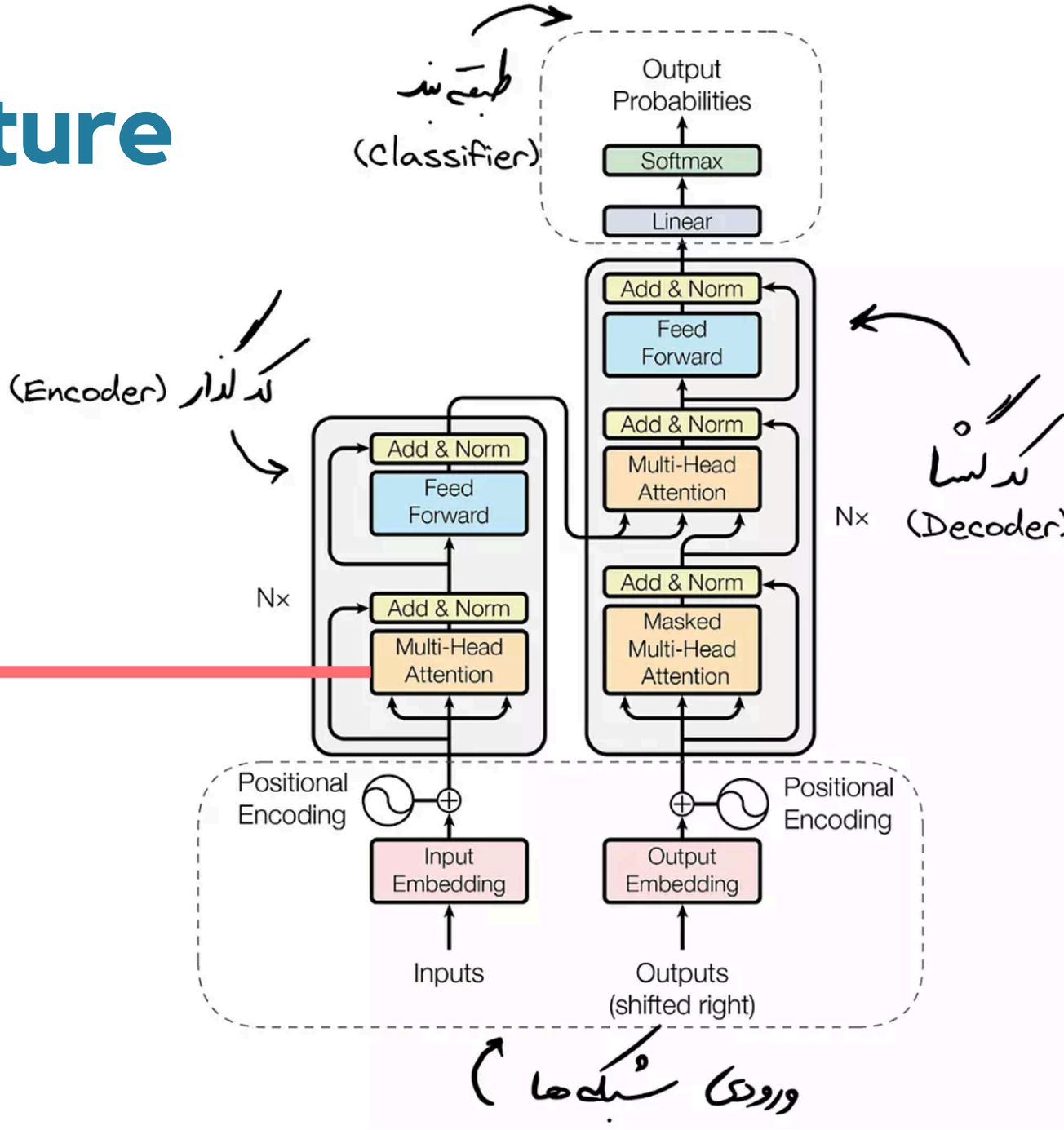
RNN & LSTM


one to one
one to many
many to one
many to many
many to many


RNN
LSTM
tanh
σ
σ
tanh
σ
tanh

Model Architecture





- “I like deep learning.”
- “I like NLP.”
- “I enjoy flying.”

Tokenization

```
["I", "like", "deep", "learning", ".", "NLP",
                "enjoy", "flying"]
```

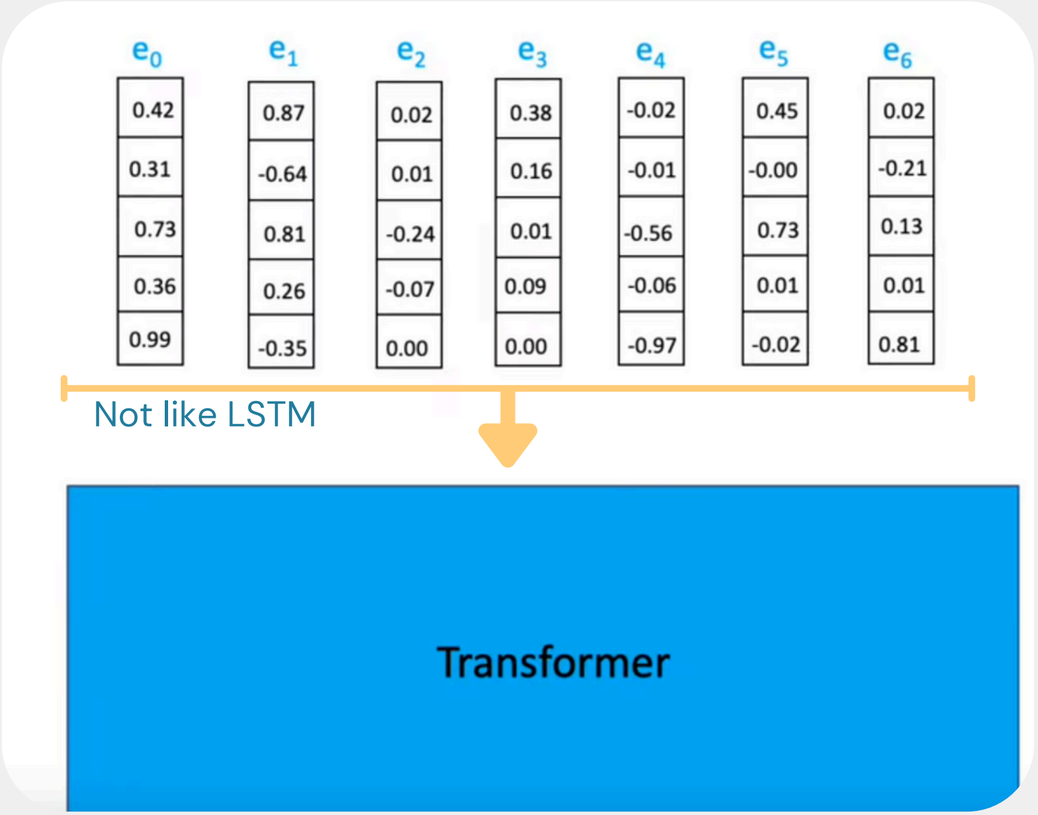
Vectorization (GloVe)

