ViT and Self-Supervised Models

CS280

Spring 2025

Angjoo Kanazawa

Attention is expensive

For Memory and Computation.

Prevents long context!

Flash Attention

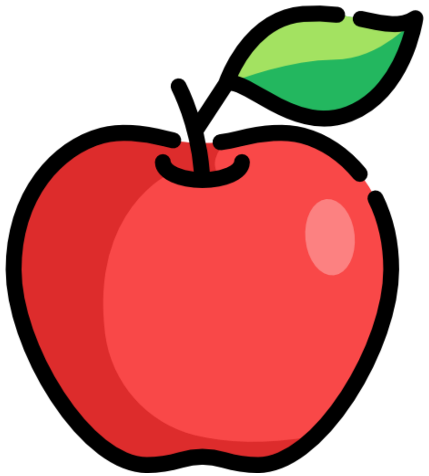
- Does not instantiate the entire N x N attention matrix A , i.e. QK^T
- Dedicated GPU kernel for efficient implementation
- Running sum for softmax…
- Memory reduced from O(N^2) to O(Nd)
- Need operation to be on Q and K independently

Dao et al. NeurIPS 2022…

Positional Encoding

I bought an apple watch

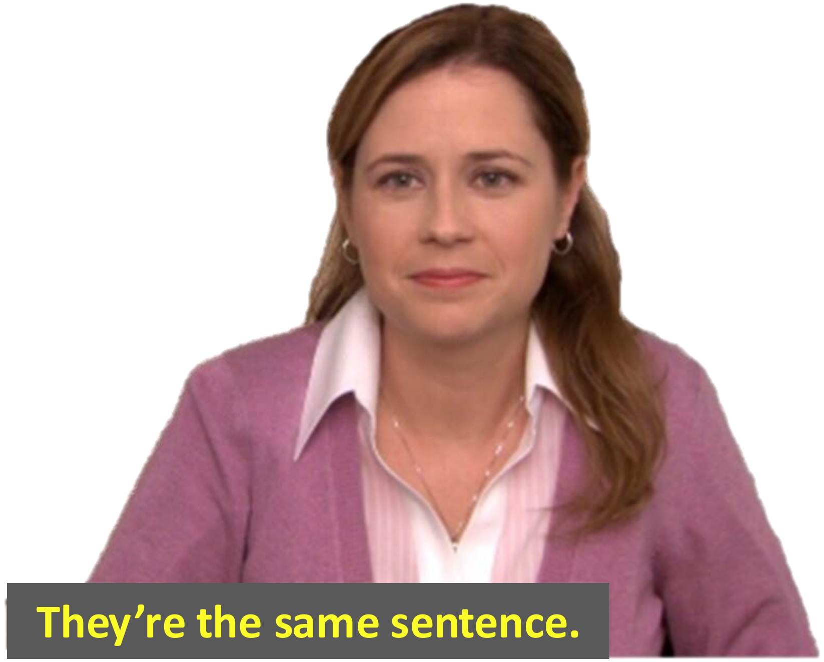
watch an apple I bought



Slide Credit: Jiabin Huang

I bought an apple watch

watch an apple I bought



Slide Credit: Jiabin Huang

Embedded Tokens	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$
Tokens	**I**	**bought**	**an**	**apple**	**watch**
Position	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$

Position k

Angular frequency

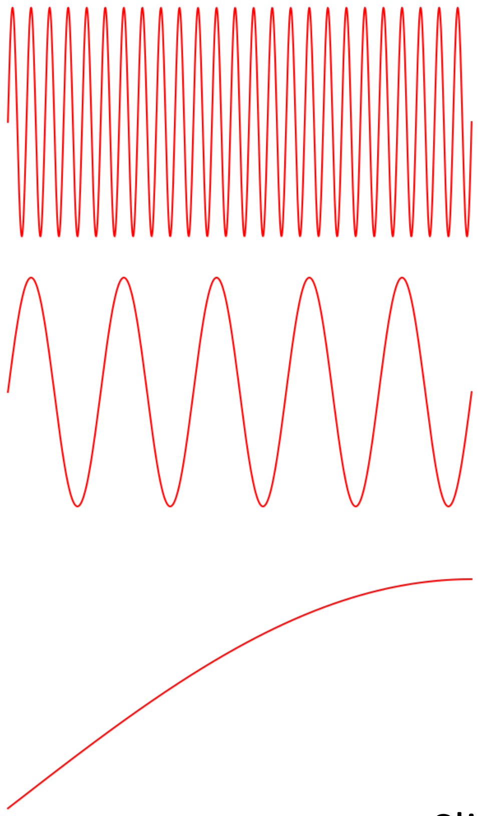
$$w_i = N^{-2i/d}$$

$$N = 100{,}000$$

$$d \quad \begin{bmatrix} \sin(w_0 m) \\ \cos(w_0 m) \\ \sin(w_1 m) \\ \cos(w_1 m) \\ \vdots \\ \vdots \\ \sin\left(w_{\frac{d}{2}-1} m\right) \\ \cos\left(w_{\frac{d}{2}-1} m\right) \end{bmatrix}$$

← Fast oscillating

← Slow oscillating



Slide Credit: Jiabin Huang

Positional Encoding

Embedded Tokens	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$
Tokens	**I**	**bought**	**an**	**apple**	**watch**
Position	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$

Position m

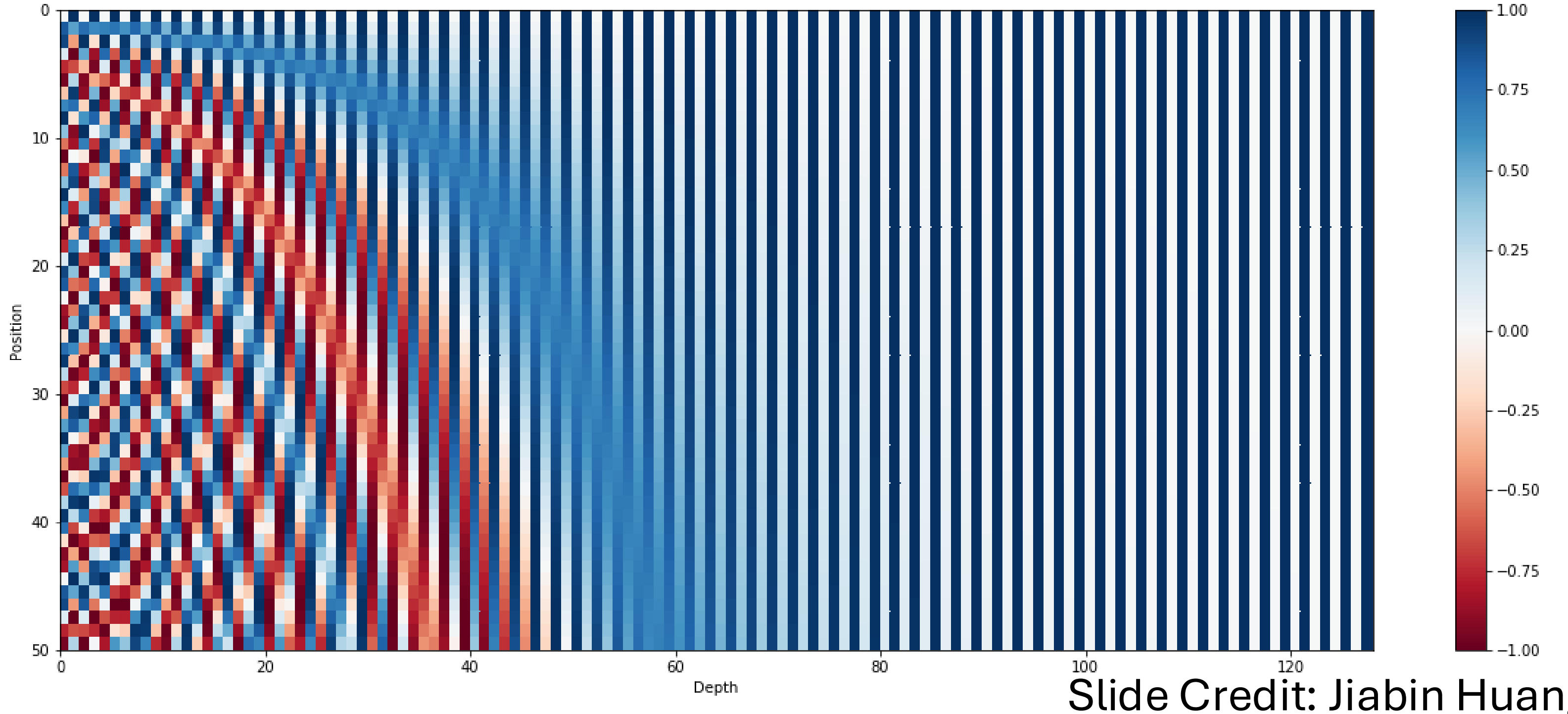
Angular frequency

$$w_i = N^{-2i/d}$$

$$N = 100{,}000$$

N is wavelength

$$d \left\{ \begin{bmatrix} \sin(w_0 m) \\ \cos(w_0 m) \\ \sin(w_1 m) \\ \cos(w_1 m) \\ \vdots \\ \vdots \\ \sin\left(w_{\frac{d}{2}-1} m\right) \\ \cos\left(w_{\frac{d}{2}-1} m\right) \end{bmatrix} \right.$$

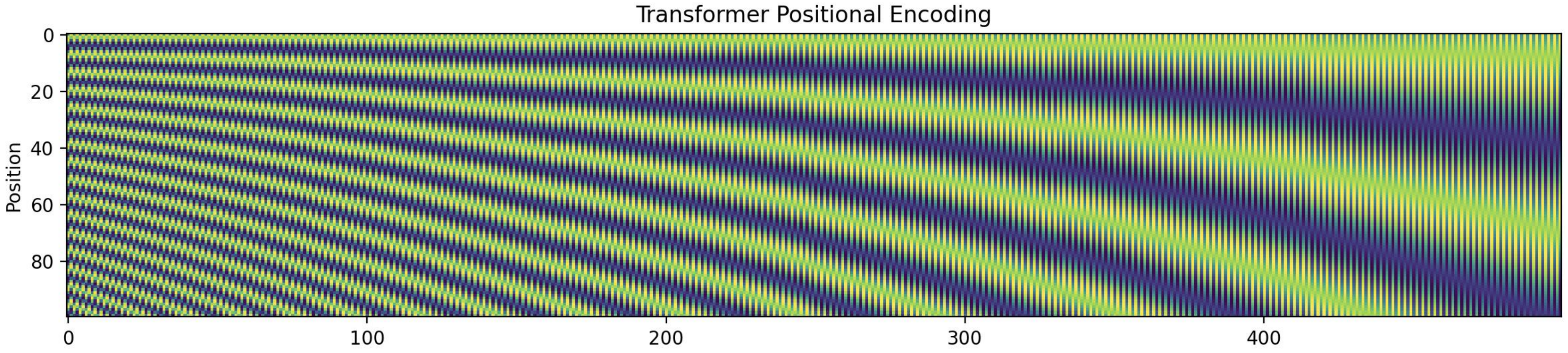


Slide Credit: Jiabin Huang

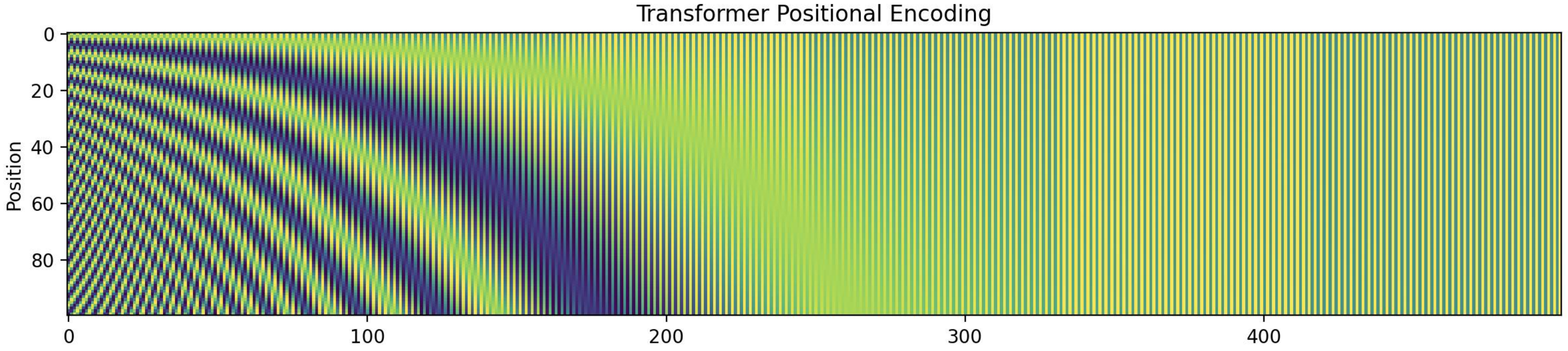
Changing N

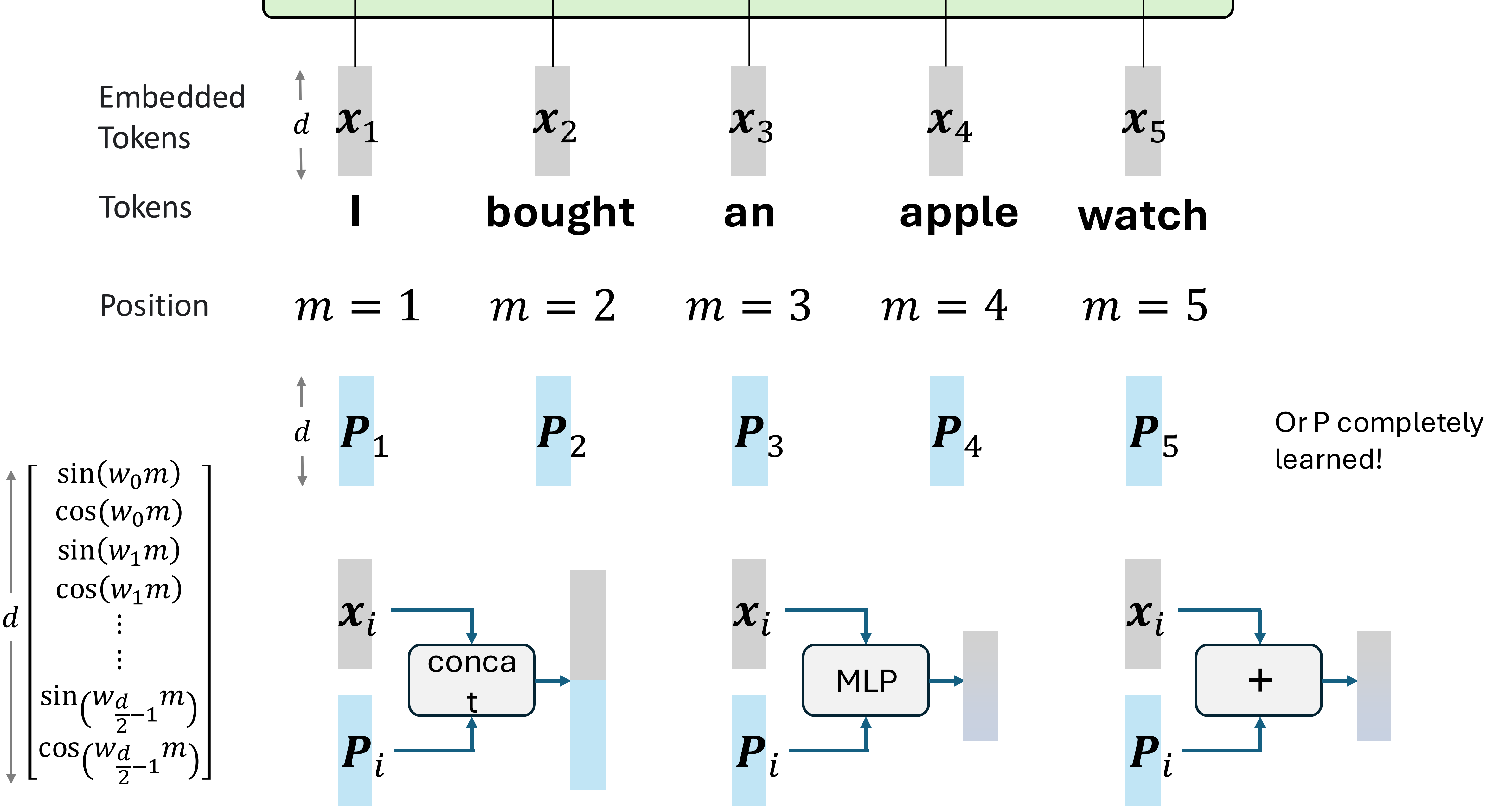
Goal: Unique positional encoding for each of your tokens.
N is the wavelength of the low-frequency band, larger it is, larger number you can represent.

N=10



What is used
N=10,000





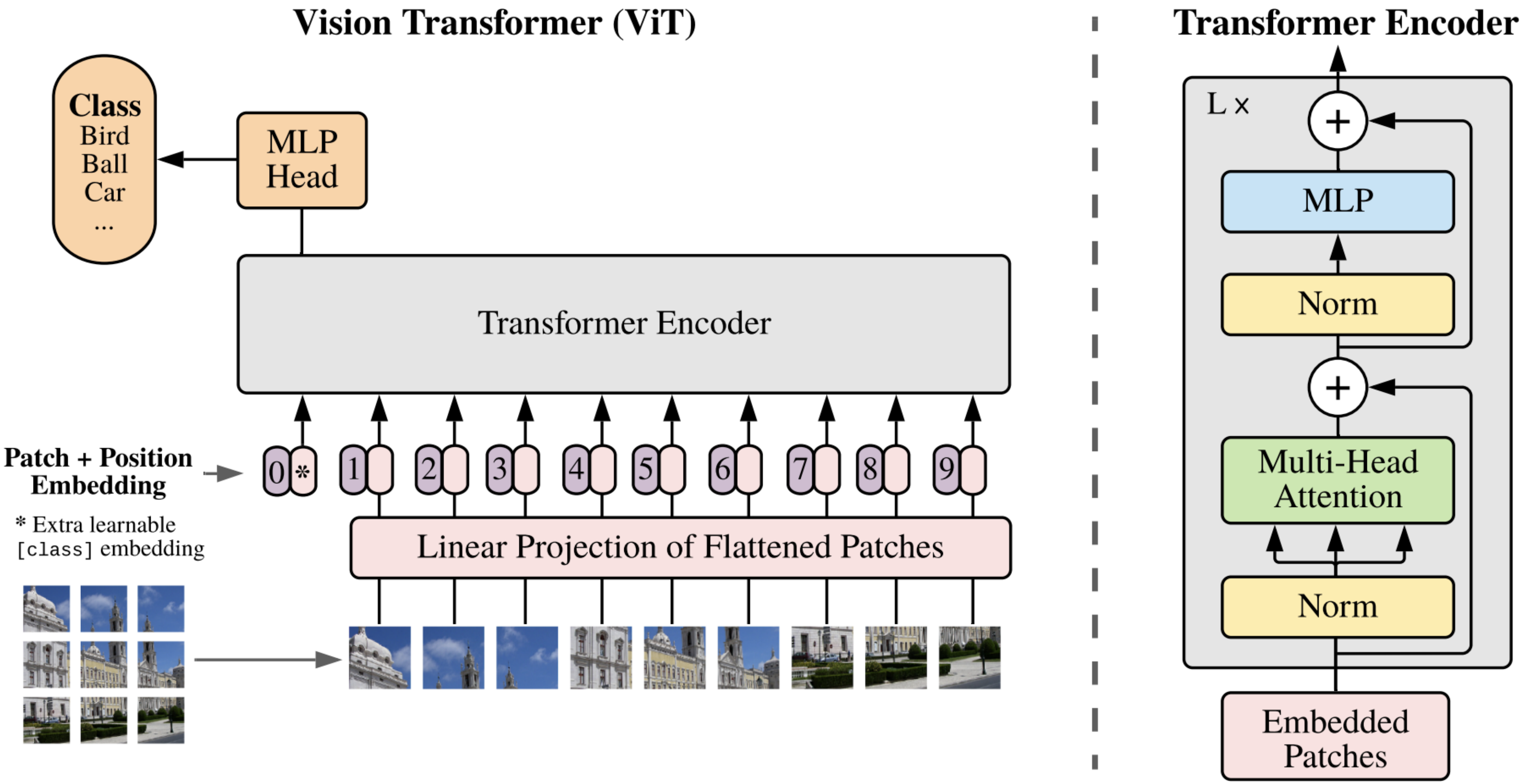
Slide Credit: Jiabin Huang

Back to pixels!