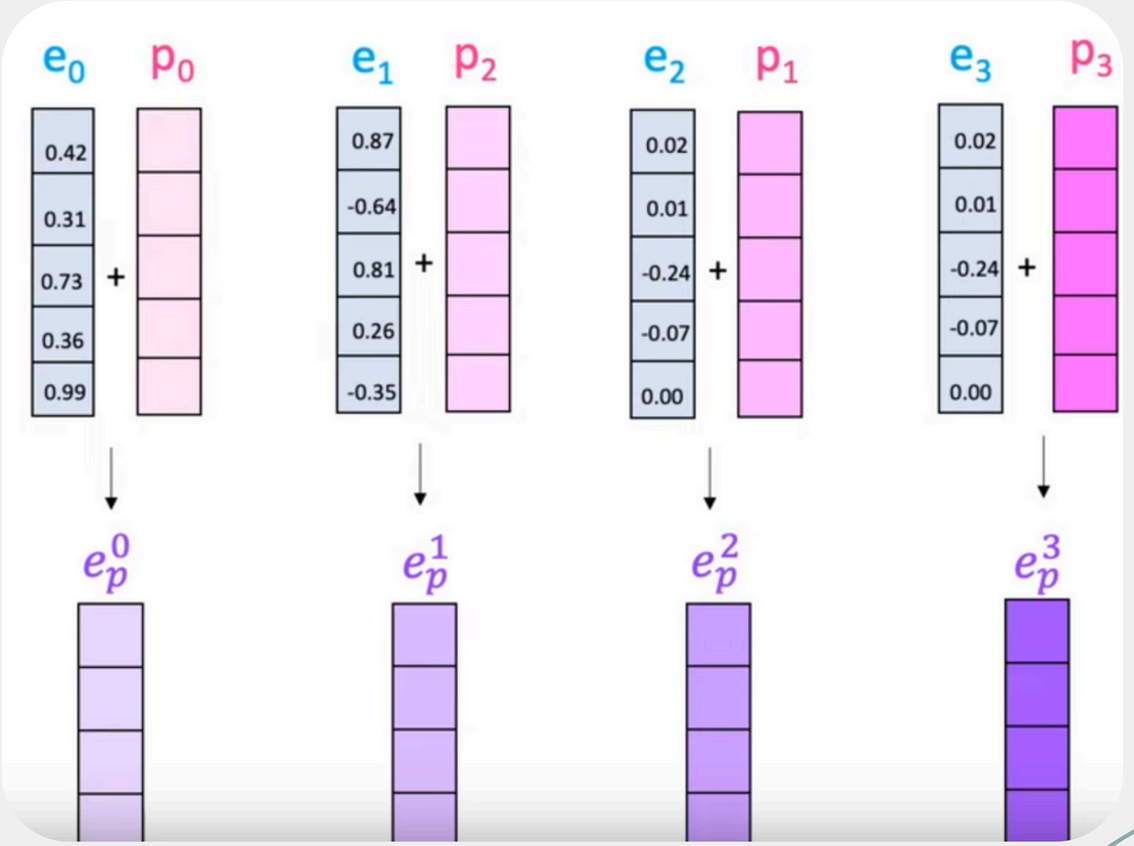
- “I like deep learning.”
- “I like NLP.”
- “I enjoy flying.”

```
[
  "I",  "like",  "deep",
  "learning",  ".",  "NLP",
  "enjoy",  "flying"
]
```

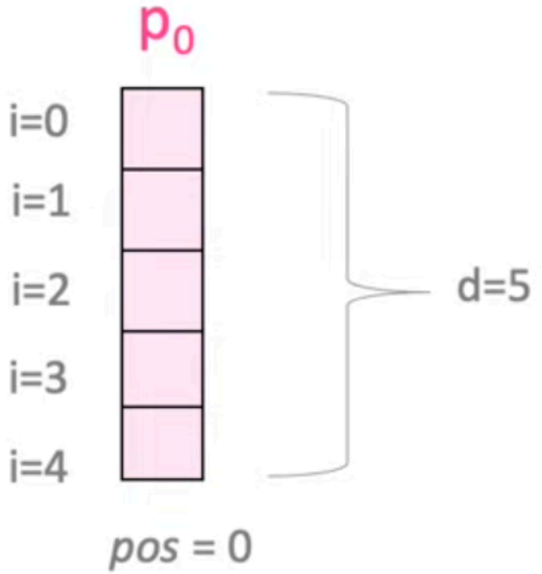
$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Positional Encoding

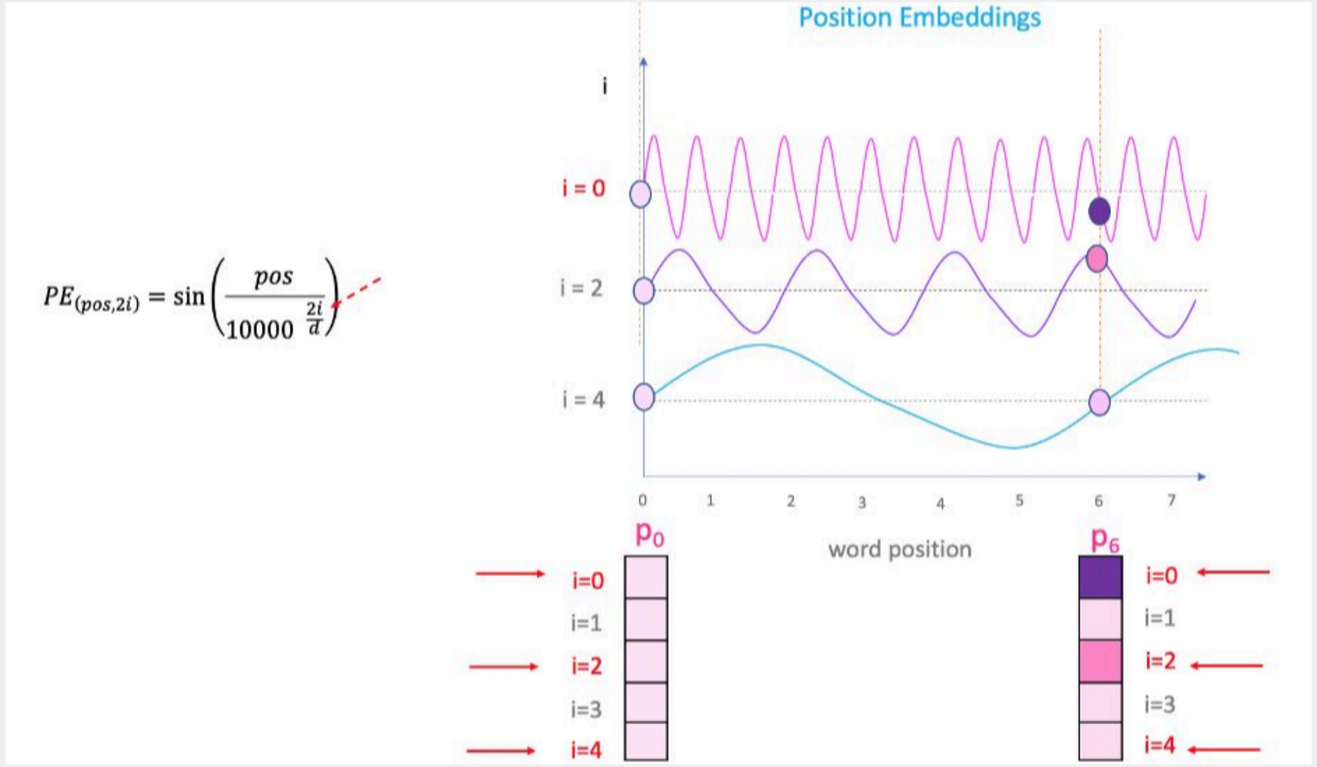




Positional Encoding



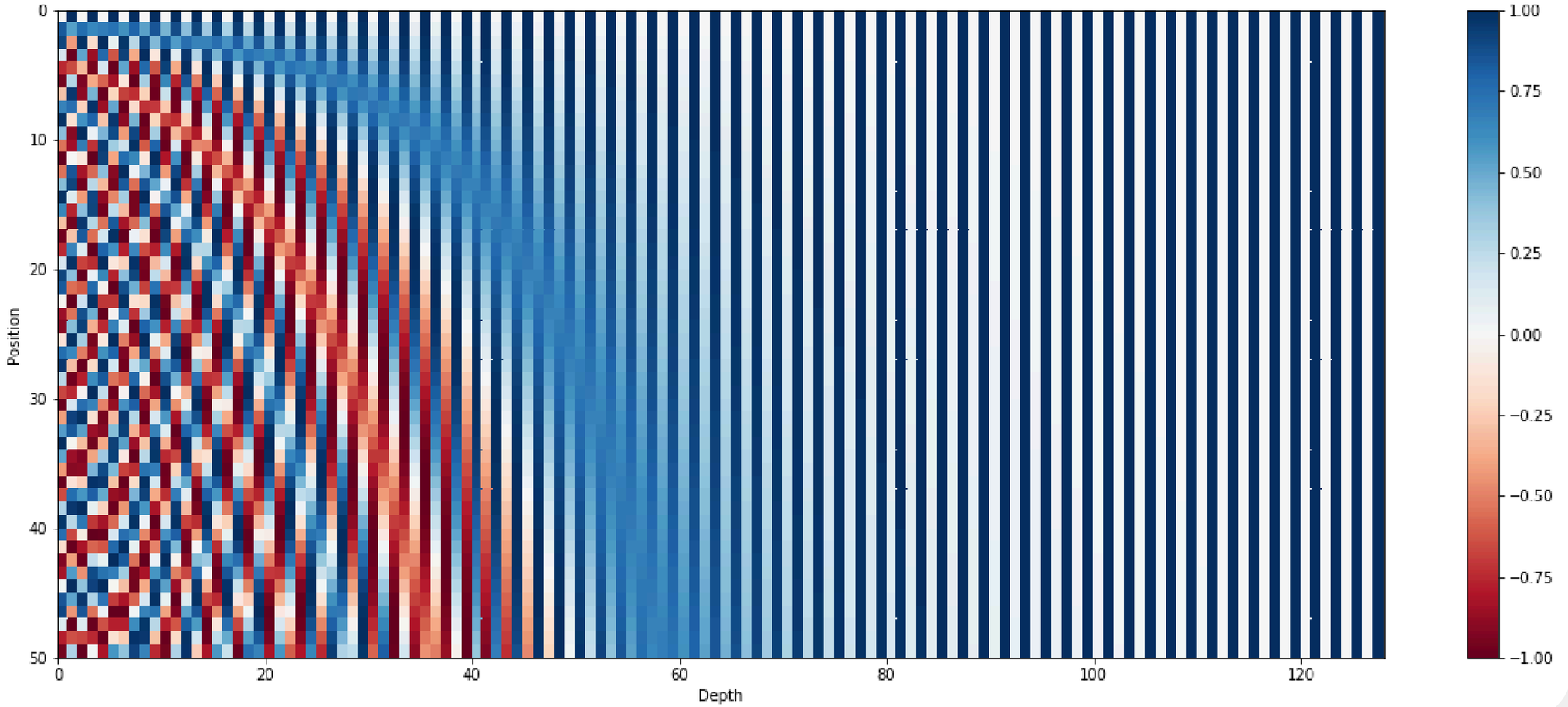
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$