How to apply transformers to pixels?

- Naïve tokenization of a pixel for 224x224 image:
  - Over 500k tokens!
- Too much!
- Solution?

Vision Transformer



An image is worth 16x16 words, Dosovitskiy e tal. ICLR 2021

Positional Encoding

- Originally, they just flattened the patches from 1-256, treated it as a 1D signal!
- Or learned positional embedding

- Problem?
- Both solutions assume fixed size!
- You can’t change your image size!
- Need to retrain..

Relative POsitional Encoding: ROPE

Absolute Positional Encoding

Position	1	2	3	4	5	6
	I	**walk**	**my**	**dog**	**every**	**day**

Slide thanks to Jiabin Huang

Absolute Positional Encoding

Position	1	2	3	4	5	6
	I	**walk**	**my**	**dog**	**every**	**day**

Position	1	2	3	4	5	6
	every	**day**	**I**	**walk**	**my**	**dog**

Slide thanks to Jiabin Huang

RELATIVe Positional Encoding

Position

I wal my dog every day
k

Position

every day I wal my dog
k

Slide thanks to Jiabin Huang

Re-write the original posenc

$$\mathrm{PE}(p) = \begin{bmatrix} \sin(w_0 p) \\ \cos(w_0 p) \\ \sin(w_1 p) \\ \cos(w_1 p) \\ \vdots \\ \sin\left(w_{\frac{d}{2}-1} p\right) \\ \cos\left(w_{\frac{d}{2}-1} p\right) \end{bmatrix} \Bigg\updownarrow d$$

- Focus on one frequency, $\theta = w_d$
- Then this is a 2D vector

$$\mathrm{PE}(p) = [\sin(\theta p), \cos(\theta p)].$$

- This is a coordinate on an unit circle!
- You can write this fourier basis in its exponential form (Euler's formula)

$$e^{i\theta p} = \cos(\theta p) + i \sin(\theta p).$$

- Has a nice property (holds across frequencies)!:

$$\langle \mathrm{PE}(p_1), \mathrm{PE}(p_2) \rangle = e^{-i\theta p_1} e^{i\theta p_2} = e^{i\theta(p_2 - p_1)}$$

Multiplying by $e^{i\theta}$ is same as applying a 2D rotation matrix

Re-write the original posenc

- Focus on one frequency, $\theta = w_d$
- Then this is a 2D vector

$$\mathrm{PE}(p) = [\sin(\theta p), \cos(\theta p)].$$

- This is a coordinate on an unit circle!
- You can write this fourier basis in its exponential form (Euler's formula)

$$e^{i\theta p} = \cos(\theta p) + i\sin(\theta p).$$

Multiplying by $e^{i\theta}$ is same as applying a 2D rotation matrix

- Has a nice property (holds across frequencies)!:

$$\langle \mathrm{PE}(p_1), \mathrm{PE}(p_2) \rangle = e^{-i\theta p_1} e^{i\theta p_2} = e^{i\theta(p_2 - p_1)}$$

ROPE — rotate tokens for relative positional encoding

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

$$Q = W_q X, \quad K = W_k X.$$

Absolute PosEnc:

$$X = \cancel{X' +} PE(m)$$

$$QK^T = (W_q X)(X^T W_k^T).$$

$$Q = W_q(Xe^{im}), \quad K = W_k(Xe^{in}).$$

$$QK^T = W_q X e^{i(m-n)} X^T W_k^T.$$

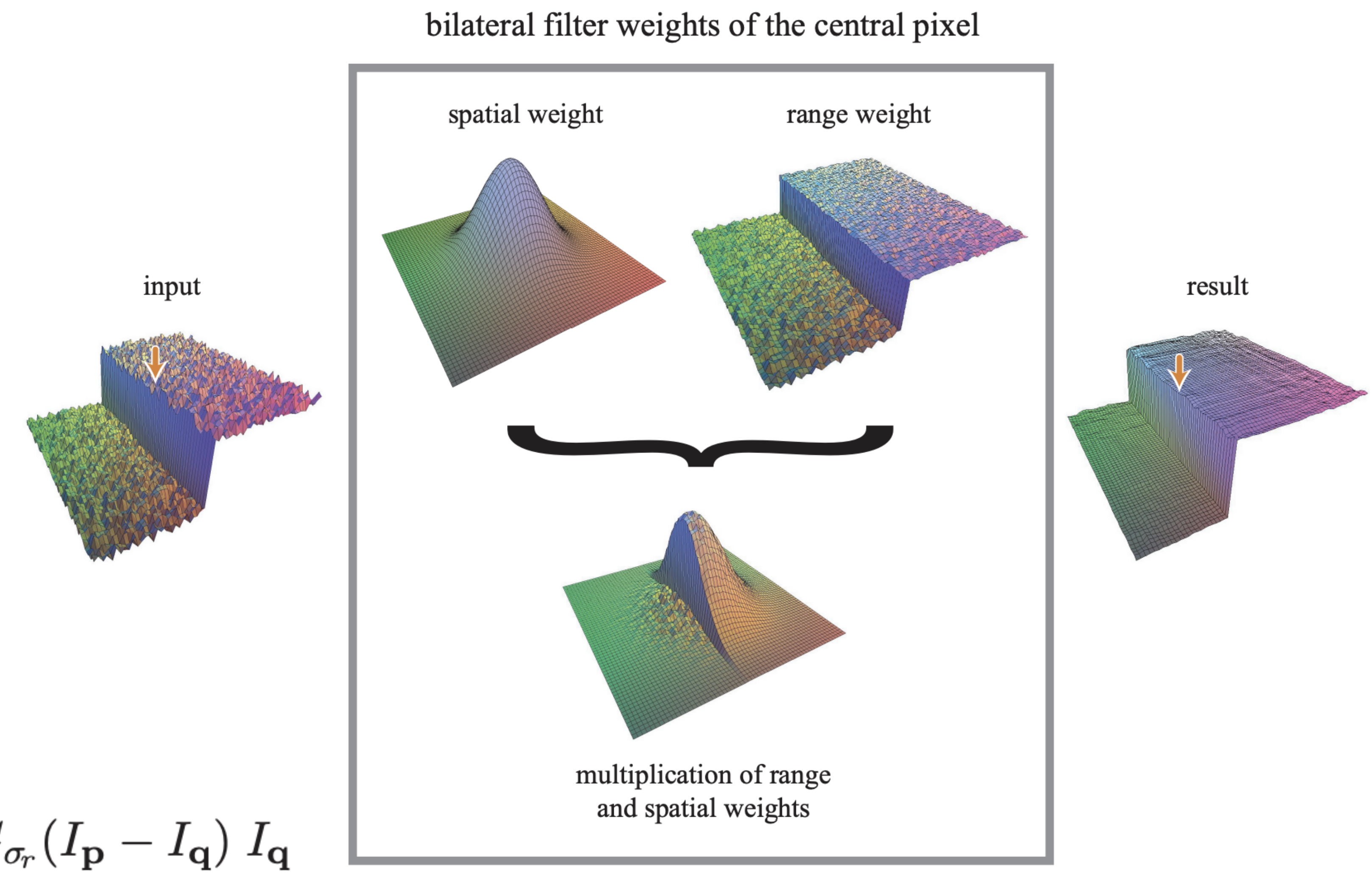
Pre-multiply the tokens with rotations

Flash Attention friendly!!

Unfortunately only works in 1D SO(2) bc of commutativitiy

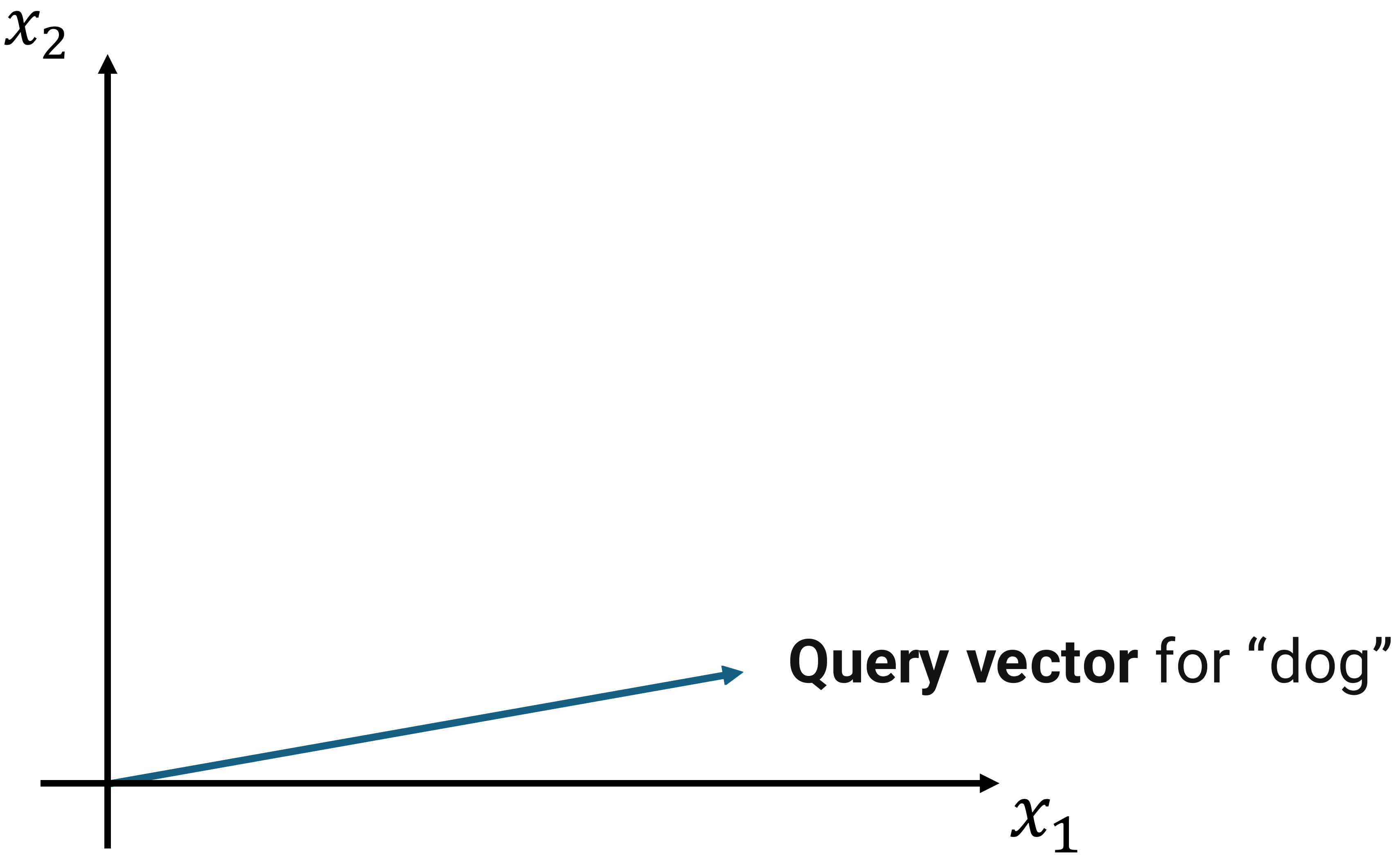
Aurich and Weule 1995
Tomasi and Manduchi 1998...

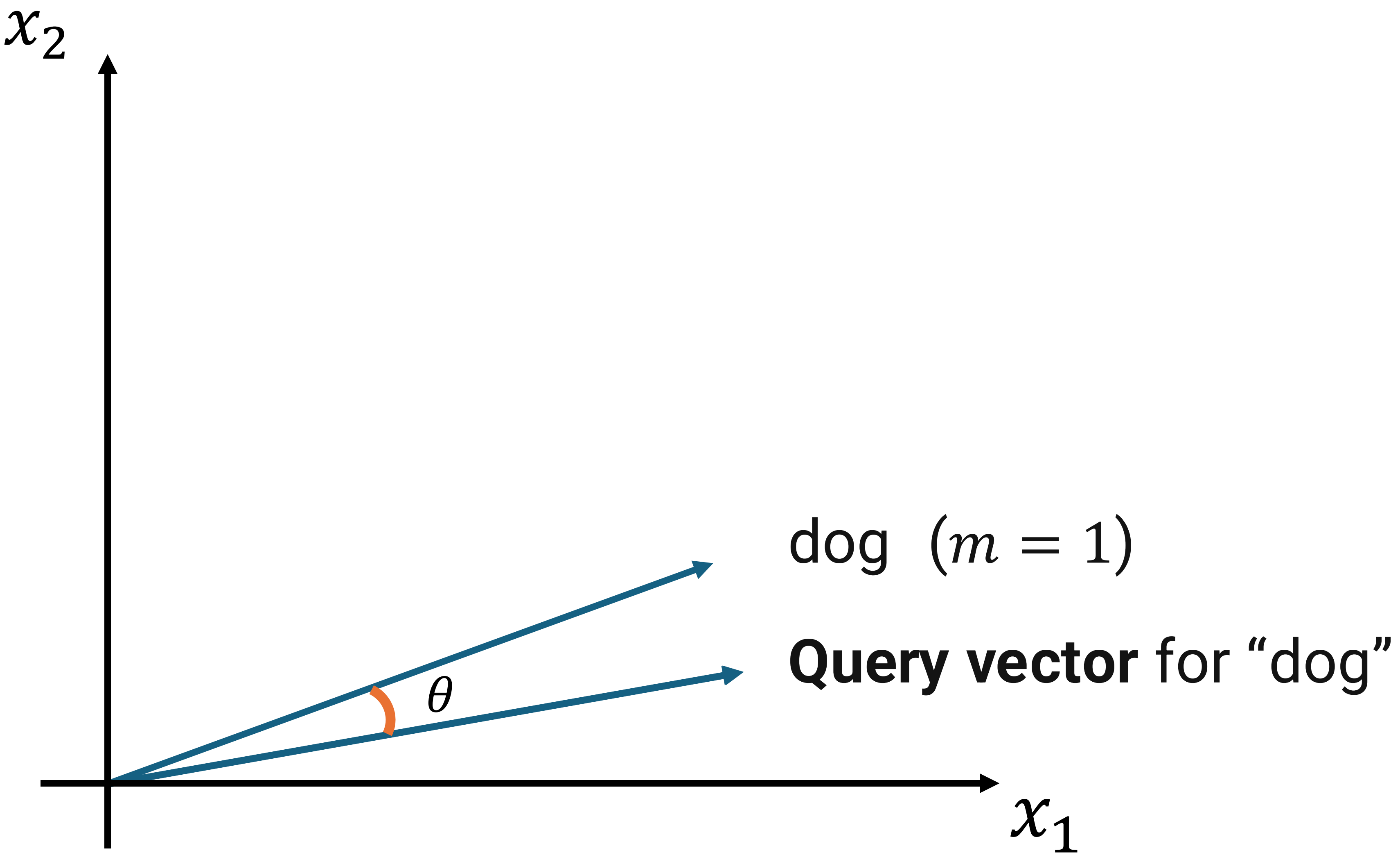
Bilateral Filter



$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} \overbrace{G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)}^{\text{Distance Similarity}} \, \underbrace{G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})}_{\text{Pixel Similarity}} \, I_{\mathbf{q}}$$

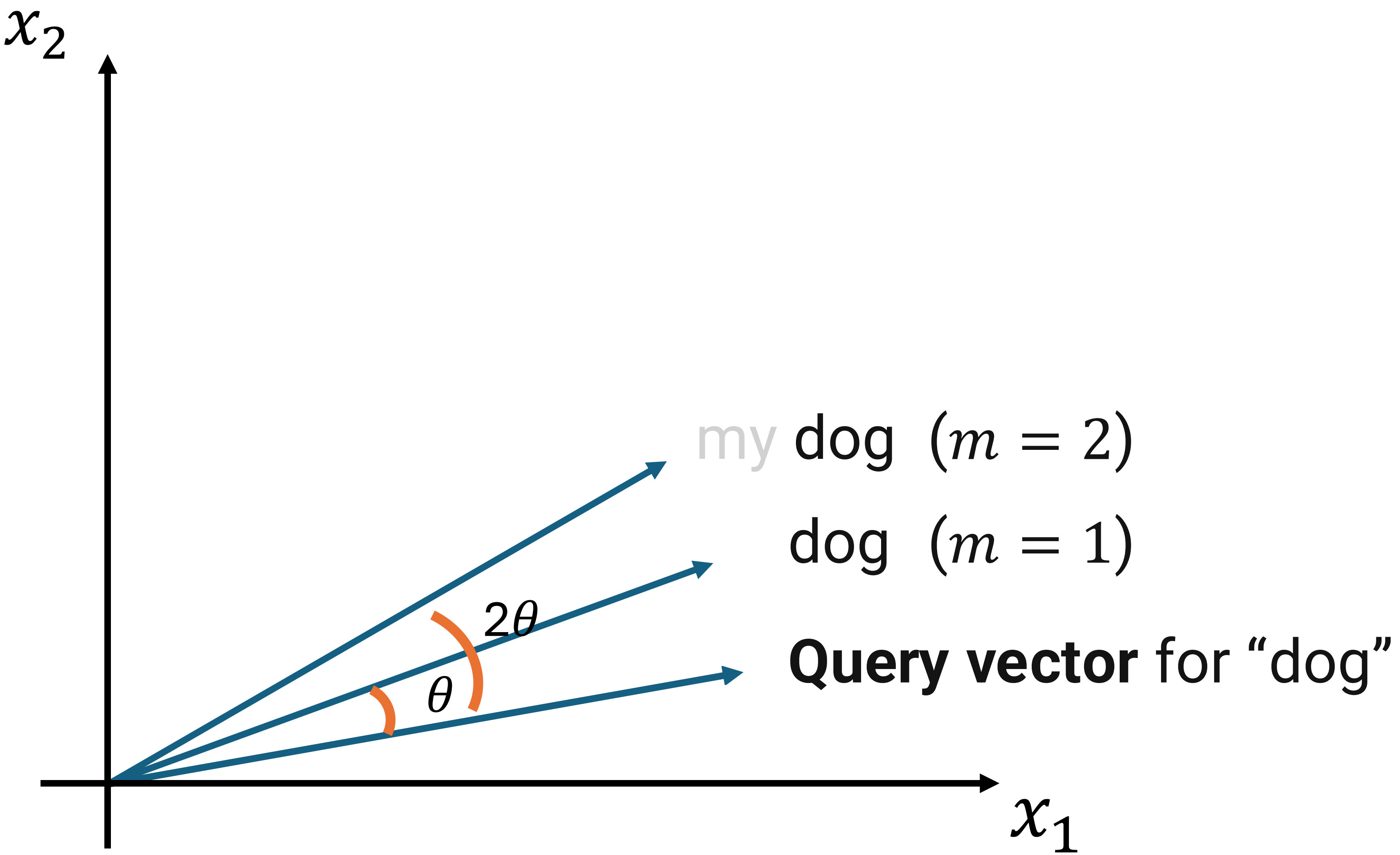
Figure from paris et al.