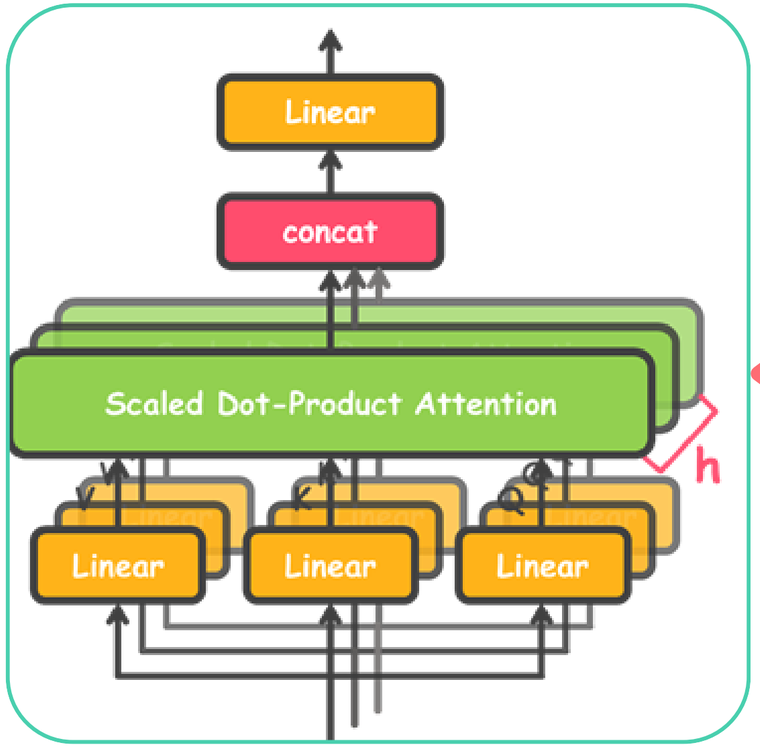
$$PE(pos, 2i) = \sin(pos/10000^{2i/d_{model}})$$

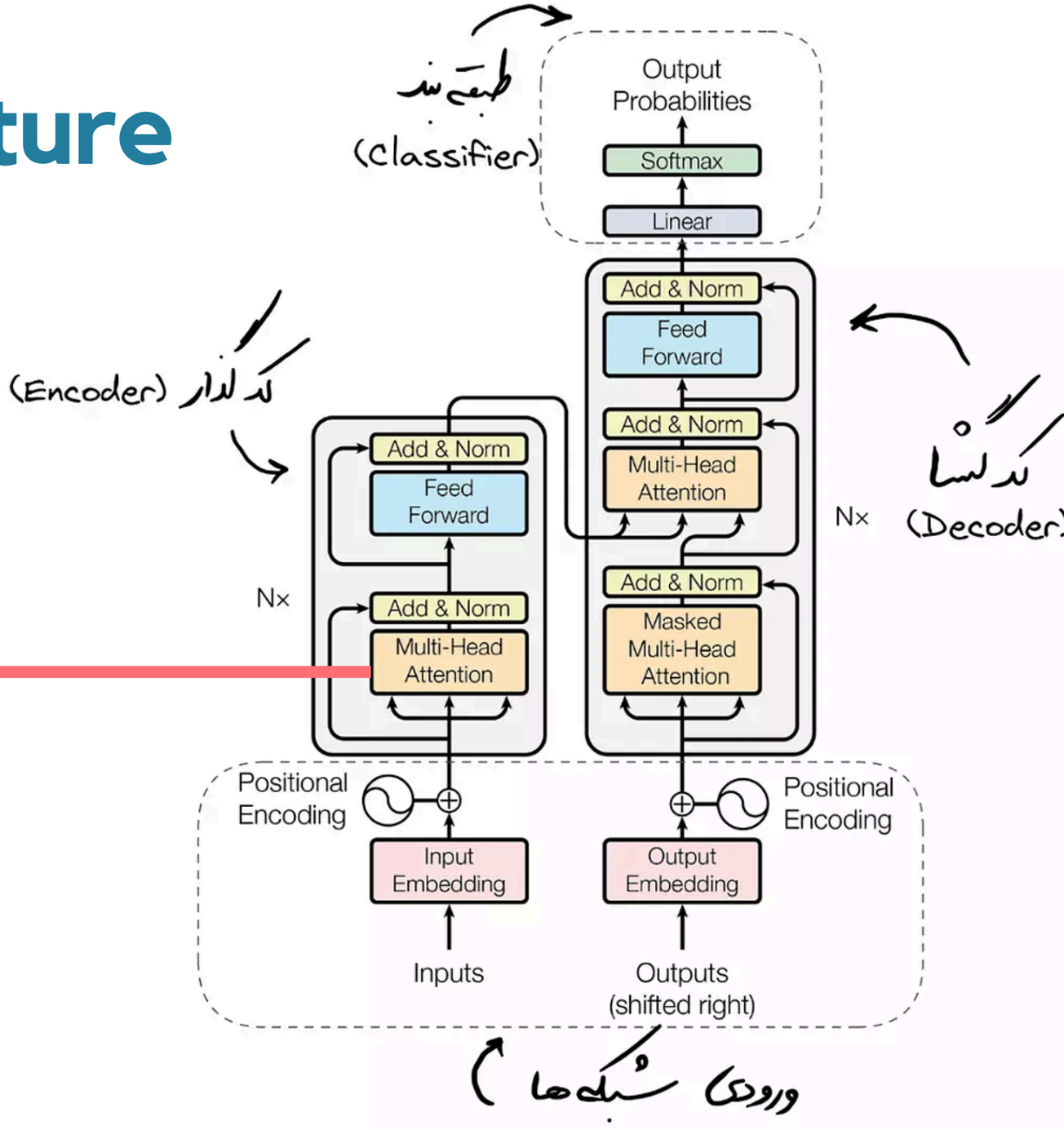
$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d_{model}})$$

Positional Encoding



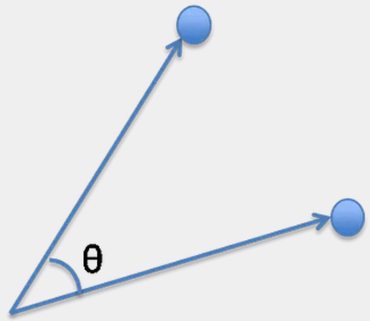
Model Architecture





Cosine Similarity

$$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



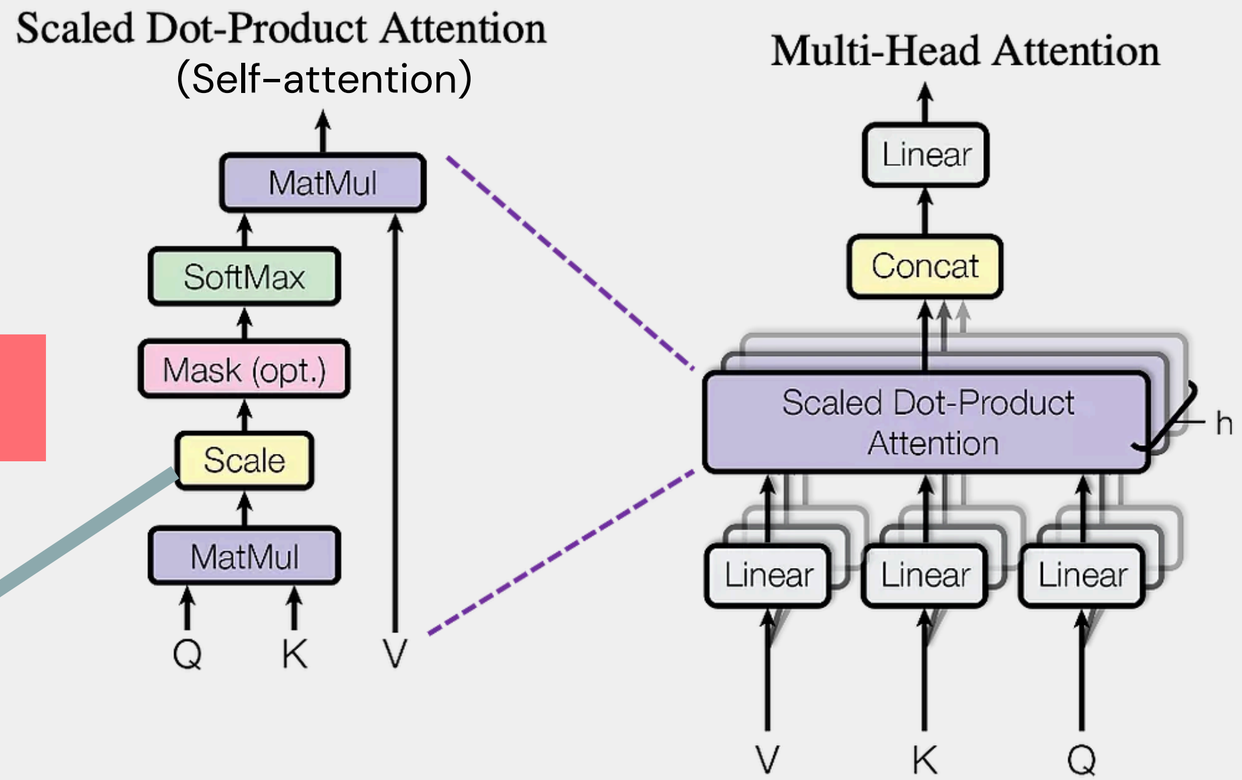
$$Similarity(A, B) = \frac{A.B}{scaling}$$