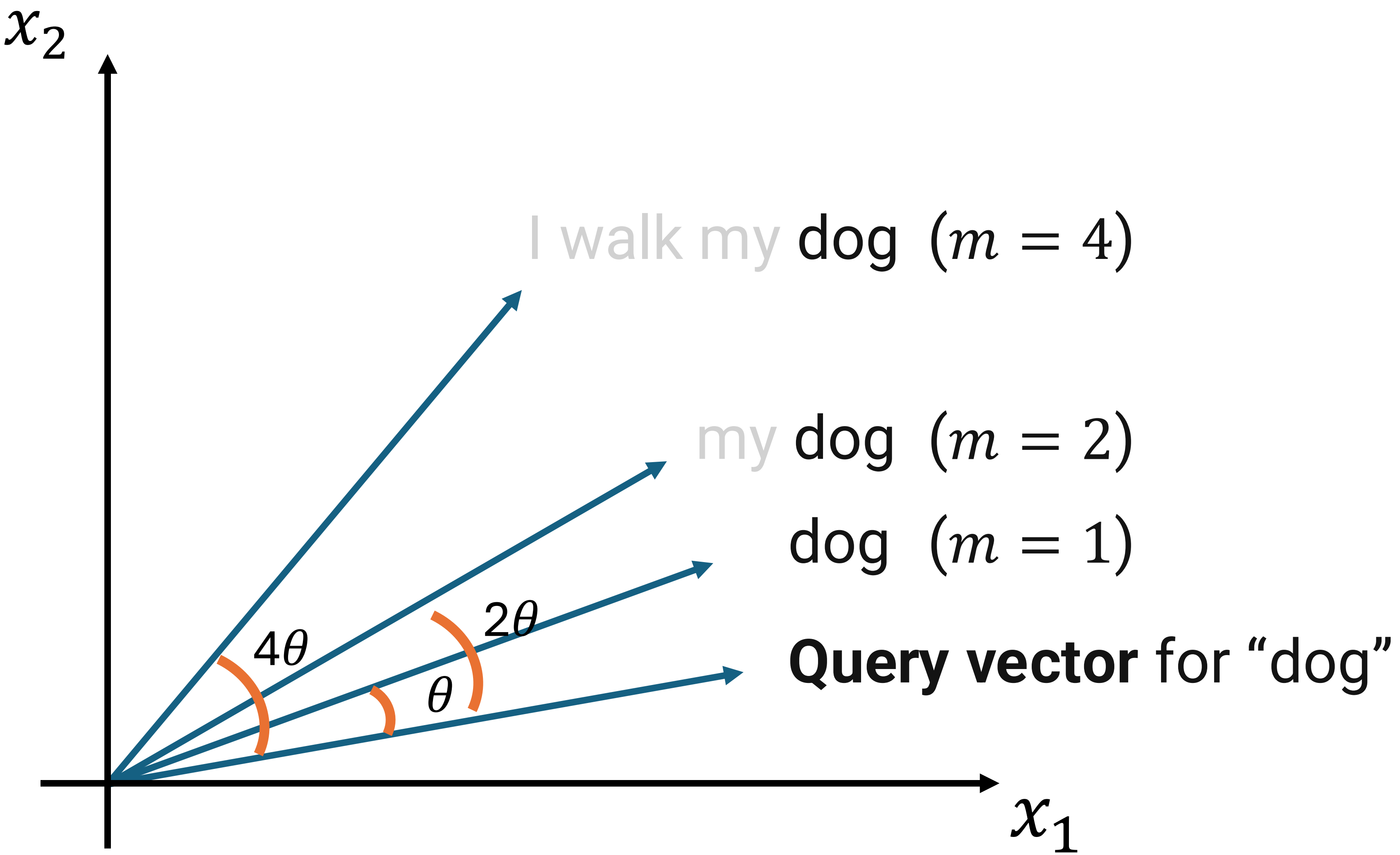

x_2
Query vector for “dog”
x_1



Slide thanks to Jiabin Huang

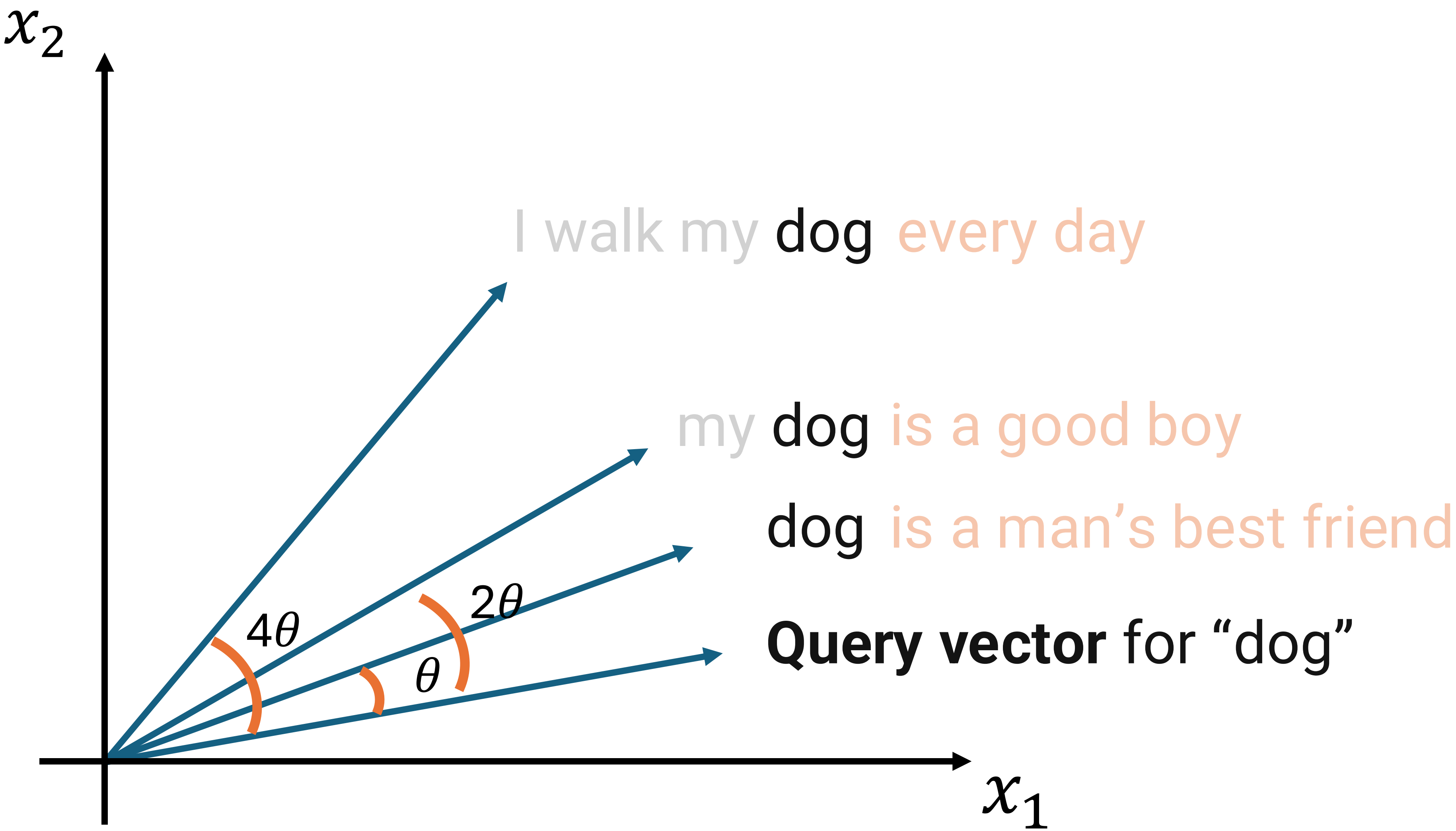
Rotary Positional Embeddings, 2021


x_2
my dog $(m = 2)$
dog $(m = 1)$
2θ
θ
Query vector for "dog"
x_1



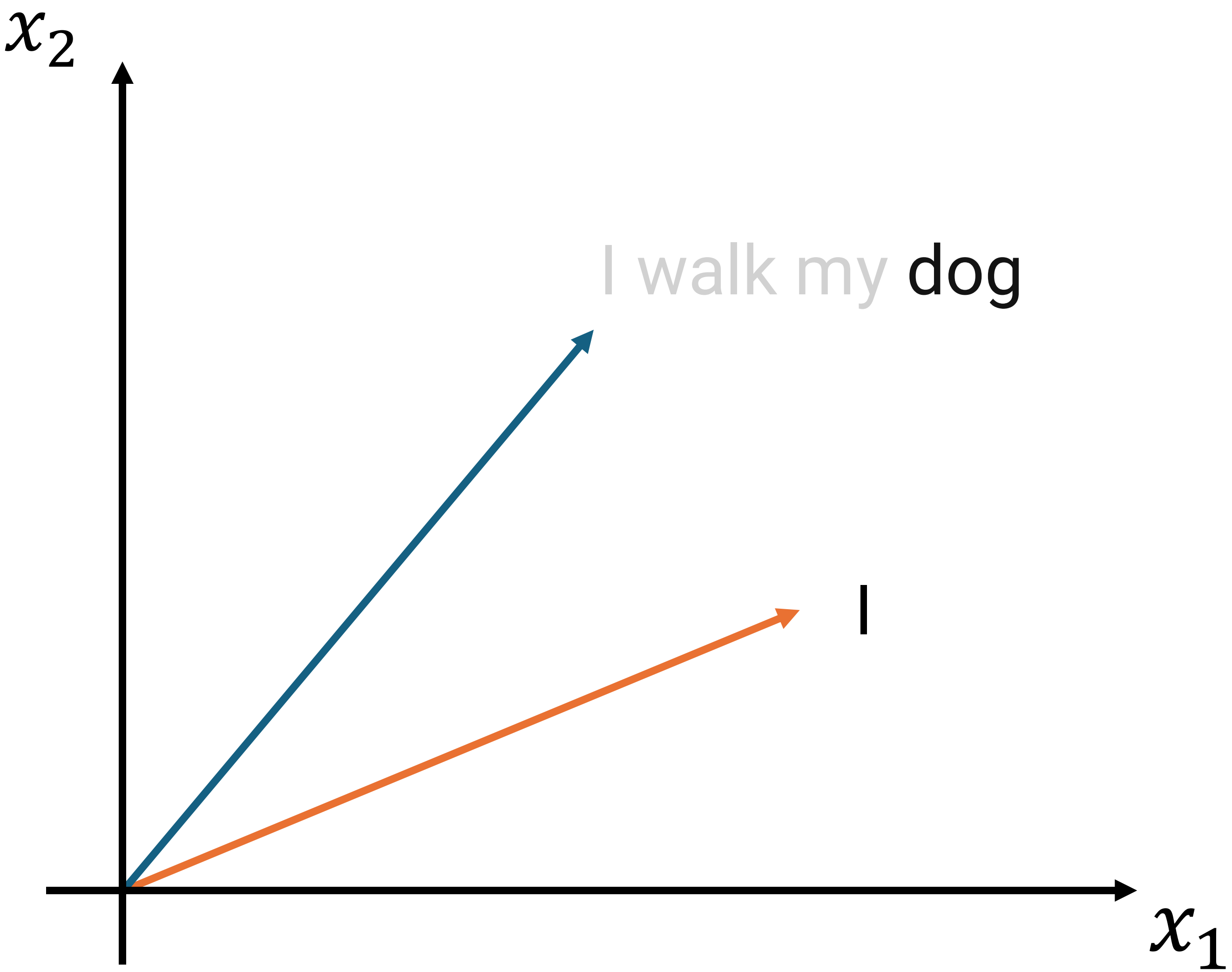
Slide thanks to Jiabin Huang

Rotary Positional Embeddings, 2021



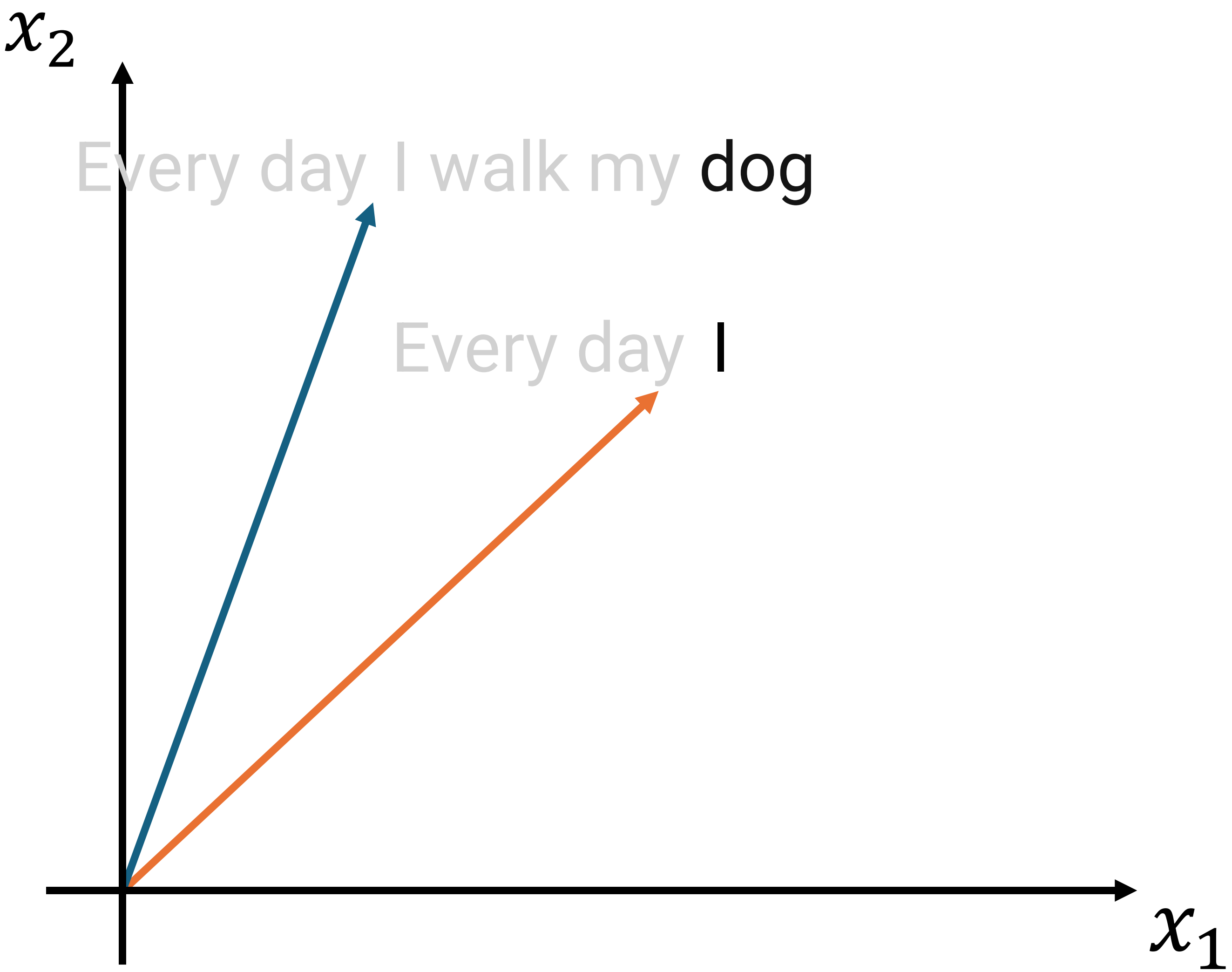
Slide thanks to Jiabin Huang

Rotary Positional Embeddings, 2021



Slide thanks to Jiabin Huang

Rotary Positional Embeddings, 2021



Slide thanks to Jiabin Huang

Rotary Positional Embeddings, 2021

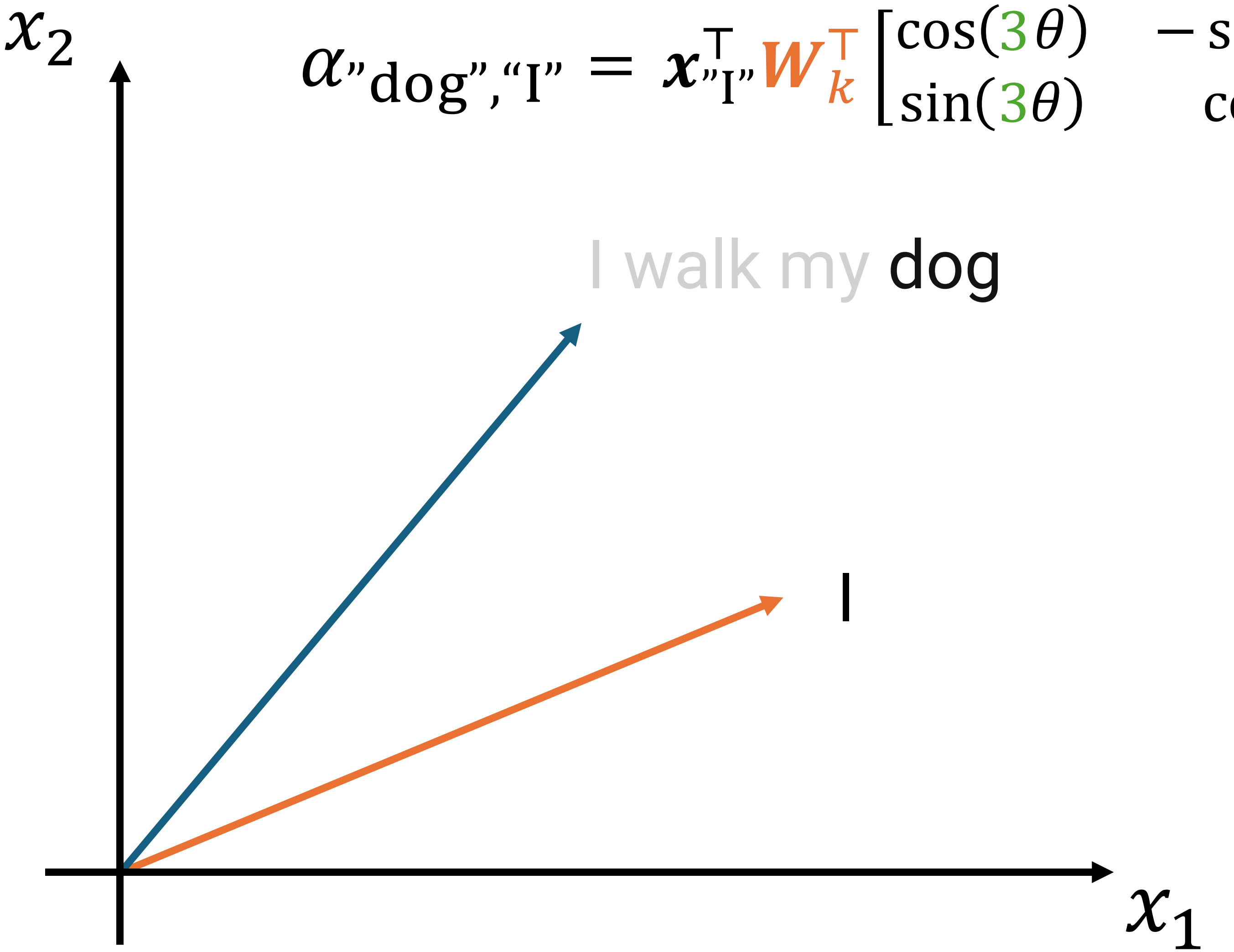
$$q_{\text{"dog"}} = \begin{bmatrix} \cos(4\theta) & -\sin(4\theta) \\ \sin(4\theta) & \cos(4\theta) \end{bmatrix} \boldsymbol{W}_q \boldsymbol{x}_{\text{"dog"}}$$

Query vector for “dog”

$$k_{\text{"I"}} = \begin{bmatrix} \cos(1\theta) & -\sin(1\theta) \\ \sin(1\theta) & \cos(1\theta) \end{bmatrix} \boldsymbol{W}_k \boldsymbol{x}_{\text{"I"}}$$

Key vector for “I”

$$\alpha_{\text{"dog","I"}} = \boldsymbol{x}_{\text{"I"}}^{\top} \boldsymbol{W}_k^{\top} \begin{bmatrix} \cos(3\theta) & -\sin(3\theta) \\ \sin(3\theta) & \cos(3\theta) \end{bmatrix} \boldsymbol{W}_q \boldsymbol{x}_{\text{"dog"}}$$