$$Similarity(A, B) = \frac{A.B^T}{scaling}$$

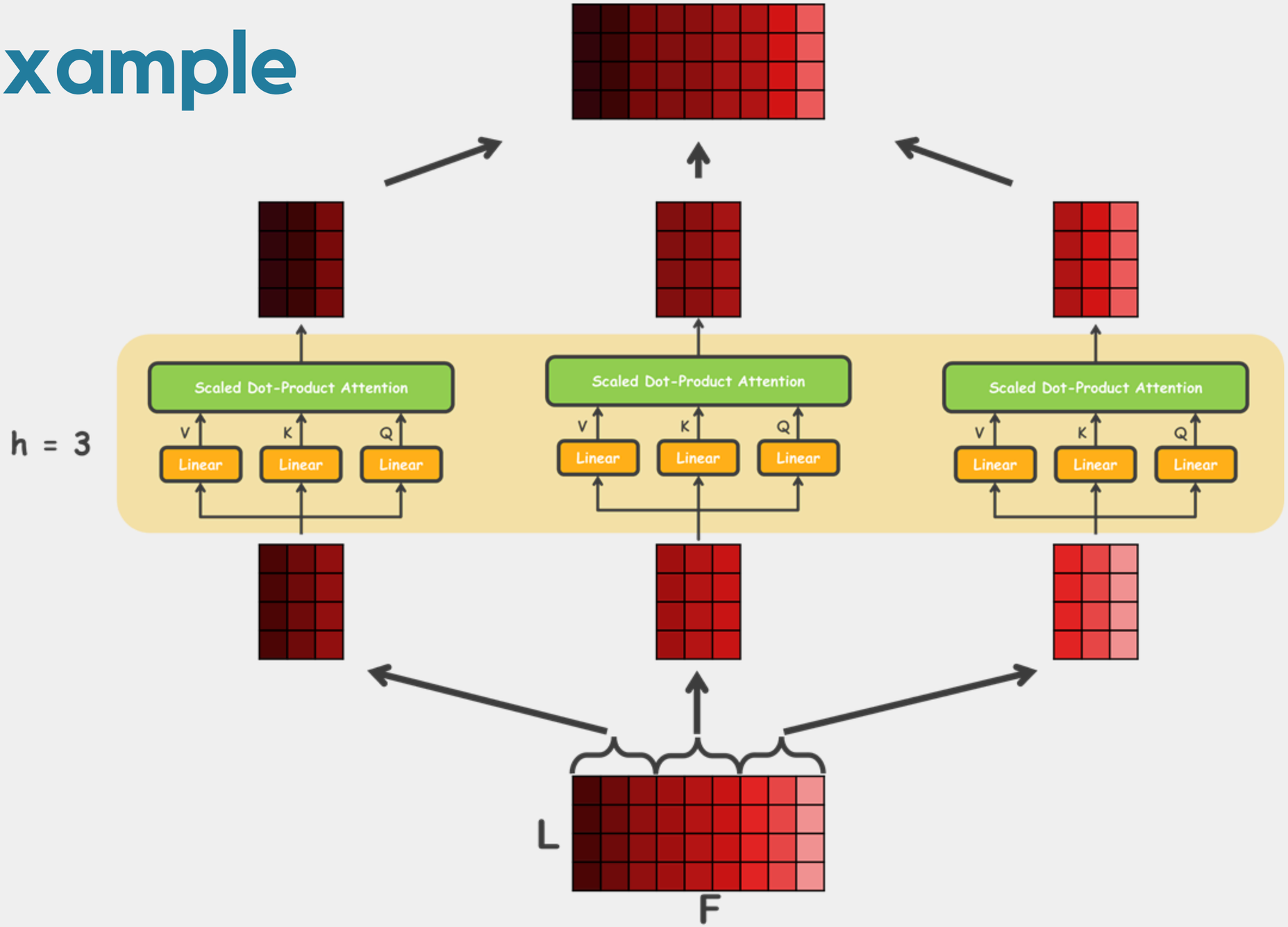
$$Similarity(Q, K) = \frac{Q.K^T}{scaling}$$

Multi-Head Attention

$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right)V$$

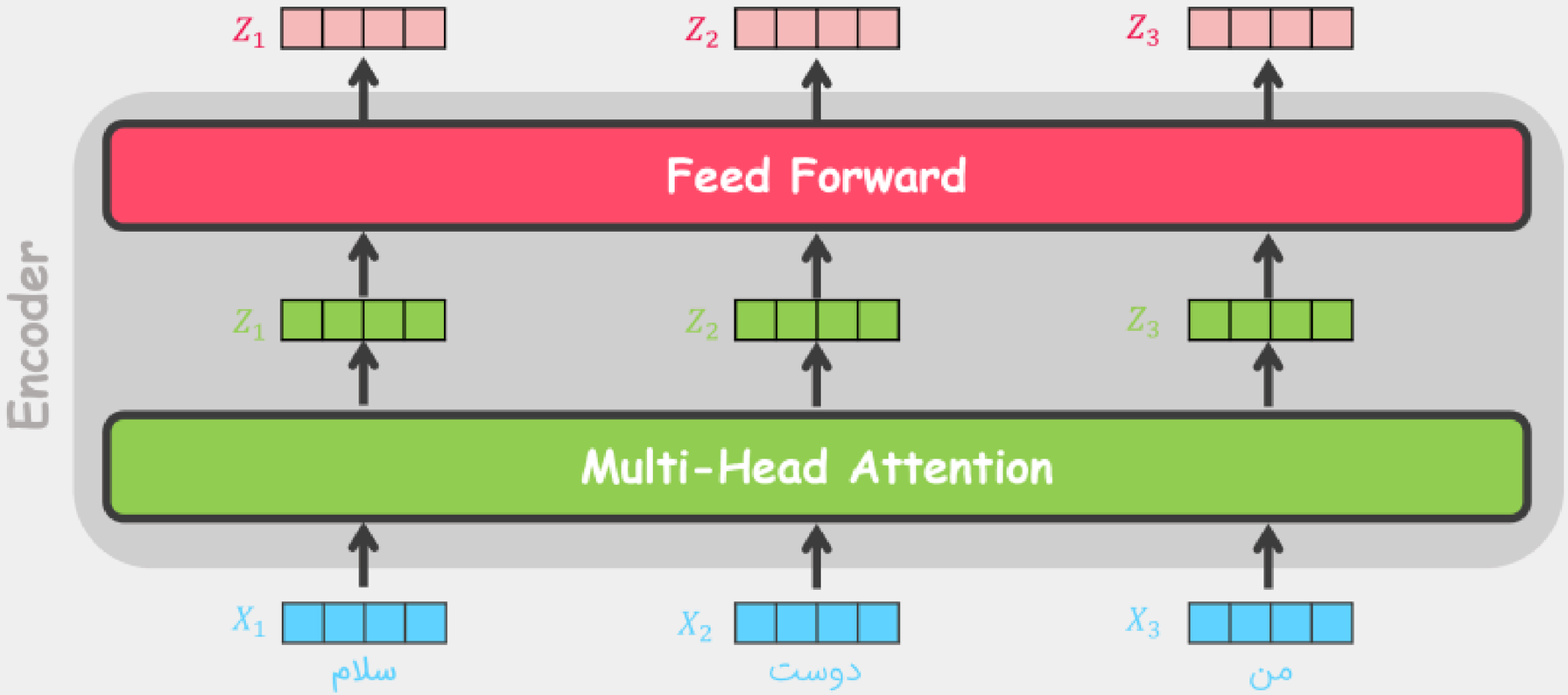


Example


h = 3
Scaled Dot-Product Attention
V
K
Q
Linear
Linear
Linear
Scaled Dot-Product Attention
V
K
Q
Linear
Linear
Linear
Scaled Dot-Product Attention
V
K
Q
Linear
Linear
Linear
L
F