Slide thanks to Jiabin Huang

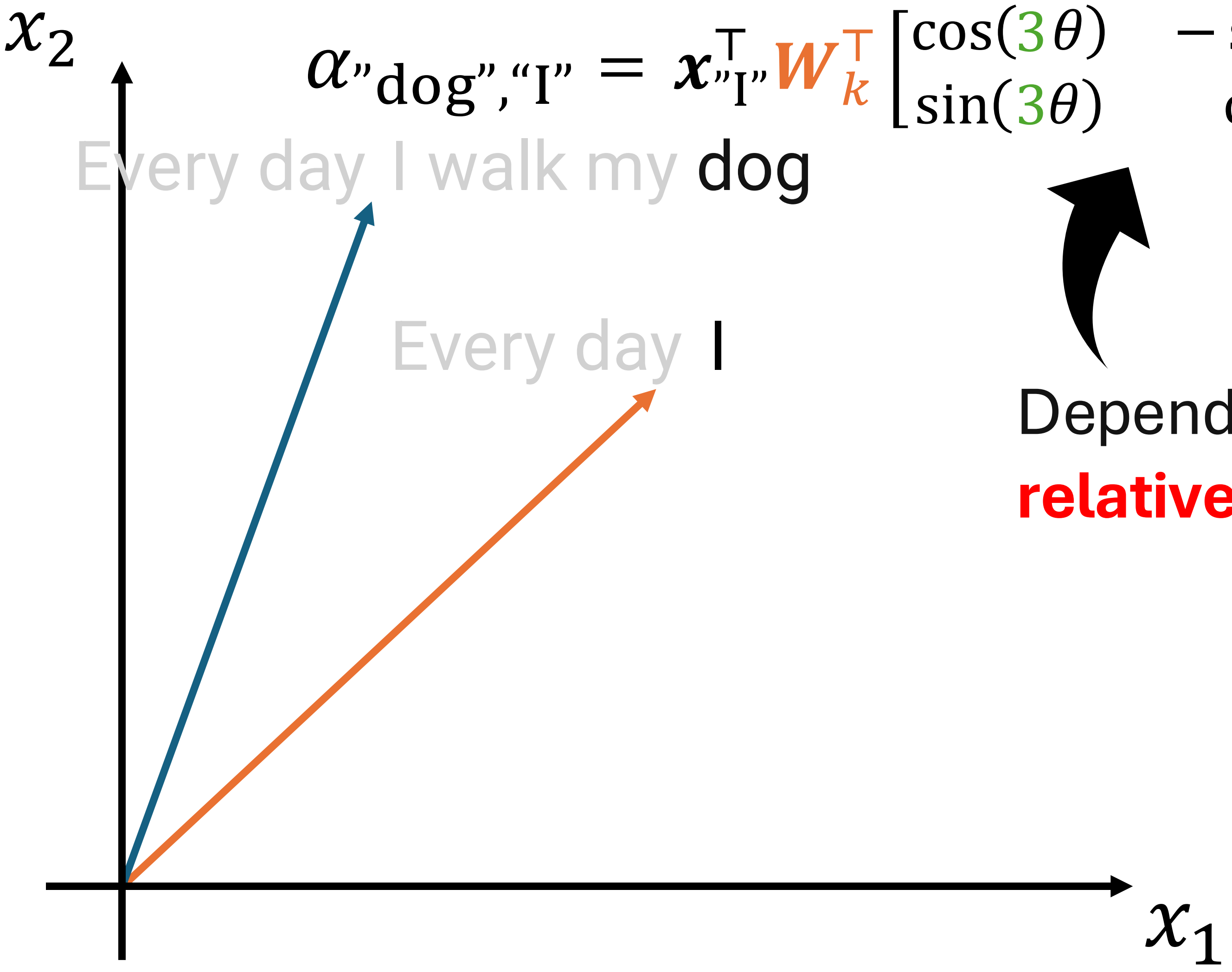
$$q_{\text{"dog"}} = \begin{bmatrix}\cos(6\theta) & -\sin(6\theta)\\ \sin(6\theta) & \cos(6\theta)\end{bmatrix} \boldsymbol{W}_q \boldsymbol{x}_{\text{"dog"}}$$

Query vector for “dog”

$$k_{\text{"I"}} = \begin{bmatrix}\cos(3\theta) & -\sin(3\theta)\\ \sin(3\theta) & \cos(3\theta)\end{bmatrix} \boldsymbol{W}_k \boldsymbol{x}_{\text{"I"}}$$

Key vector for “I”

$$\alpha_{\text{"dog","I"}} = \boldsymbol{x}_{\text{"I"}}^{\top} \boldsymbol{W}_k^{\top} \begin{bmatrix}\cos(3\theta) & -\sin(3\theta)\\ \sin(3\theta) & \cos(3\theta)\end{bmatrix} \boldsymbol{W}_q \boldsymbol{x}_{\text{"dog"}}$$



Depends only on the **relative positions**

Slide thanks to Jiabin Huang

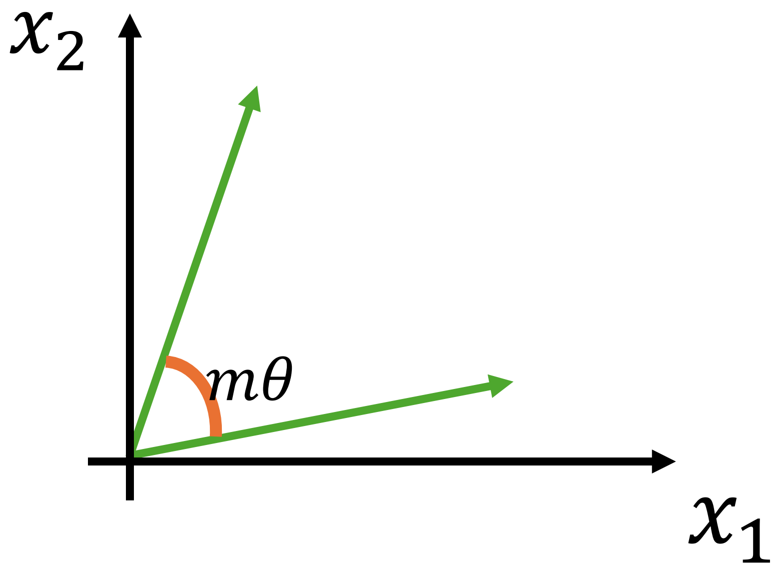
$$R_{\Theta}^{m} \boldsymbol{W}_{q} \boldsymbol{x}_{m}$$

query vector for the token at position m

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$

$$R_{\Theta}^{n} \boldsymbol{W}_{k} \boldsymbol{x}_{n}$$

key vector for the token at position n



Slide thanks to Jiabin Huang

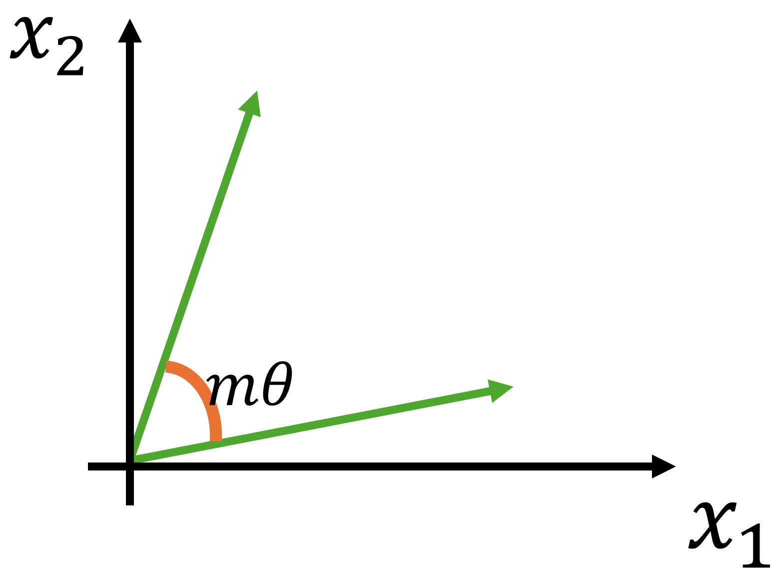
$$R_\Theta^m \boldsymbol{W}_q \boldsymbol{x}_m$$

query vector for the token at position m

$$R_\Theta^m = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$

$$R_\Theta^n \boldsymbol{W}_k \boldsymbol{x}_n$$

key vector for the token at position n



$$\alpha_{m,n} = \boldsymbol{x}_n^\top \boldsymbol{W}_k^\top (R_\Theta^n)^\top R_\Theta^m \boldsymbol{W}_q \boldsymbol{x}_m$$

Slide thanks to Jiabin Huang

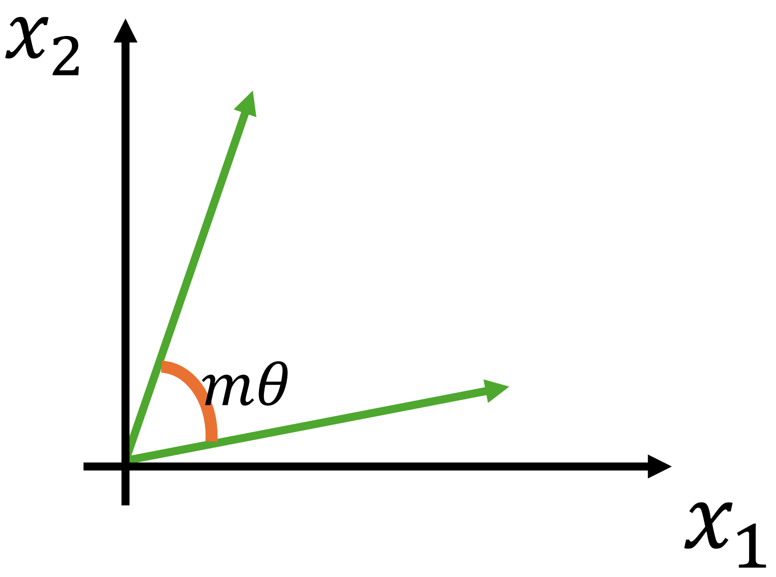
$$R_\Theta^m \boldsymbol{W}_q \boldsymbol{x}_m$$

query vector for the token at position m

$$R_\Theta^m = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$

$$R_\Theta^n \boldsymbol{W}_k \boldsymbol{x}_n$$

key vector for the token at position n



$$\alpha_{m,n} = \boldsymbol{x}_n^\top \boldsymbol{W}_k^\top (R_\Theta^n)^\top R_\Theta^m \boldsymbol{W}_q \boldsymbol{x}_m$$

Slide thanks to Jiabin Huang

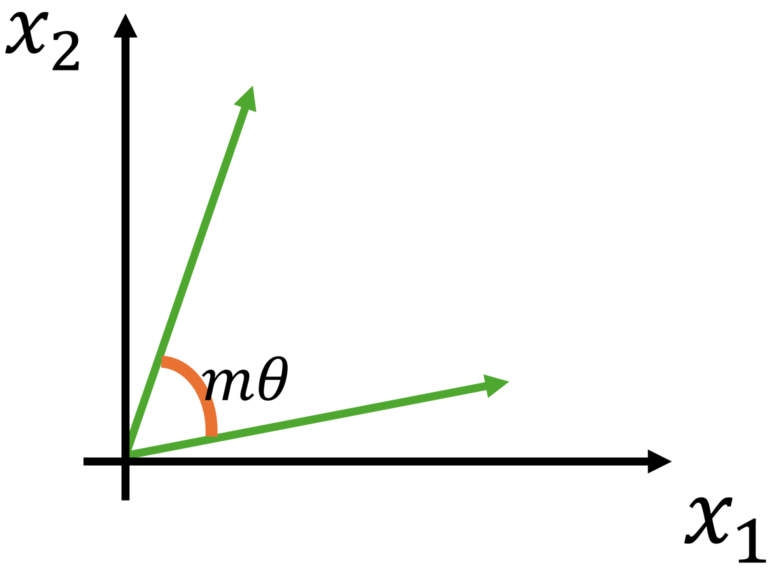
$$R_{\Theta}^{m}\,\boldsymbol{W}_q\,\boldsymbol{x}_m$$

query vector for the token at position m

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$

$$R_{\Theta}^{n}\,\boldsymbol{W}_k\,\boldsymbol{x}_n$$

key vector for the token at position n



$$\alpha_{m,n} = \boldsymbol{x}_n^{\top}\,\boldsymbol{W}_k^{\top}(R_{\Theta}^{n})^{\top}R_{\Theta}^{m}\,\boldsymbol{W}_q\,\boldsymbol{x}_m$$

Slide thanks to Jiabin Huang

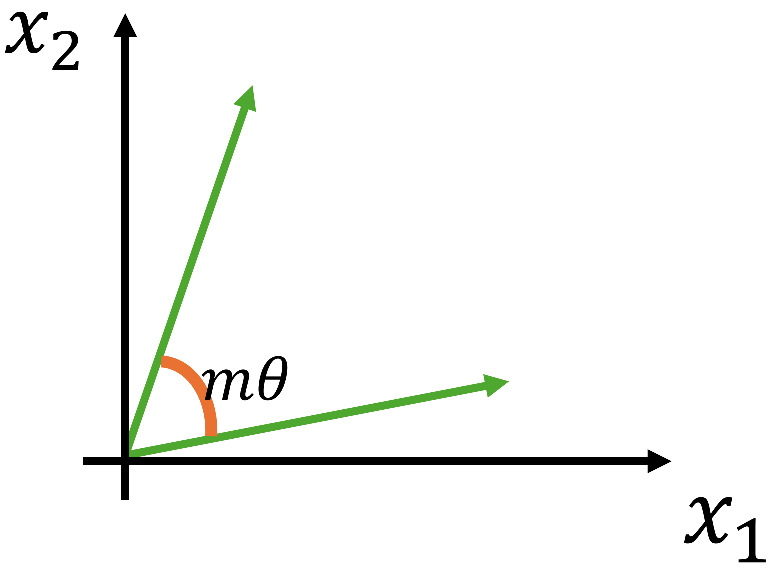
$$R_{\Theta}^{m}\, \boldsymbol{W}_q \boldsymbol{x}_m$$

query vector for the token at position m

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$

$$R_{\Theta}^{n}\, \boldsymbol{W}_k \boldsymbol{x}_n$$

key vector for the token at position n



$$\alpha_{m,n} = \boldsymbol{x}_n^{\top} \boldsymbol{W}_k^{\top} R_{\Theta}^{-n} R_{\Theta}^{m} \boldsymbol{W}_q \boldsymbol{x}_m$$

Slide thanks to Jiabin Huang

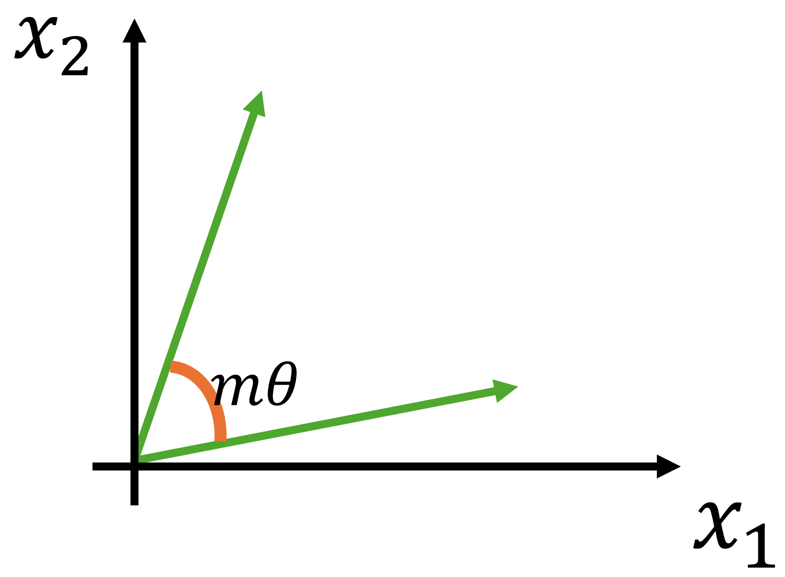
$$R_{\Theta}^{m}\, \boldsymbol{W}_q\, \boldsymbol{x}_m$$

query vector for the token at position m

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$

$$R_{\Theta}^{n}\, \boldsymbol{W}_k\, \boldsymbol{x}_n$$

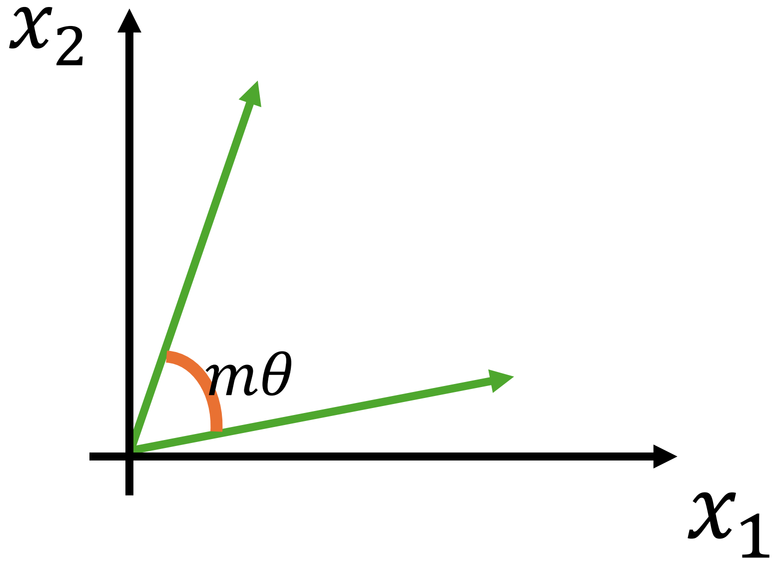
key vector for the token at position n

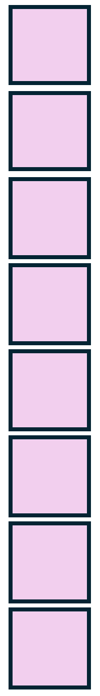


$$\alpha_{m,n} = \boldsymbol{x}_n^{\top}\, \boldsymbol{W}_k^{\top}\, R_{\Theta}^{m-n}\, \boldsymbol{W}_q\, \boldsymbol{x}_m$$

Slide thanks to Jiabin Huang

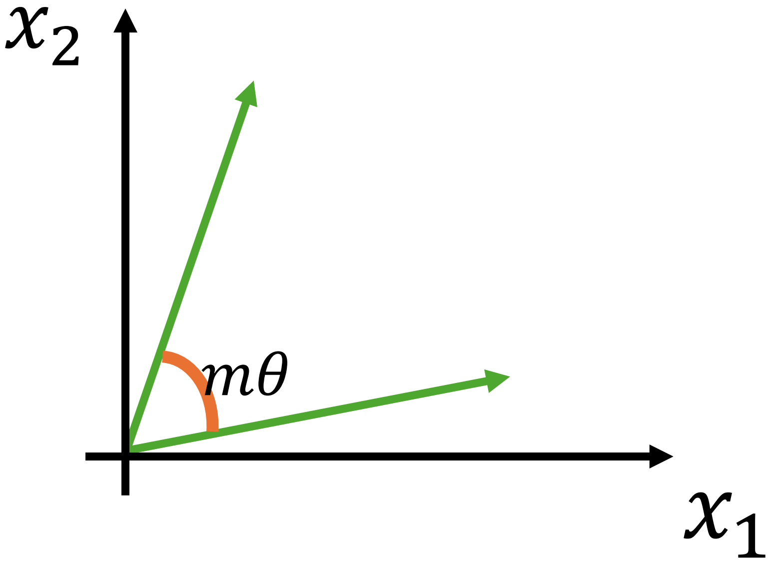
$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$





Slide thanks to Jiabin Huang

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$



$$\begin{matrix}\square\\\square\end{matrix} = \begin{bmatrix} \cos(m\theta_1) & -\sin(m\theta_1) \\ \sin(m\theta_1) & \cos(m\theta_1) \end{bmatrix} \begin{matrix}\square\\\square\end{matrix}$$

$$\begin{matrix}\square\\\square\end{matrix} = \begin{bmatrix} \cos(m\theta_2) & -\sin(m\theta_2) \\ \sin(m\theta_2) & \cos(m\theta_2) \end{bmatrix} \begin{matrix}\square\\\square\end{matrix}$$

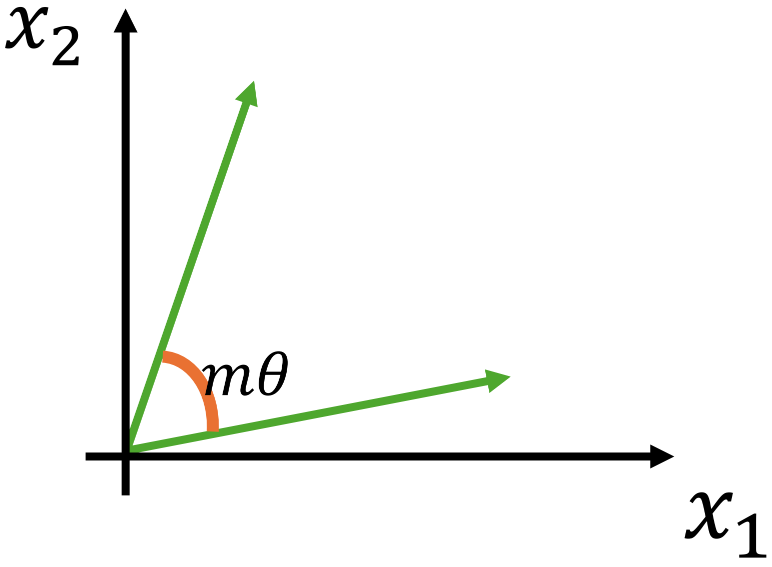
$$\begin{matrix}\square\\\square\end{matrix} = \begin{bmatrix} \cos(m\theta_3) & -\sin(m\theta_3) \\ \sin(m\theta_3) & \cos(m\theta_3) \end{bmatrix} \begin{matrix}\square\\\square\end{matrix}$$

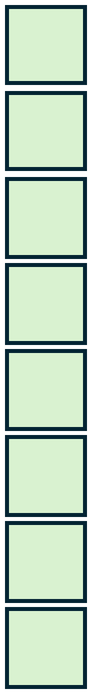
$$\begin{matrix}\square\\\square\end{matrix} = \begin{bmatrix} \cos(m\theta_4) & -\sin(m\theta_4) \\ \sin(m\theta_4) & \cos(m\theta_4) \end{bmatrix} \begin{matrix}\square\\\square\end{matrix}$$