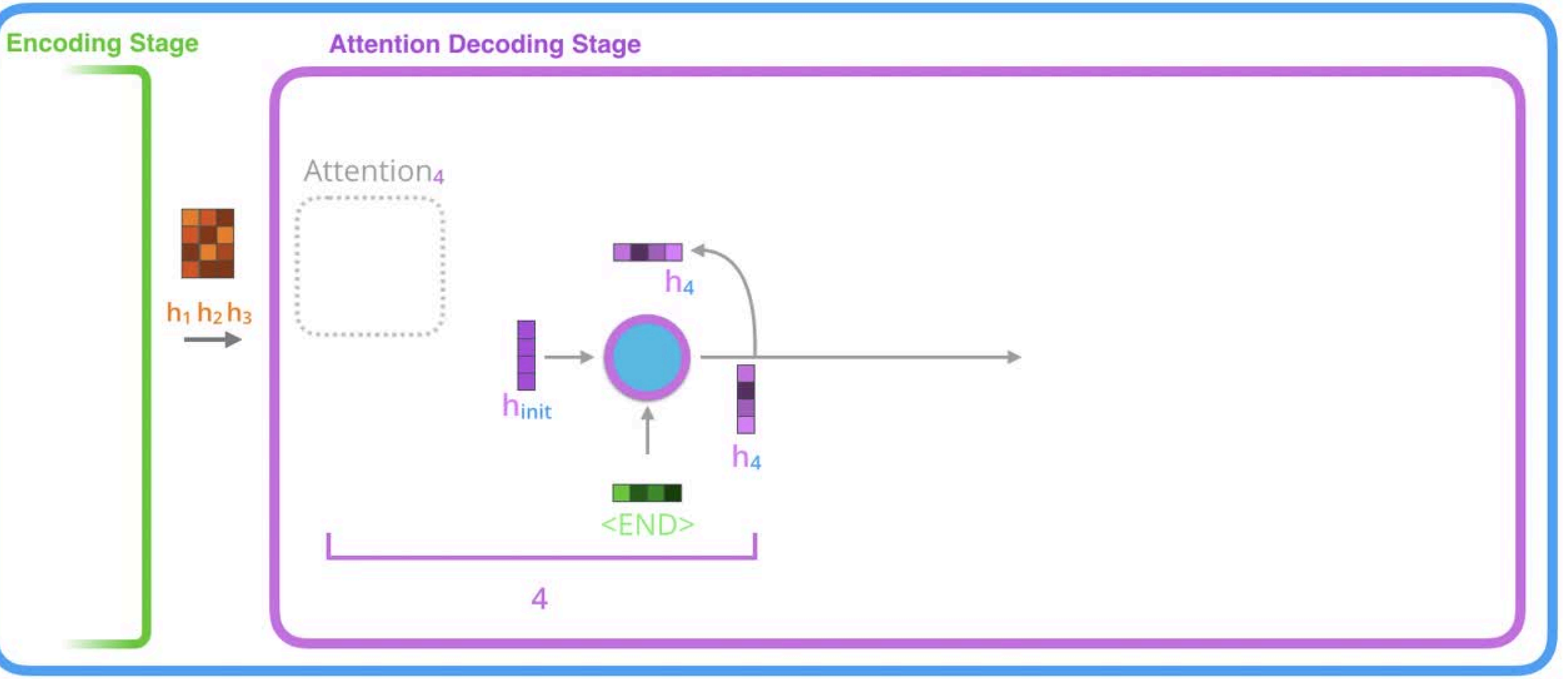
Feed Forward

پس از اتمام مراحل قبل، نتیجه برای پردازش بیشتر به یک شبکه‌ی پیش‌خور داده می‌شود که متشکل از چندین لایه با تابع فعال‌ساز ReLU می‌باشد. هم‌چنین، مراحل استفاده از اتصالات اضافی (residual connection) و نرمال‌سازی نیز مجدد تکرار می‌شوند.