$\boldsymbol{W}_q \boldsymbol{x}_m$

Slide thanks to Jiabin Huang

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$





$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_1) & -\sin(m\theta_1) \\ \sin(m\theta_1) & \cos(m\theta_1) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_2) & -\sin(m\theta_2) \\ \sin(m\theta_2) & \cos(m\theta_2) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

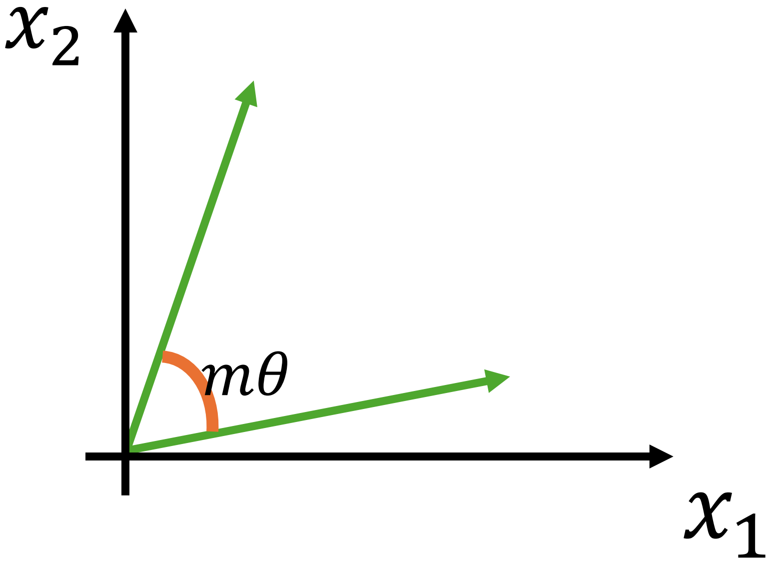
$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_3) & -\sin(m\theta_3) \\ \sin(m\theta_3) & \cos(m\theta_3) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

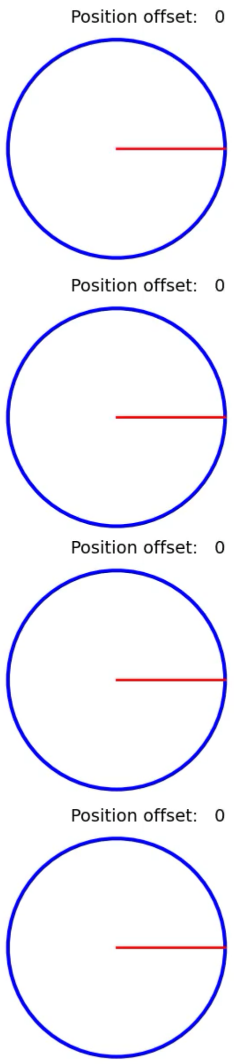
$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_4) & -\sin(m\theta_4) \\ \sin(m\theta_4) & \cos(m\theta_4) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

$f_q(\boldsymbol{x}_m, m)$ $\qquad$ $\boldsymbol{W}_q \boldsymbol{x}_m$

Slide thanks to Jiabin Huang

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$





$$= \begin{bmatrix} \cos(m\theta_1) & -\sin(m\theta_1) \\ \sin(m\theta_1) & \cos(m\theta_1) \end{bmatrix}$$

$$= \begin{bmatrix} \cos(m\theta_2) & -\sin(m\theta_2) \\ \sin(m\theta_2) & \cos(m\theta_2) \end{bmatrix}$$

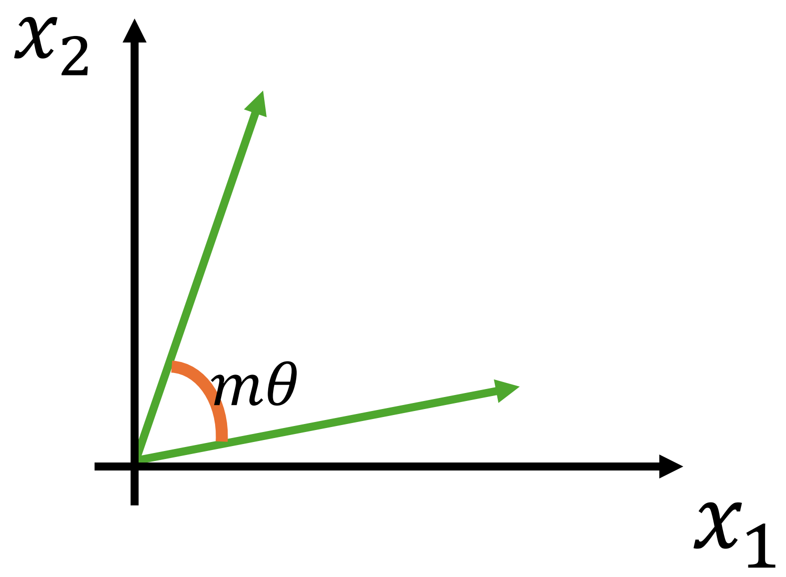
$$= \begin{bmatrix} \cos(m\theta_3) & -\sin(m\theta_3) \\ \sin(m\theta_3) & \cos(m\theta_3) \end{bmatrix}$$

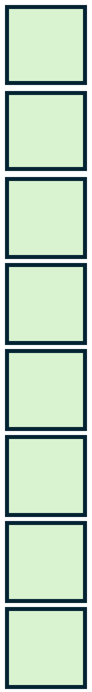
$$= \begin{bmatrix} \cos(m\theta_4) & -\sin(m\theta_4) \\ \sin(m\theta_4) & \cos(m\theta_4) \end{bmatrix}$$

$\boldsymbol{W}_q \boldsymbol{x}_m$

Slide thanks to Jiabin Huang

$$R_{\Theta}^{m} = \begin{bmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{bmatrix}$$





$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_1) & -\sin(m\theta_1) \\ \sin(m\theta_1) & \cos(m\theta_1) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_2) & -\sin(m\theta_2) \\ \sin(m\theta_2) & \cos(m\theta_2) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

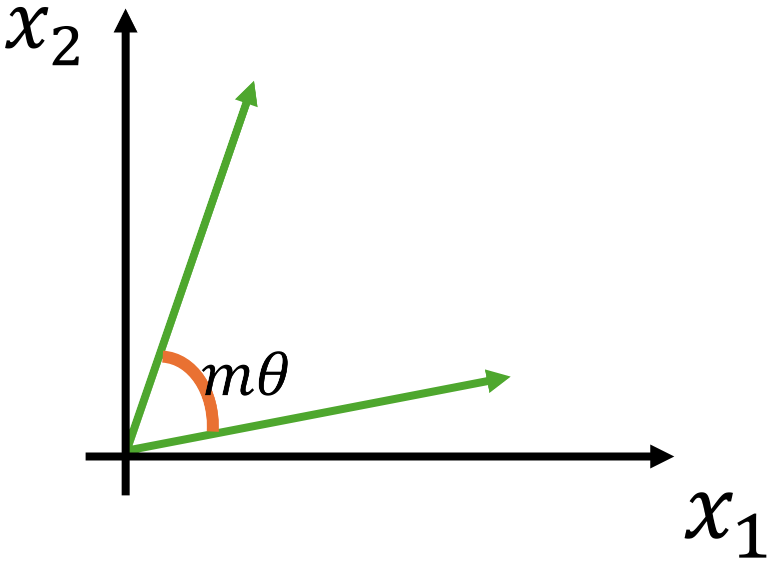
$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_3) & -\sin(m\theta_3) \\ \sin(m\theta_3) & \cos(m\theta_3) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

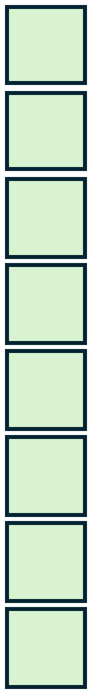
$$\begin{bmatrix} \square \\ \square \end{bmatrix} = \begin{bmatrix} \cos(m\theta_4) & -\sin(m\theta_4) \\ \sin(m\theta_4) & \cos(m\theta_4) \end{bmatrix} \begin{bmatrix} \square \\ \square \end{bmatrix}$$

$f_q(\boldsymbol{x}_m, m)$

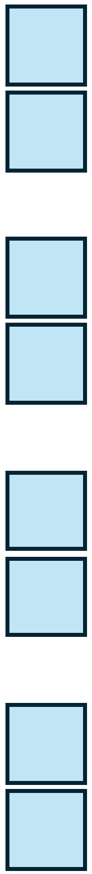
$\boldsymbol{W}_q \boldsymbol{x}_m$

Slide thanks to Jiabin Huang





$f_q(\boldsymbol{x}_m, m)$



$\boldsymbol{W}_q \boldsymbol{x}_m$

$$
\underbrace{}_{f_q(\boldsymbol{x}_m, m)} = \underbrace{\begin{pmatrix}
\cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\
\sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\
0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & \cdots & \cos m\theta_l & -\sin m\theta_l \\
0 & 0 & 0 & 0 & \cdots & \sin m\theta_l & \cos m\theta_l
\end{pmatrix}}_{R^m_\Theta}
\underbrace{\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ \vdots \\ q_8 \end{bmatrix}}_{W_q \boldsymbol{x}_m}
$$

Fast changing

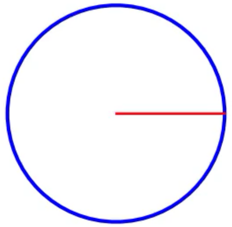
$$
f_q(\boldsymbol{x}_m, m) = \begin{bmatrix} \cos(m\theta_1) \\ \cos(m\theta_1) \\ \cos(m\theta_2) \\ \cos(m\theta_2) \\ \cos(m\theta_3) \\ \cos(m\theta_3) \\ \cos(m\theta_4) \\ \cos(m\theta_4) \end{bmatrix} \otimes \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} + \begin{bmatrix} \sin(m\theta_1) \\ \sin(m\theta_1) \\ \sin(m\theta_2) \\ \sin(m\theta_2) \\ \sin(m\theta_3) \\ \sin(m\theta_3) \\ \sin(m\theta_4) \\ \sin(m\theta_4) \end{bmatrix} \otimes \begin{bmatrix} -q_2 \\ q_1 \\ -q_4 \\ q_3 \\ -q_6 \\ q_5 \\ -q_8 \\ q_7 \end{bmatrix}
$$

Slowly changing

$$\theta_i = N^{-2i/d} \qquad N = 10{,}000$$

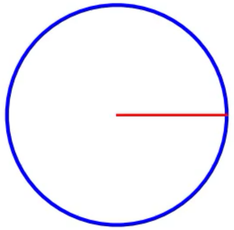
Slide thanks to Jiabin Huang

Position offset: 0



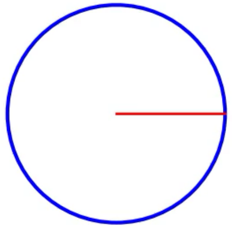
$$\begin{bmatrix}\square\\ \square\end{bmatrix} = \begin{bmatrix}\cos(m\theta_1) & -\sin(m\theta_1)\\ \sin(m\theta_1) & \cos(m\theta_1)\end{bmatrix}\begin{bmatrix}\square\\ \square\end{bmatrix}$$

Position offset: 0



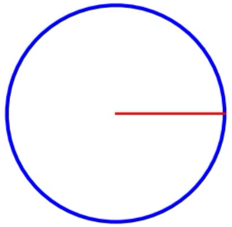
$$\begin{bmatrix}\square\\ \square\end{bmatrix} = \begin{bmatrix}\cos(m\theta_2) & -\sin(m\theta_2)\\ \sin(m\theta_2) & \cos(m\theta_2)\end{bmatrix}\begin{bmatrix}\square\\ \square\end{bmatrix}$$

Position offset: 0



$$\begin{bmatrix}\square\\ \square\end{bmatrix} = \begin{bmatrix}\cos(m\theta_3) & -\sin(m\theta_3)\\ \sin(m\theta_3) & \cos(m\theta_3)\end{bmatrix}\begin{bmatrix}\square\\ \square\end{bmatrix}$$

Position offset: 0



$$\begin{bmatrix}\square\\ \square\end{bmatrix} = \begin{bmatrix}\cos(m\theta_4) & -\sin(m\theta_4)\\ \sin(m\theta_4) & \cos(m\theta_4)\end{bmatrix}\begin{bmatrix}\square\\ \square\end{bmatrix}$$

$f_q(\boldsymbol{x}_m, m)$ $\qquad\qquad$ $\boldsymbol{W}_q\boldsymbol{x}_m$

Slide thanks to Jiabin Huang

2D ROPE

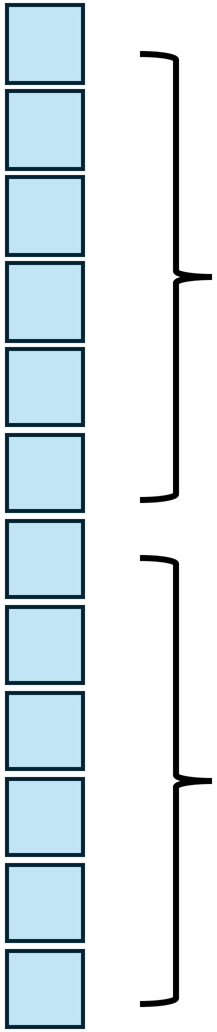
Patch (m, n)

$$
\begin{bmatrix}
\cos(\theta_1 x) & -\sin(\theta_1 x) & 0 & 0 & \dots & 0 & 0 \\
\sin(\theta_1 x) & \cos(\theta_1 x) & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \cos(\theta_2 x) & -\sin(\theta_2 x) & \dots & 0 & 0 \\
0 & 0 & \sin(\theta_2 x) & \cos(\theta_2 x) & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \dots & \cos(\theta_l x) & -\sin(\theta_l x) \\
0 & 0 & 0 & & \dots & \sin(\theta_l x) & \cos(\theta_l x)
\end{bmatrix}
$$

$$R_x$$

$$
\begin{bmatrix}
\cos(\theta_1 y) & -\sin(\theta_1 y) & 0 & 0 & \dots & 0 & 0 \\
\sin(\theta_1 y) & \cos(\theta_1 y) & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \cos(\theta_2 y) & -\sin(\theta_2 y) & \dots & 0 & 0 \\
0 & 0 & \sin(\theta_2 y) & \cos(\theta_2 y) & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \dots & \cos(\theta_l y) & -\sin(\theta_l y) \\
0 & 0 & 0 & 0 & \dots & \sin(\theta_l y) & \cos(\theta_l y)
\end{bmatrix}
$$

$$R_y$$



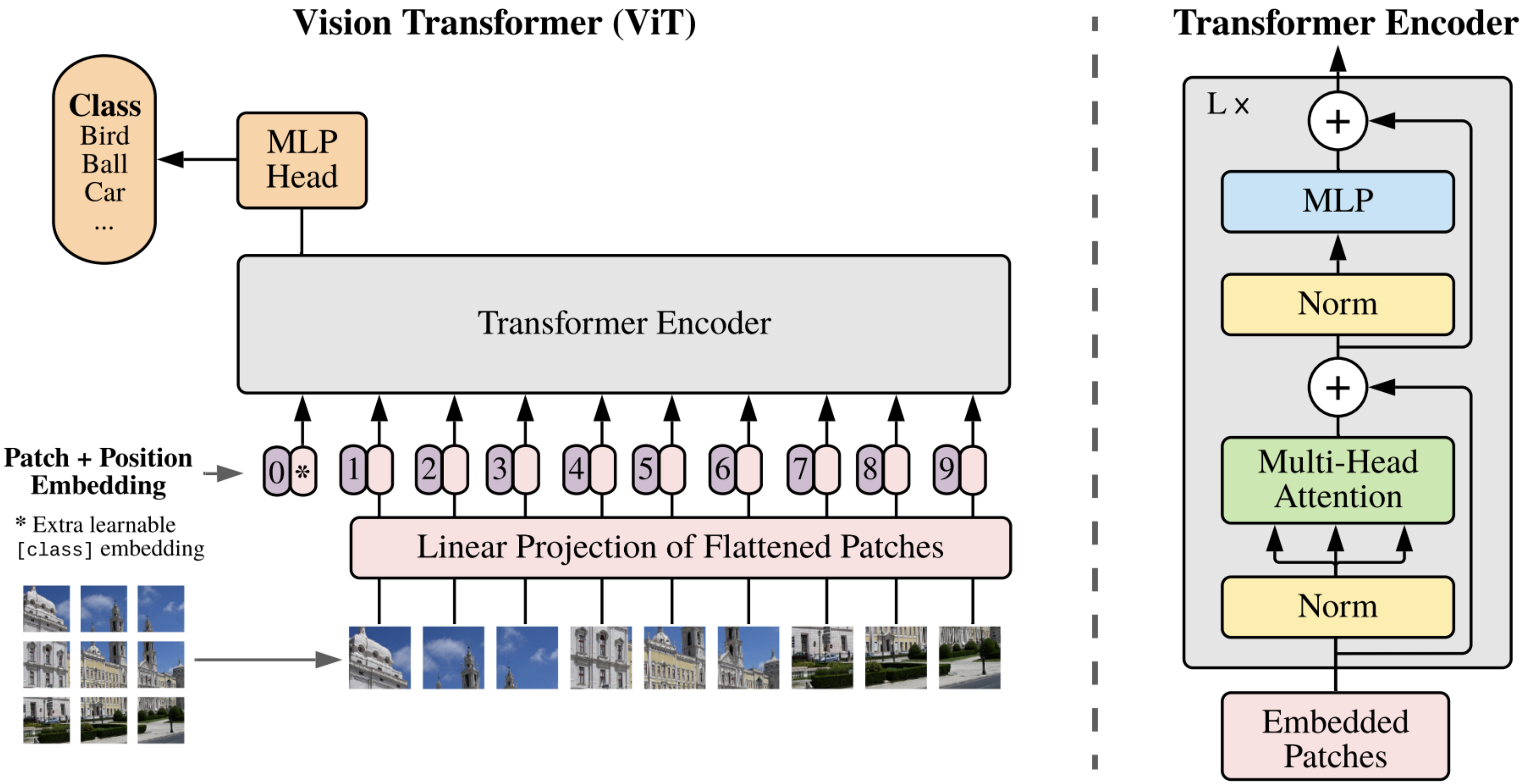
Rope on X values

Rope on Y values

Switching Gears

Using ViT!

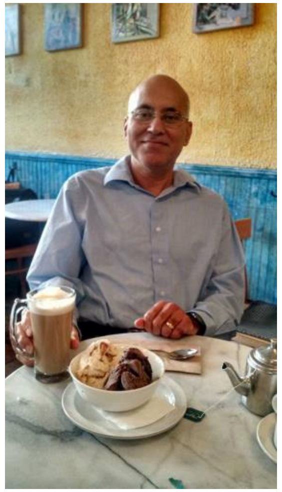
Vision Transformer



An image is worth 16x16 words, Dosovitskiy e tal. ICLR 2021

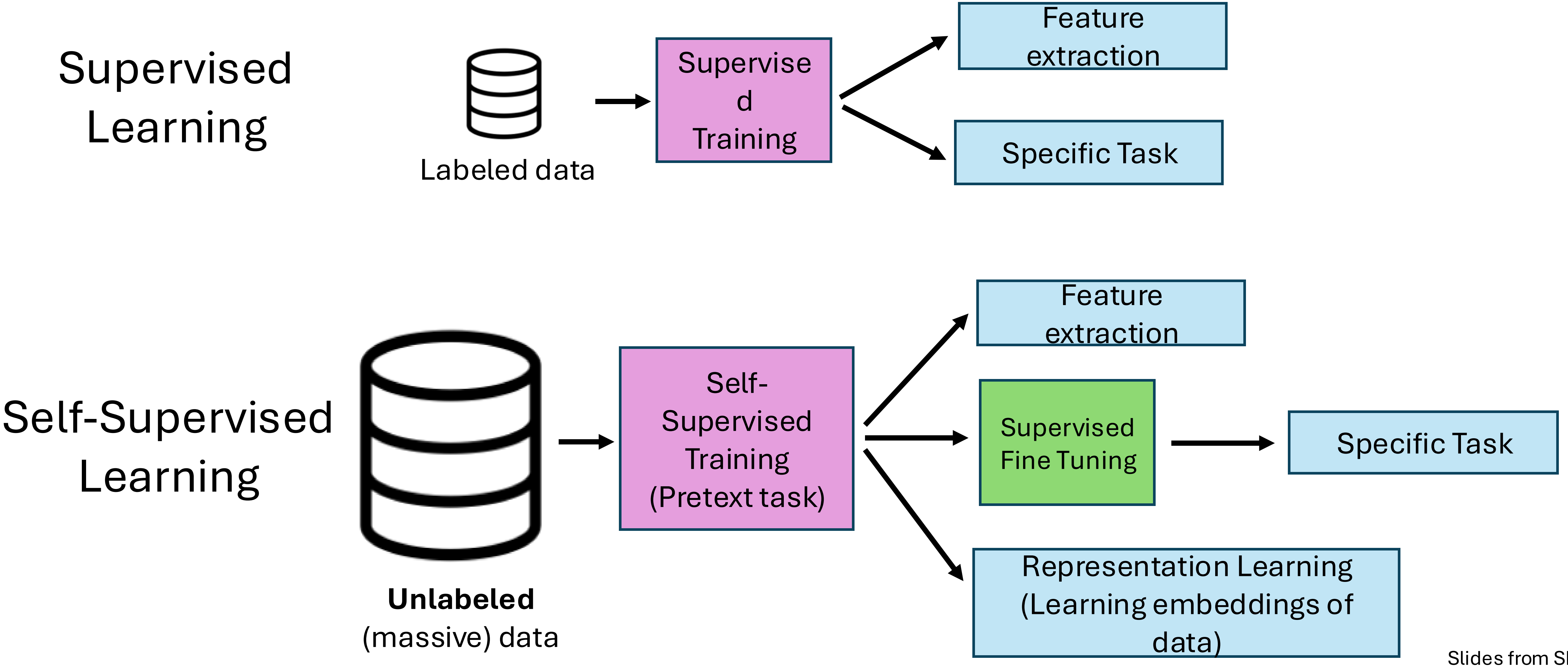
Is semantic supervision necessary to learn good representations?