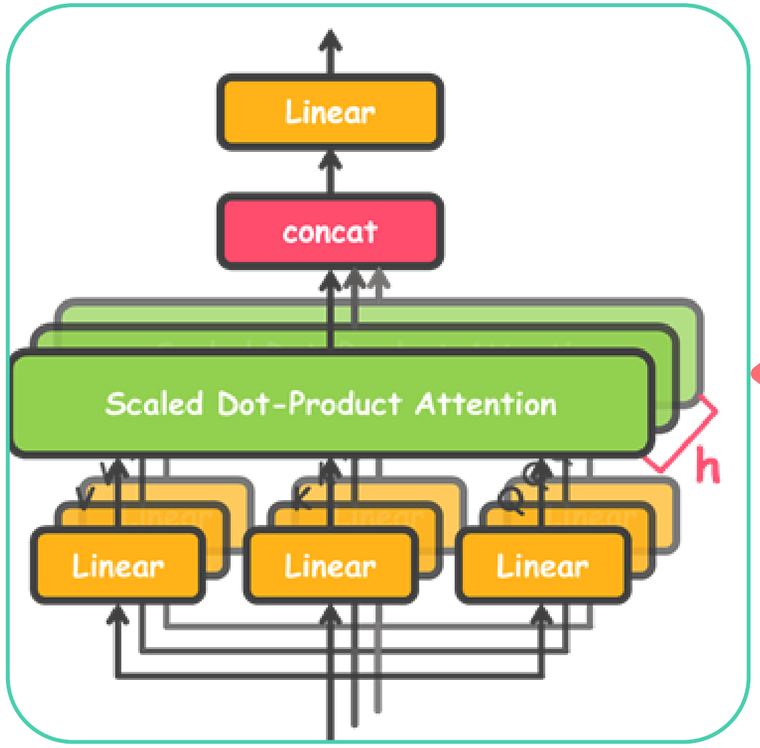


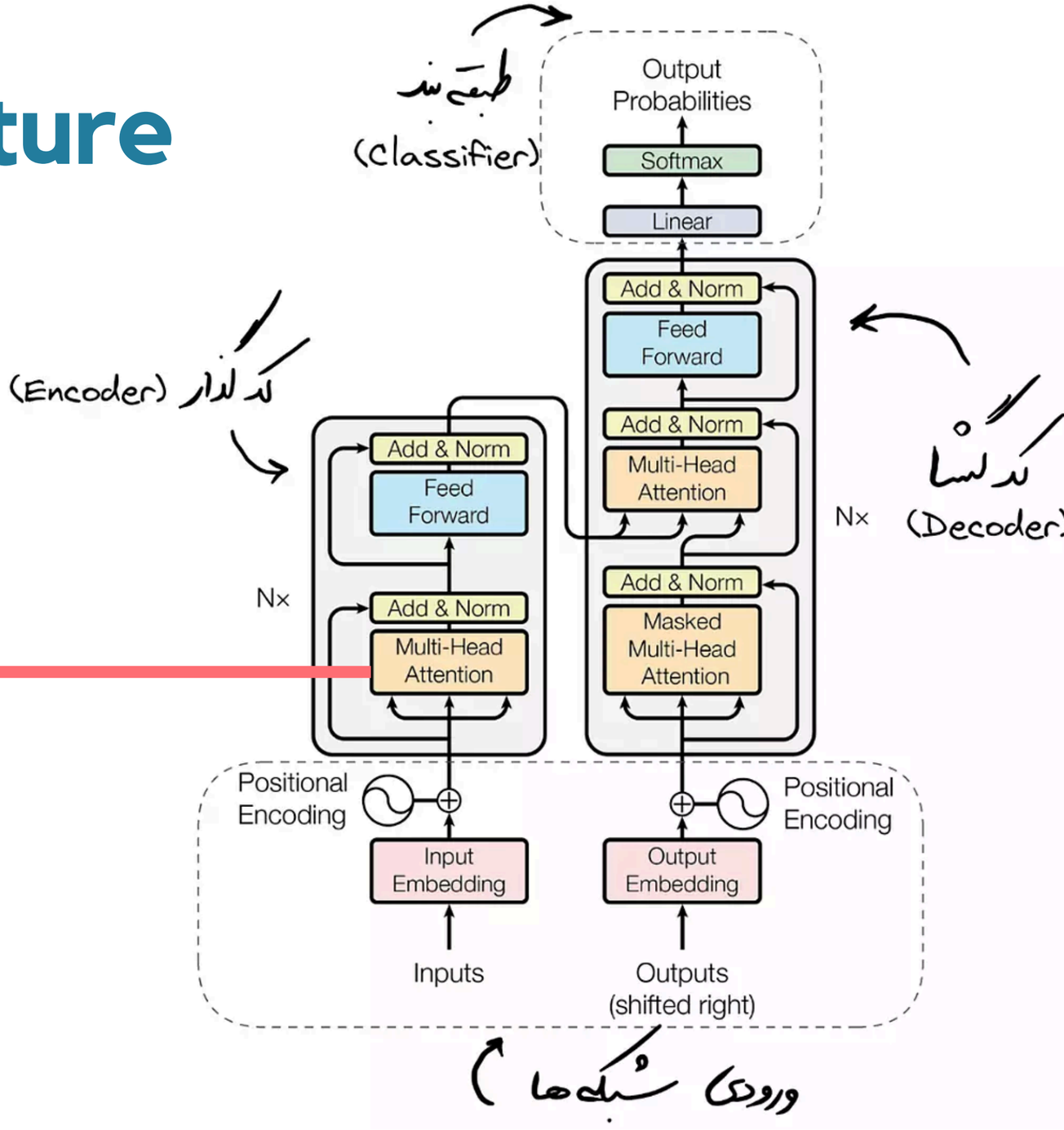
Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



Model Architecture





References

- https://medium.com/@geetkal67/attention-networks-a-simple-way-to-understand-self-attention-f5fb363c736d
- https://blog.faradars.org/implement-gradient-descent-in-python/
- https://blog.faradars.org/implement-gradient-descent-in-python/
- https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/
- https://paperswithcode.com/paper/squeezebert-what-can-computer-vision-teach
- https://biasvariance.net/articles/868ss/softmax-behaviour/

References

- https://paperswithcode.com/method/dropout
- http://jmlr.org/papers/v15/srivastava14a.html
- https://howsam.org/transformer/
- https://virgool.io/@shenasa/%D9%85%DA%A9%D8%A7%D9%86%DB%8C%D8%B2%D9%85-%D8%AA%D9%88%D8%AC%D9%87-%D9%88-%D9%85%D8%AF%D9%84-%D9%87%D8%A7%DB%8C-%D8%AA%D8%A8%D8%AF%DB%8C%D9%84-%DA%A9%D9%86%D9%86%D8%AF%D9%87-transformers-zxboffthj9xi
- https://www.youtube.com/watch?v=dichIcUZfOw

THANK YOU

mahdirahmani8@yahoo.com