- Manual labeling doesn't scale, suffers from biases
- Plenty of unlabeled visual data already, and growing really fast
- *And subject of the Gelato Bet:*
- *Made on Sept 23 2014*
- *If, by the first day of autumn (Sept 23) of 2015, a method will exist that can match or beat the performance of R-CNN on Pascal VOC detection, without the use of any extra, human annotations (e.g. ImageNet) as pre-training, Mr. Malik promises to buy Mr. Efros one (1) gelato (2 scoops: one chocolate, one vanilla).*



The Gelato Bet

Self supervised learning with ViTs



Slides from Shai Bagon

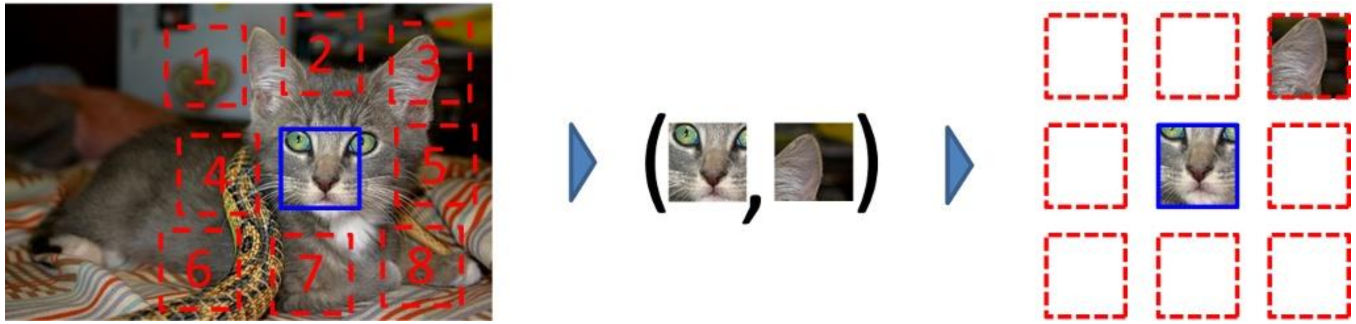
Pre-train representations on a pre-text task

E.g. Colorization

After pre-training, use representation for down-stream tasks.

Many other possibilities,

- Spatial relationship between pair of patches



- Predict sound / frame ordering in a video
- Encourage two augmentations of same image to be closer to each other than to another image
- Predict hidden image patches from context

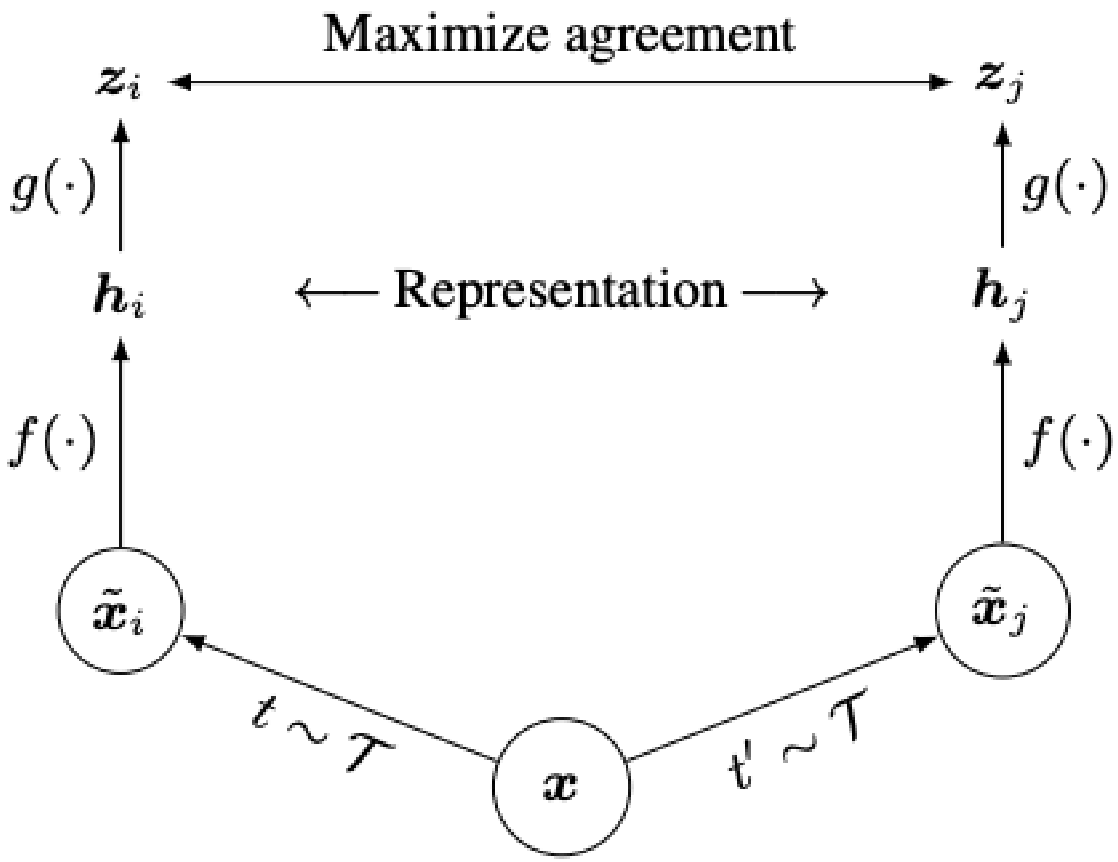
[Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction], Zhang et al. CVPR 2017
[Context as Supervisory Signal: Discovering Objects with Predictable Context], Doersch et al. ICCV 2015

Slide credit Saurabh Gupta

Contrastive Learning

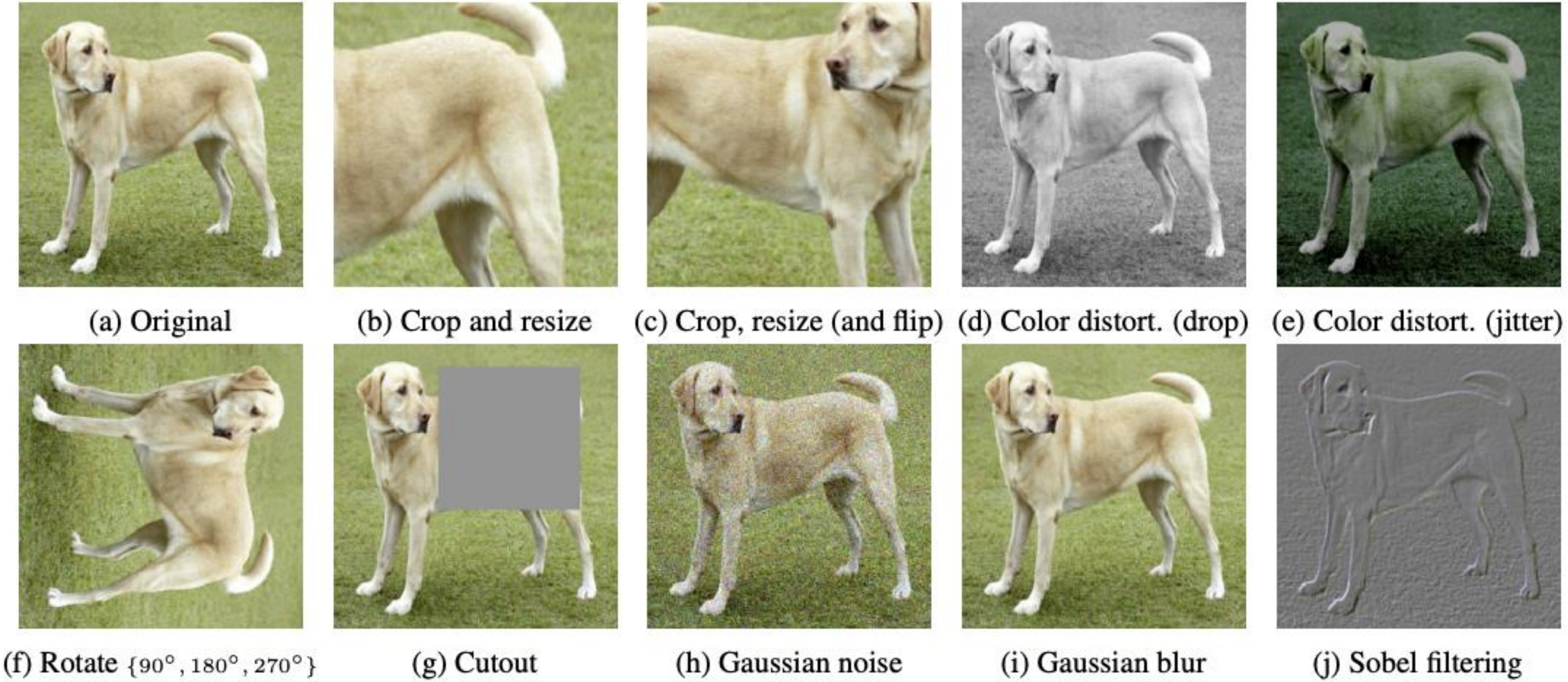
- Encourage two augmentations of an image to be close.
- Using a contrastive loss:

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$



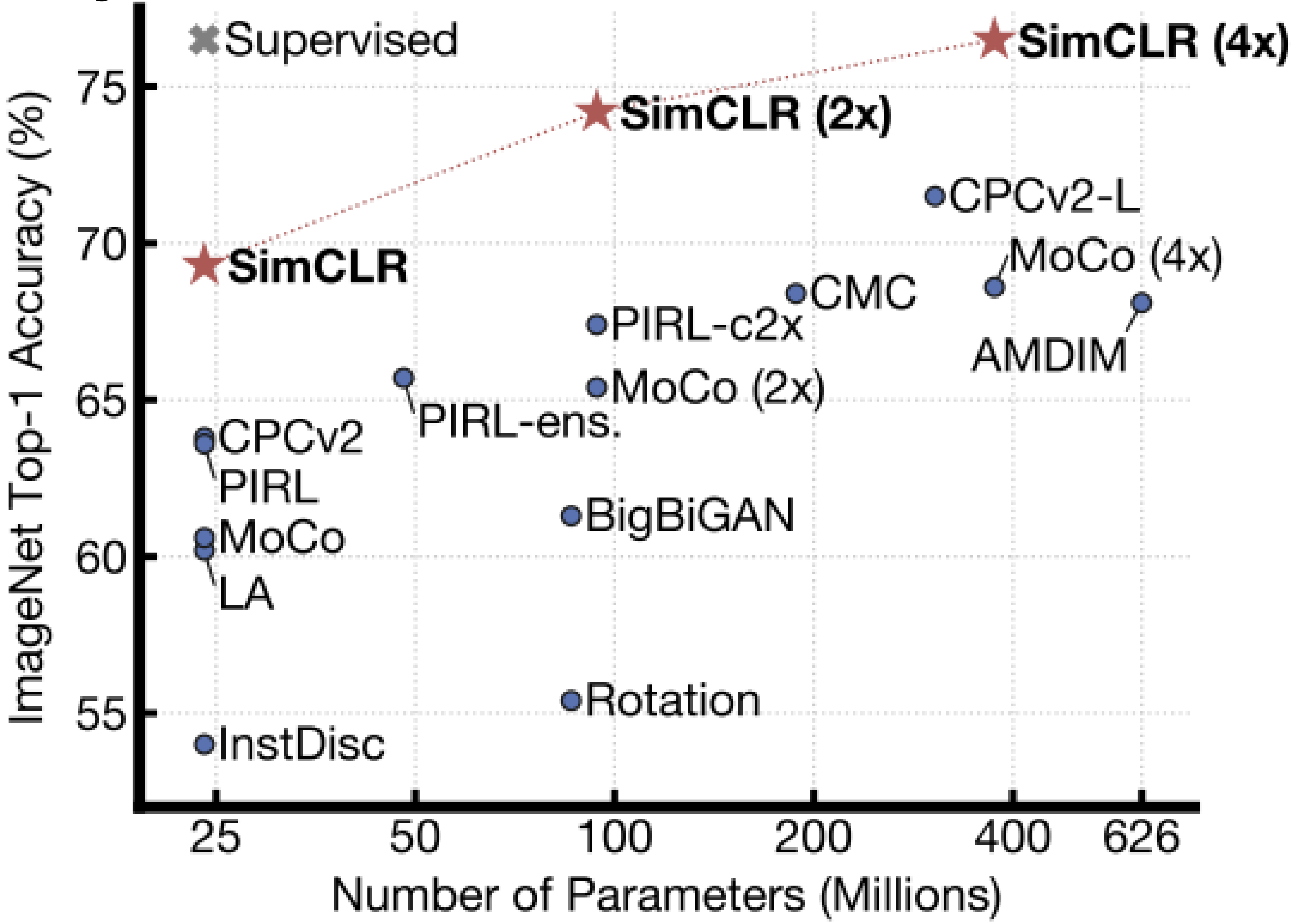
A Simple Framework for Contrastive Learning of Visual Representations, Chen et al. ICML 2020
See also: Momentum Contrast for Unsupervised Visual Representation Learning, He et al. CVPR 2020

Augmentations



[A Simple Framework for Contrastive Learning of Visual Representations](), Chen et al. ICML 2020

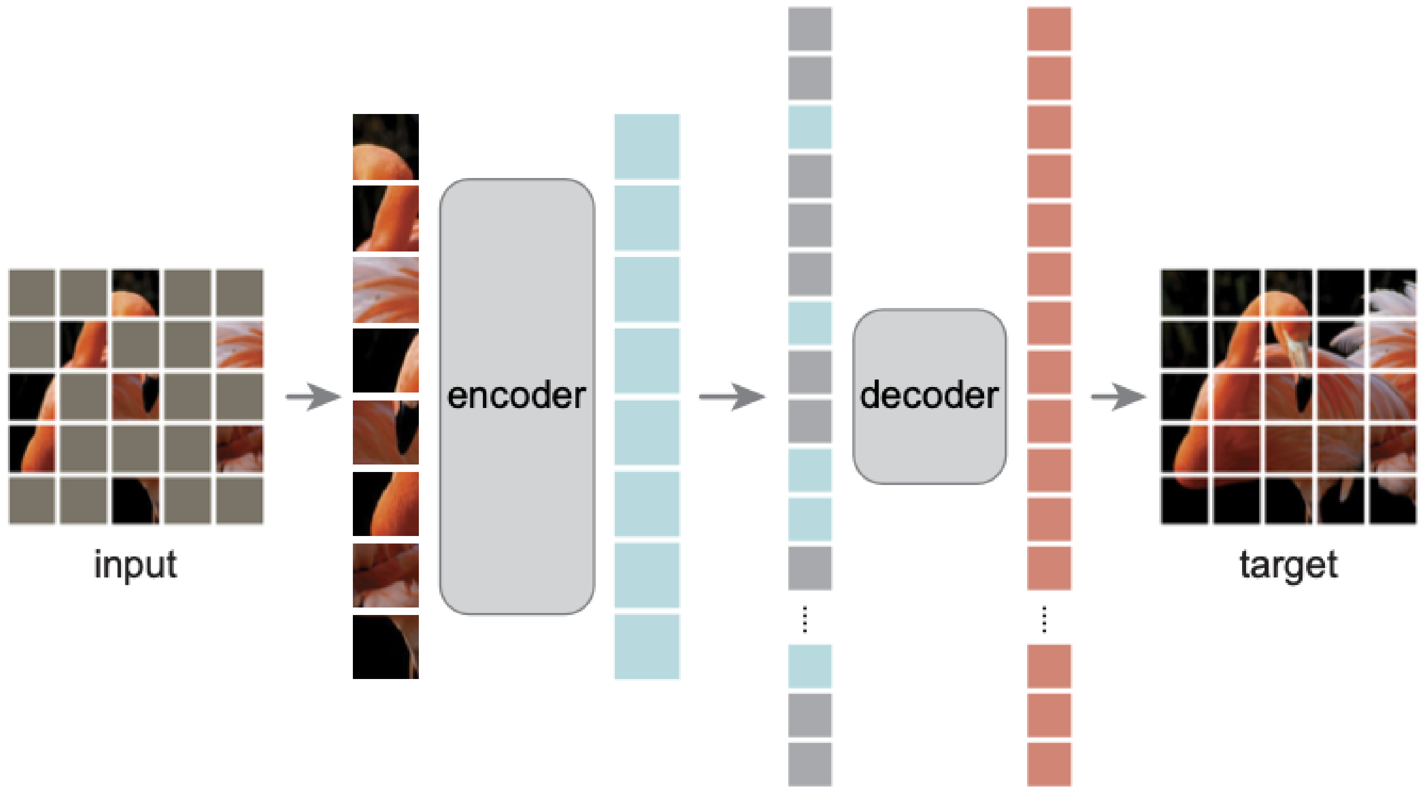
Was a very active area



[A Simple Framework for Contrastive Learning of Visual Representations](), Chen et al. ICML 2020

Masked Auto-Encoders

- Mask out image patches, predict masked patches from visible patches.
- Pre-train encoder & decoder.
- Use encoder as an image representation.
- Used as frozen backbone for various vision tasks like detection/segmentation (later in class)



Masked Autoencoders Are Scalable Vision Learners, He et al. CVPR 2022

Slide credit Saurabh Gupta

Better than semantic supervision on ImageNet 1K!

method	pre-train data	AP^{box} ViT-B	AP^{box} ViT-L	AP^{mask} ViT-B	AP^{mask} ViT-L
supervised	IN1K w/ labels	47.9	49.3	42.9	43.9
MoCo v3	IN1K	47.9	49.3	42.7	44.0
BEiT	IN1K+DALLE	49.8	**53.3**	44.4	47.1
MAE	IN1K	**50.3**	**53.3**	**44.9**	**47.2**

Table 4. **COCO object detection and segmentation** using a ViT Mask R-CNN baseline. All entries are based on our implementation. Self-supervised entries use IN1K data *without* labels. Mask AP follows a similar trend as box AP.

method	pre-train data	ViT-B	ViT-L
supervised	IN1K w/ labels	47.4	49.9
MoCo v3	IN1K	47.3	49.1
BEiT	IN1K+DALLE	47.1	53.3
MAE	IN1K	**48.1**	**53.6**

Table 5. **ADE20K semantic segmentation** (mIoU) using UperNet. BEiT results are reproduced using the official code. Other entries are based on our implementation. Self-supervised entries use IN1K data *without* labels.

Scaling behavior →

dataset	ViT-B	ViT-L	ViT-H	ViT-H_{448}	prev best
iNat 2017	70.5	75.7	79.3	**83.4**	75.4 [55]
iNat 2018	75.4	80.1	83.0	**86.8**	81.2 [54]
iNat 2019	80.5	83.4	85.7	**88.3**	84.1 [54]
Places205	63.9	65.8	65.9	**66.8**	66.0 [19]†
Places365	57.9	59.4	59.8	**60.3**	58.0 [40]‡

Table 6. **Transfer learning accuracy on classification datasets**, using MAE pre-trained on IN1K and then fine-tuned. We provide system-level comparisons with the previous best results.

†: pre-trained on 1 billion images. ‡: pre-trained on 3.5 billion images.

Masked Autoencoders Are Scalable Vision Learners, He et al. CVPR 2022

Slides credit Saurabh Gupta

Improves performance on ImageNet itself

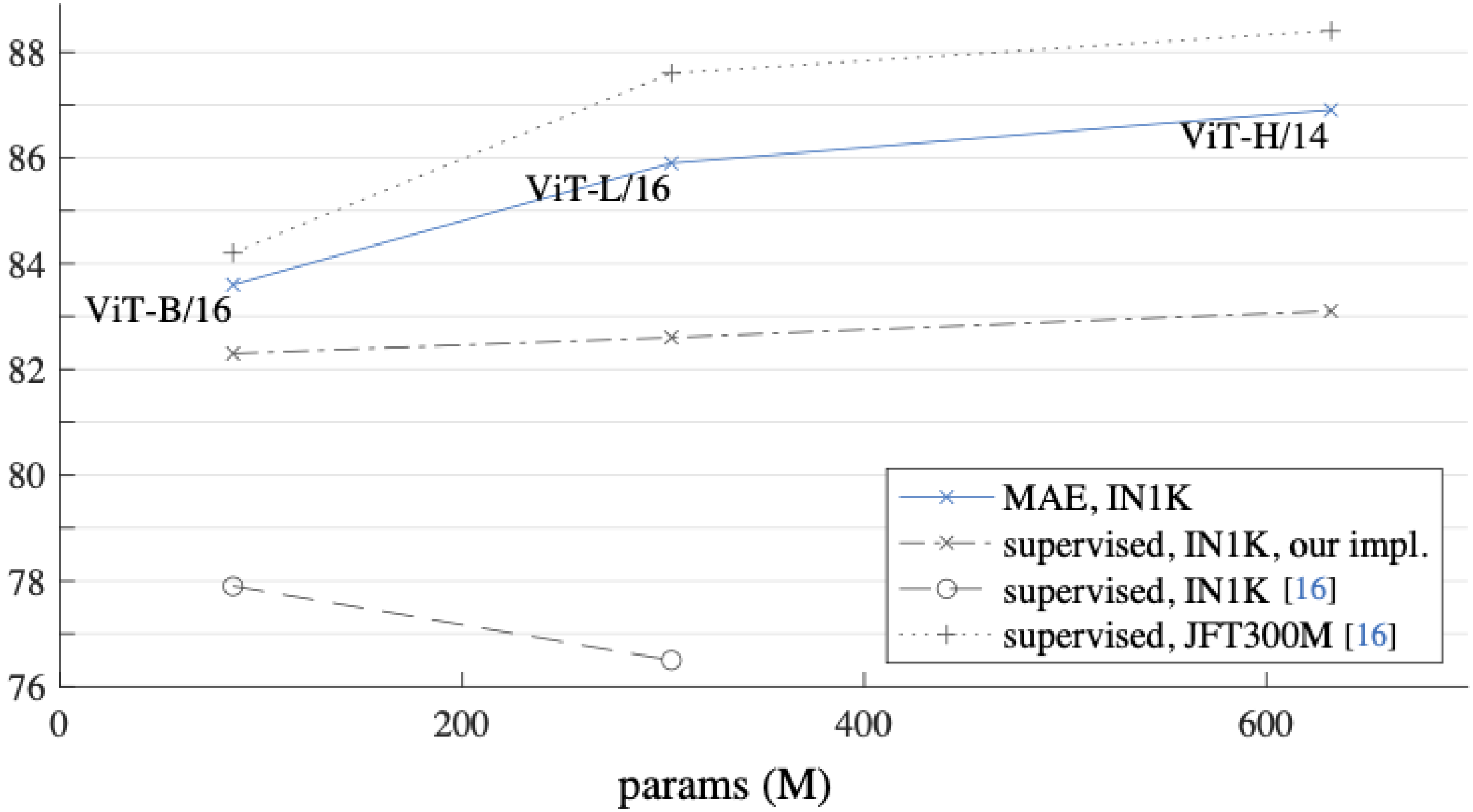


Figure 8. **MAE pre-training *vs*. supervised pre-training**, evaluated by fine-tuning in ImageNet-1K (224 size). We compare with the original ViT results [16] trained in IN1K or JFT300M.

Masked Autoencoders Are Scalable Vision Learners, He et al. CVPR 2022

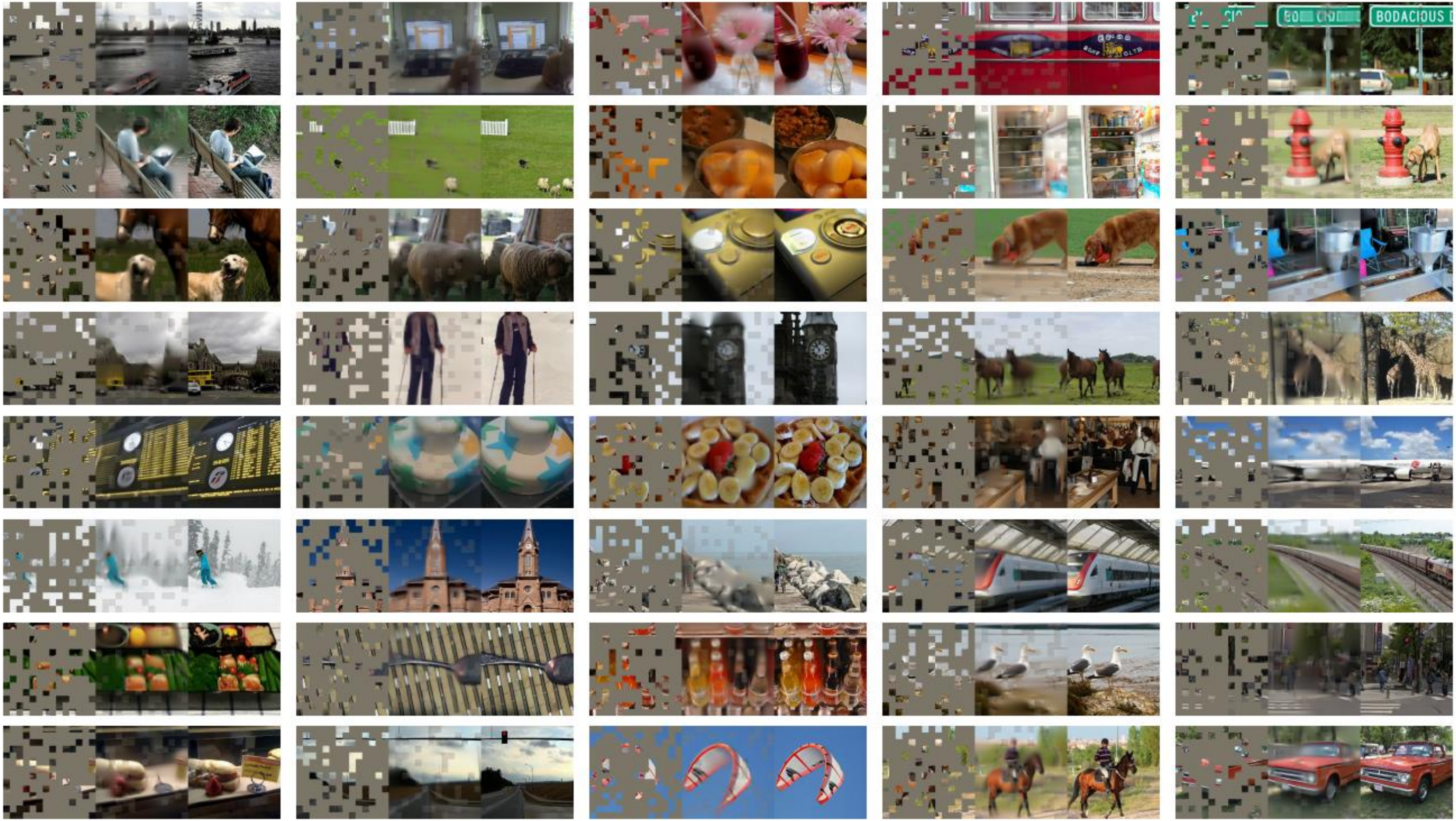
Slides from Saurabh Gupta

Better than past self-supervision approaches

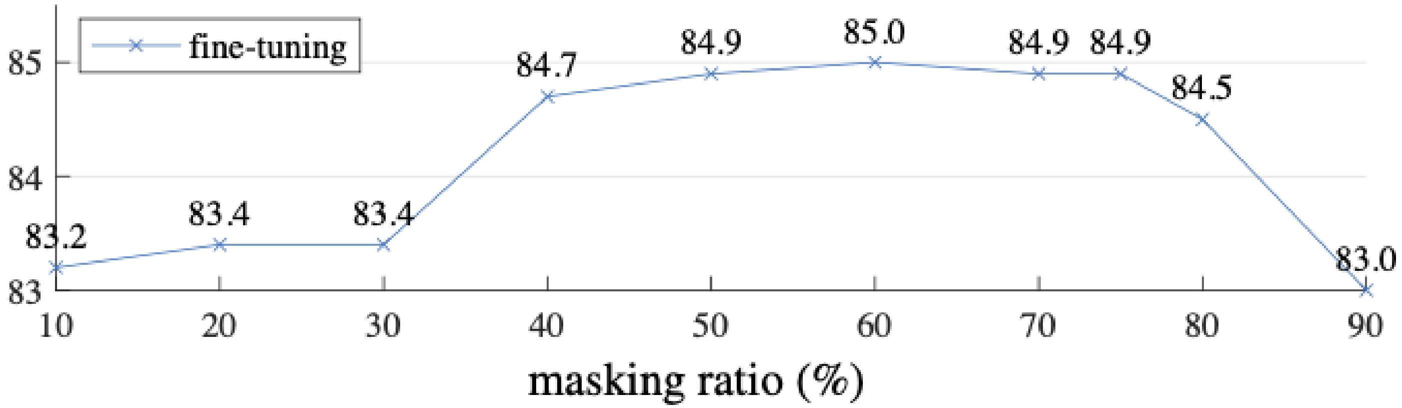
method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H_{448}
scratch, our impl.	-	82.3	82.6	83.1	-
DINO [5]	IN1K	82.8	-	-	-
MoCo v3 [9]	IN1K	83.2	84.1	-	-
BEiT [2]	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	<u>83.6</u>	<u>85.9</u>	<u>86.9</u>	**87.8**

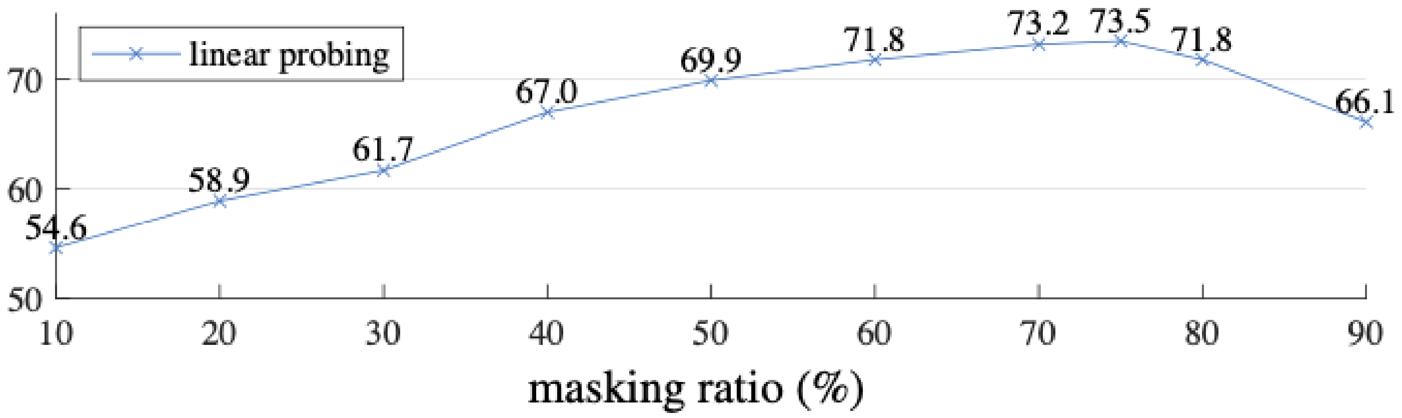
Masked Autoencoders Are Scalable Vision Learners, He et al. CVPR 2022

Slides from Saurabh Gupta


BODACIOUS

Ablations





Need high masking ratio for good learning.
NLP models use 15-20% masking ratio.

case	ft	lin	FLOPs
encoder w/ [M]	84.2	59.6	3.3×
encoder w/o [M]	**84.9**	**73.5**	**1**×

Faster and better to not input masked out patches to encoder

case	ft	lin
pixel (w/o norm)	84.9	73.5
pixel (w/ norm)	**85.4**	**73.9**
PCA	84.6	72.3
dVAE token	85.3	71.6

Normalized pixels are a better target than discrete tokens / PCA coefficients

Masked Autoencoders Are Scalable Vision Learners, He et al. CVPR 2022

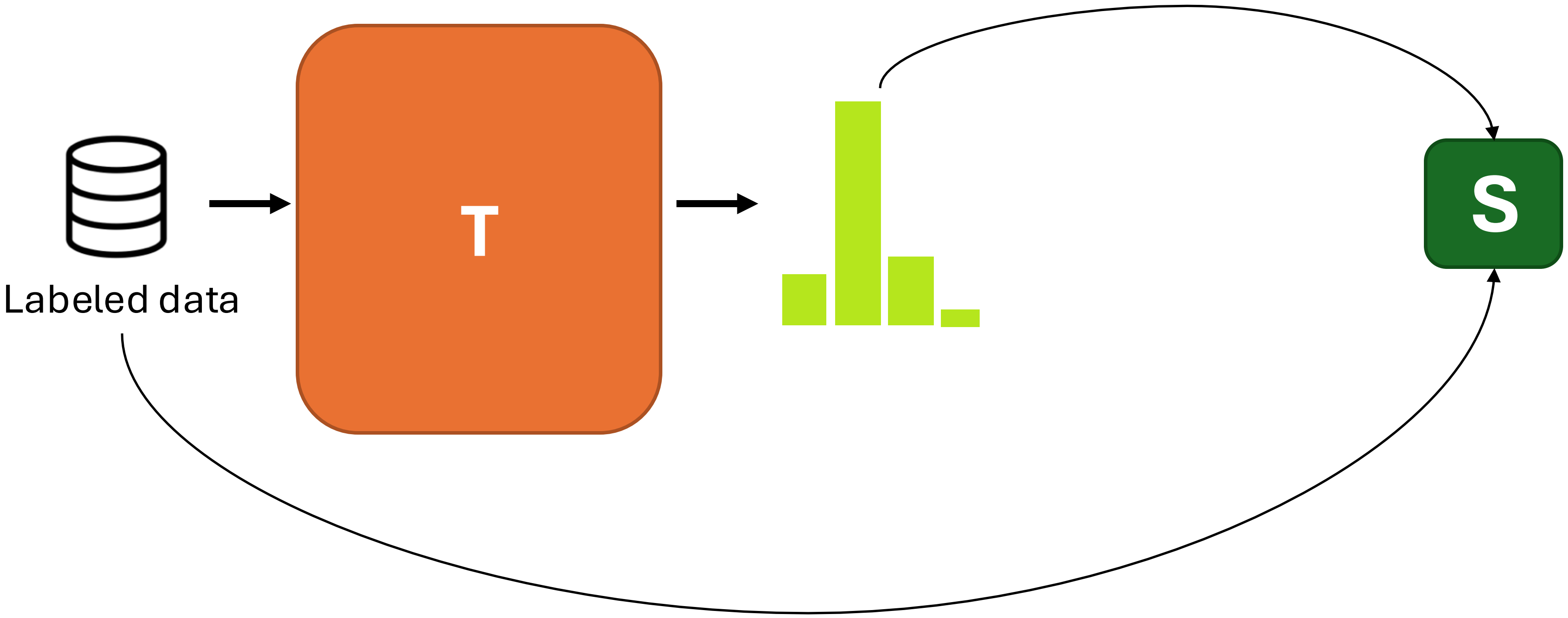
Slides from Saurabh Gupta

Take away

- Able to pre-train data-hungry ViT model
- Very well structured experiments! Read and learn from it.

DiNO - Approach

- Self supervised learning as a special case of **knowledge distillation**

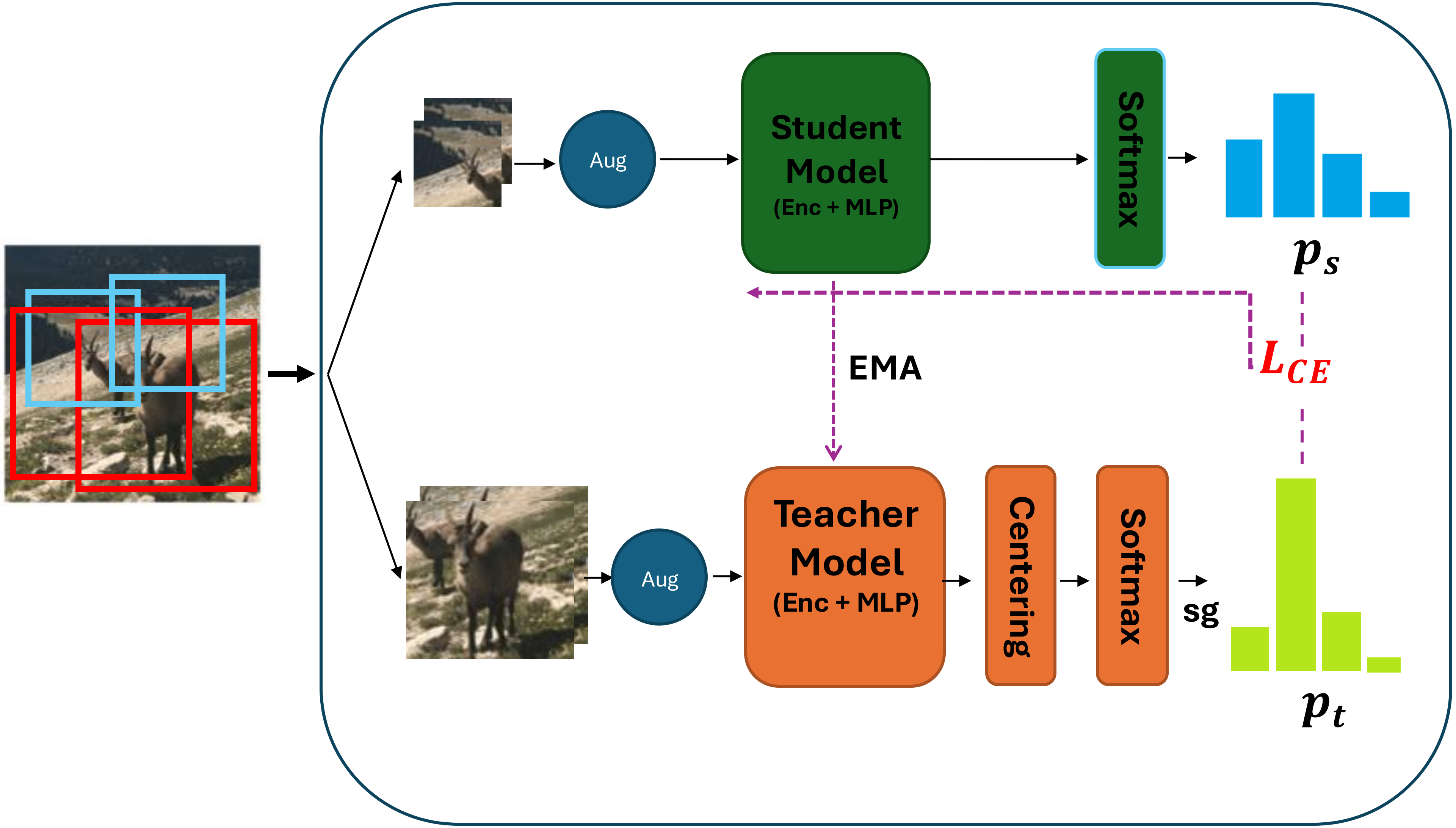


Slides from Shai Bagon

DiNO - Training

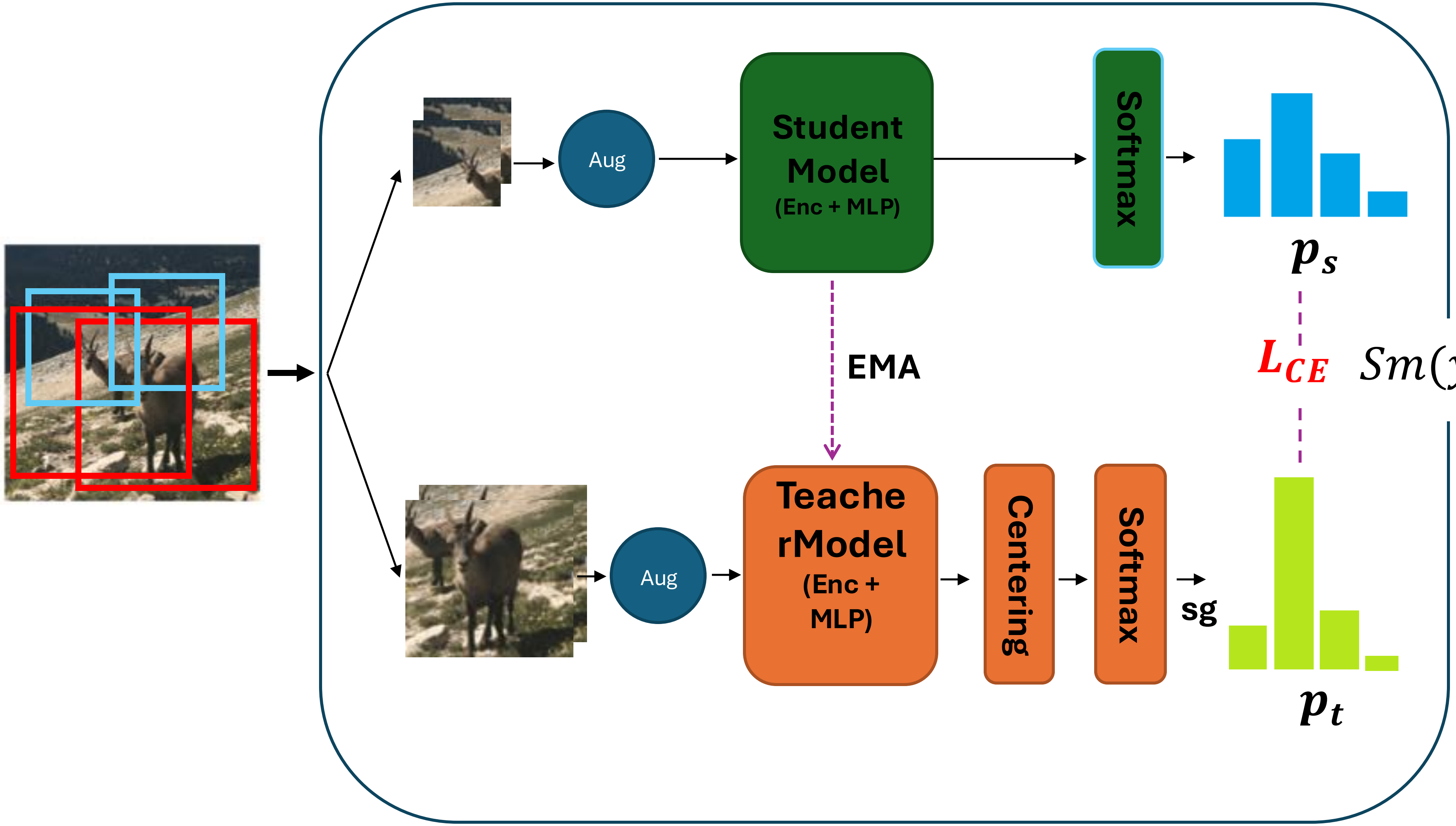
Slides from Shai Bagon

DiNO - Training



Slides from Shai Bagon

DiNO - Training



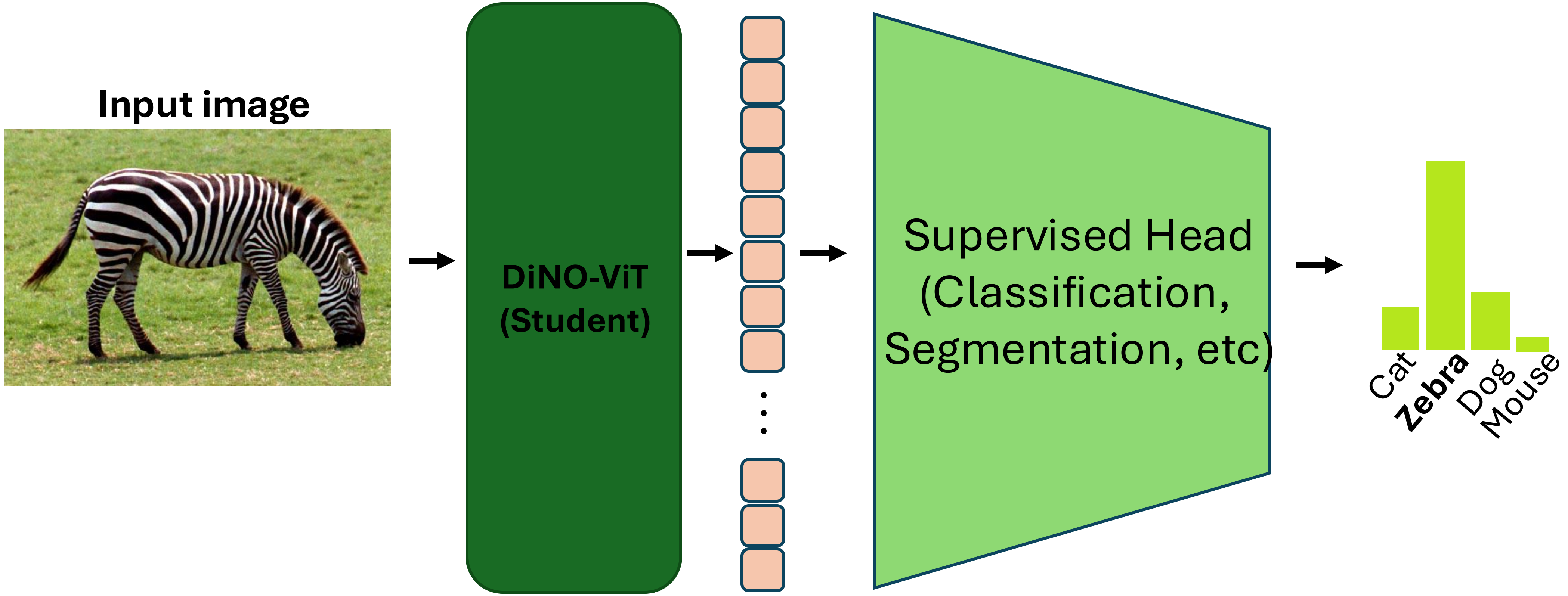
$$Sm(y) = \frac{exp(y_i/\tau)}{\sum_{j=1}^{K} exp(y_j/\tau)}$$

Slides from Shai Bagon

DiNO - Explanation

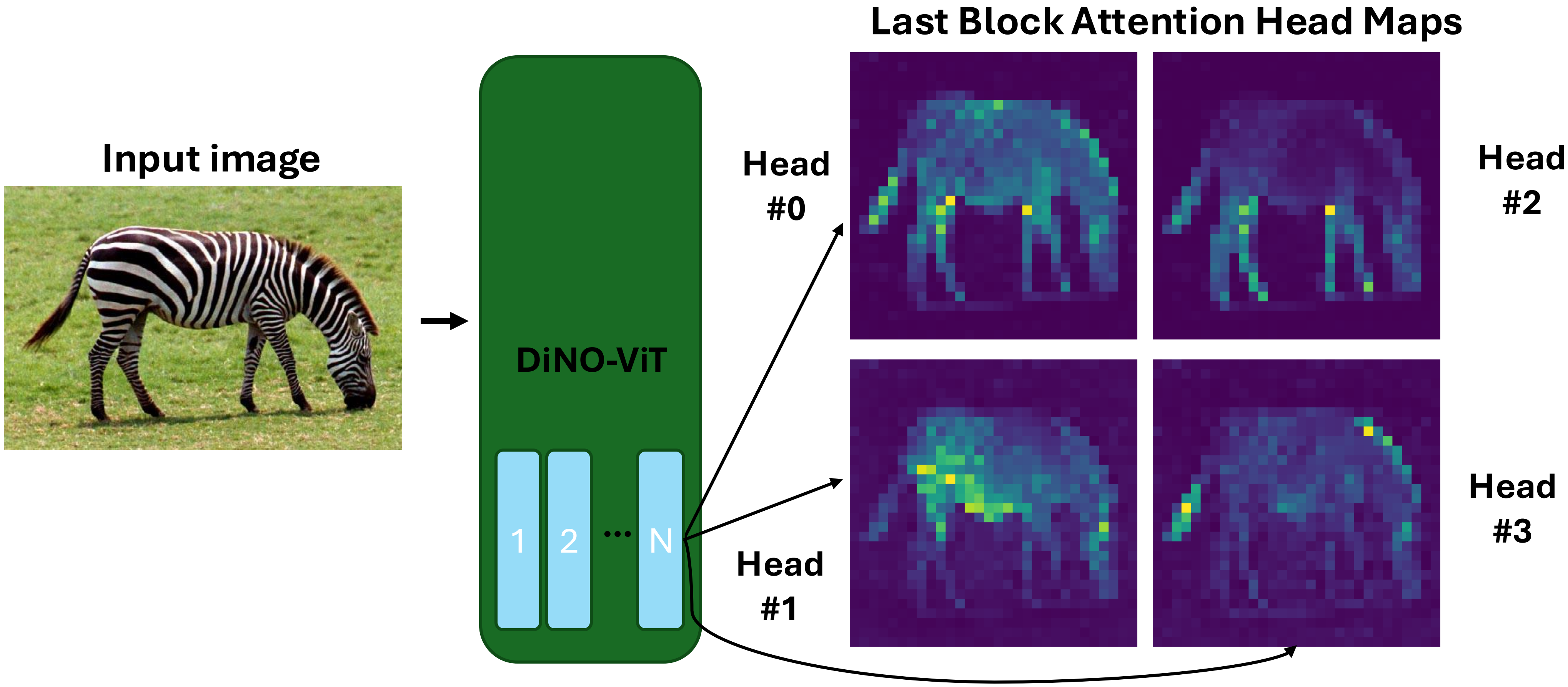
- Augmentations
  - Tells the model what to ignore
    - Collor jitter, Gaussian Blur, Solarize
    - Acts as a data prior
- “Global – local” cropping
- Teacher out-distribution sharpening via centering & Low-temperature in softmax
- The student encoder learns “abstract representations”
  - No awareness of “class labels” or meaning behind logits

DiNO - Inference #1



Slides from Shai Bagon

DiNO - Inference #2



Slides from Shai Bagon

DiNO - Inference #2

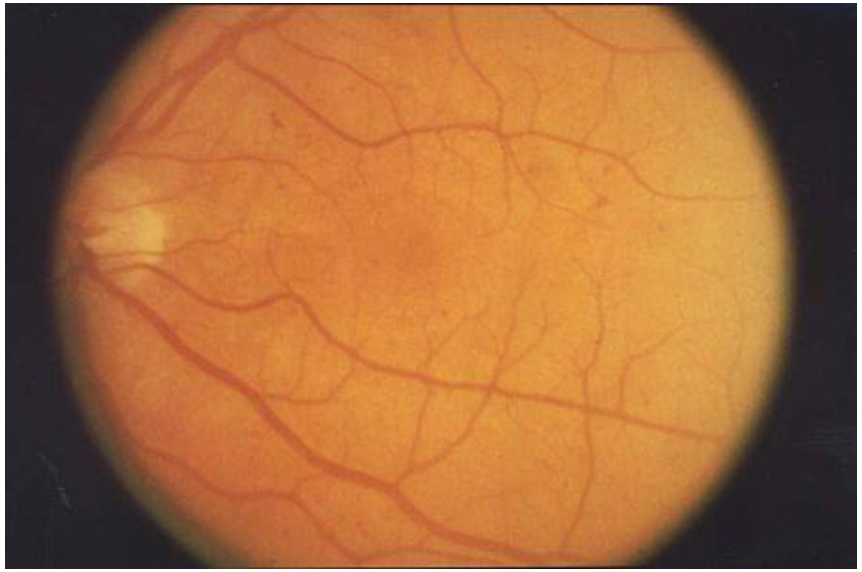
Input **Supervised Segmentation** **DiNO (SSL) Segmentation**

Slides from Shai Bagon

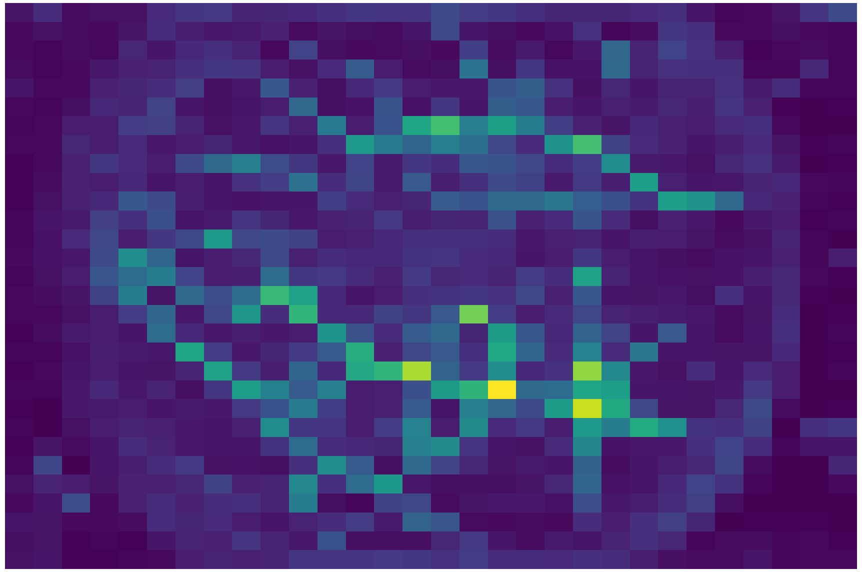
DiNO - Inference #2

- Self supervised learning also makes learned representations applicable to out-of-distribution data

Input image



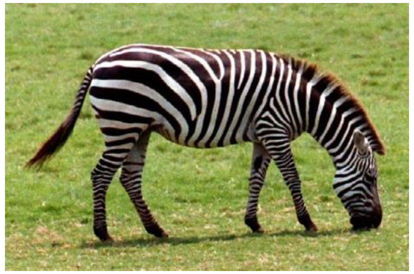
Last block attention map

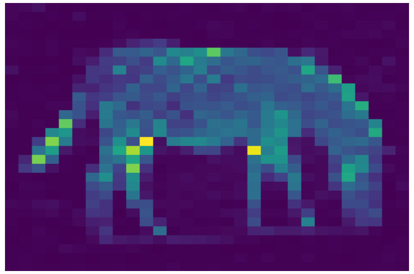


Slides from Shai Bagon

Usage

```
vit_model = torch.hub.load('facebookresearch/dino:main’,
                          f'dino_vits16', pretrained=True)
img = imread(‘zebra.png’)

x = vit_model.prepare_tokens(img)
for blk in vit_model.blocks[:-1]:
    x = blk(x)
attn_maps = vit_model.blocks[-1](x, return_attention=True)

# Choose head, Get attention map of class token
attn_map = attn_maps[0, HEAD, 0, 1:].reshape((1, 1, H_PATCHES, W_PATCHES))
attn_map = F.interpolate(attn_map, scale_factor=16, mode="nearest")
```

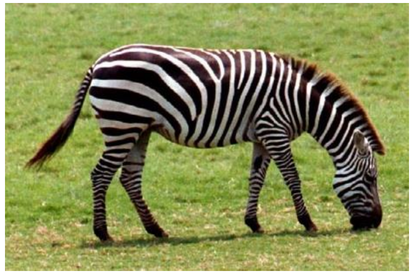


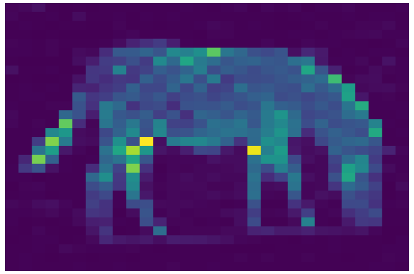


Slides from Shai Bagon

Usage

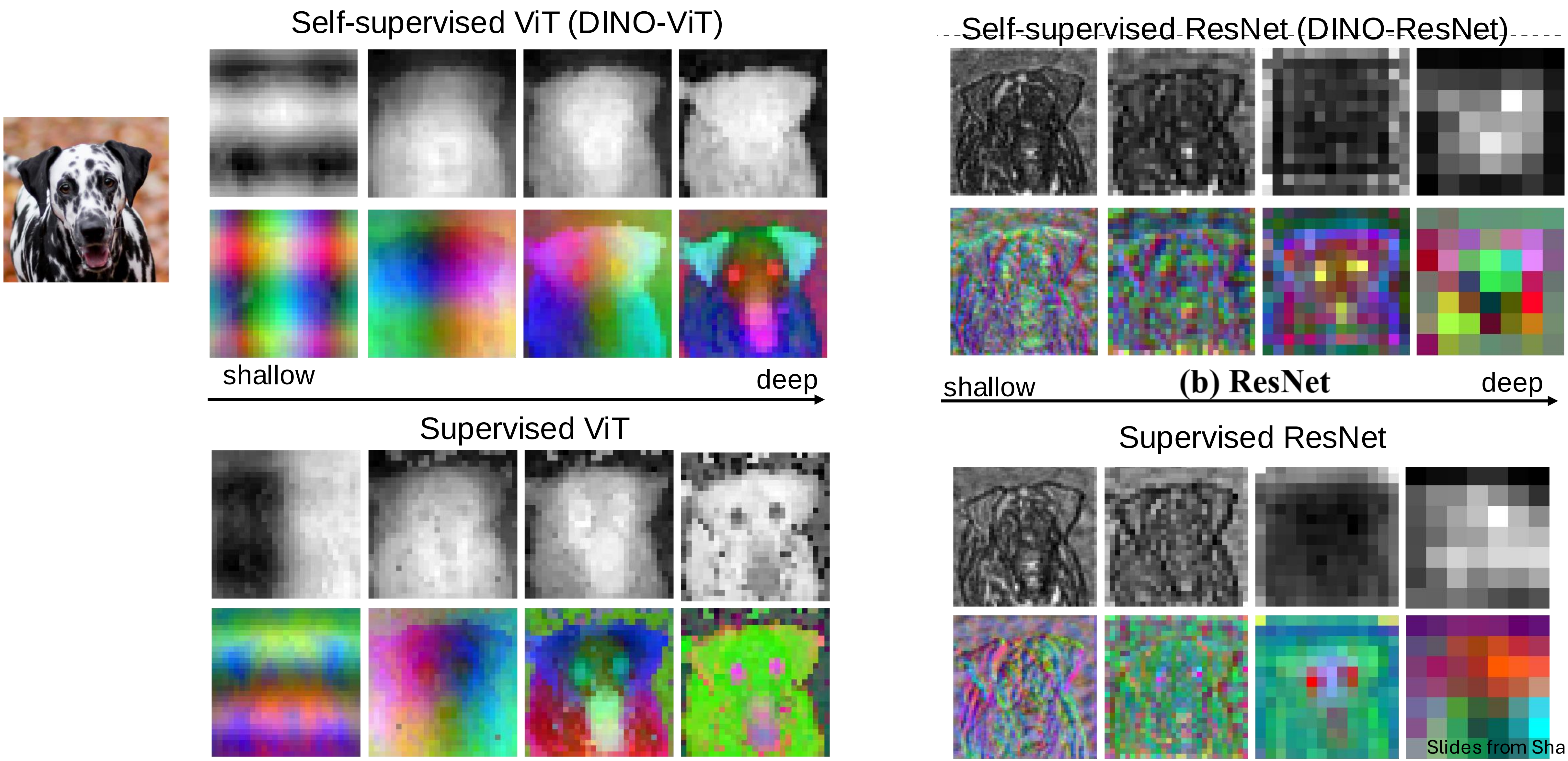
```
vit_model = torch.hub.load('facebookresearch/dino:main',
                           f'dino_vits16', pretrained=True)

img = imread('zebra.png')

attn_maps = vit_model.get_last_selfattention(img)

# Choose head, Get attention map of class token
attn_map = attn_maps[0, HEAD, 0, 1:].reshape((1, 1, H_PATCHES, W_PATCHES))
attn_map = F.interpolate(attn_map, scale_factor=16, mode="nearest")
```





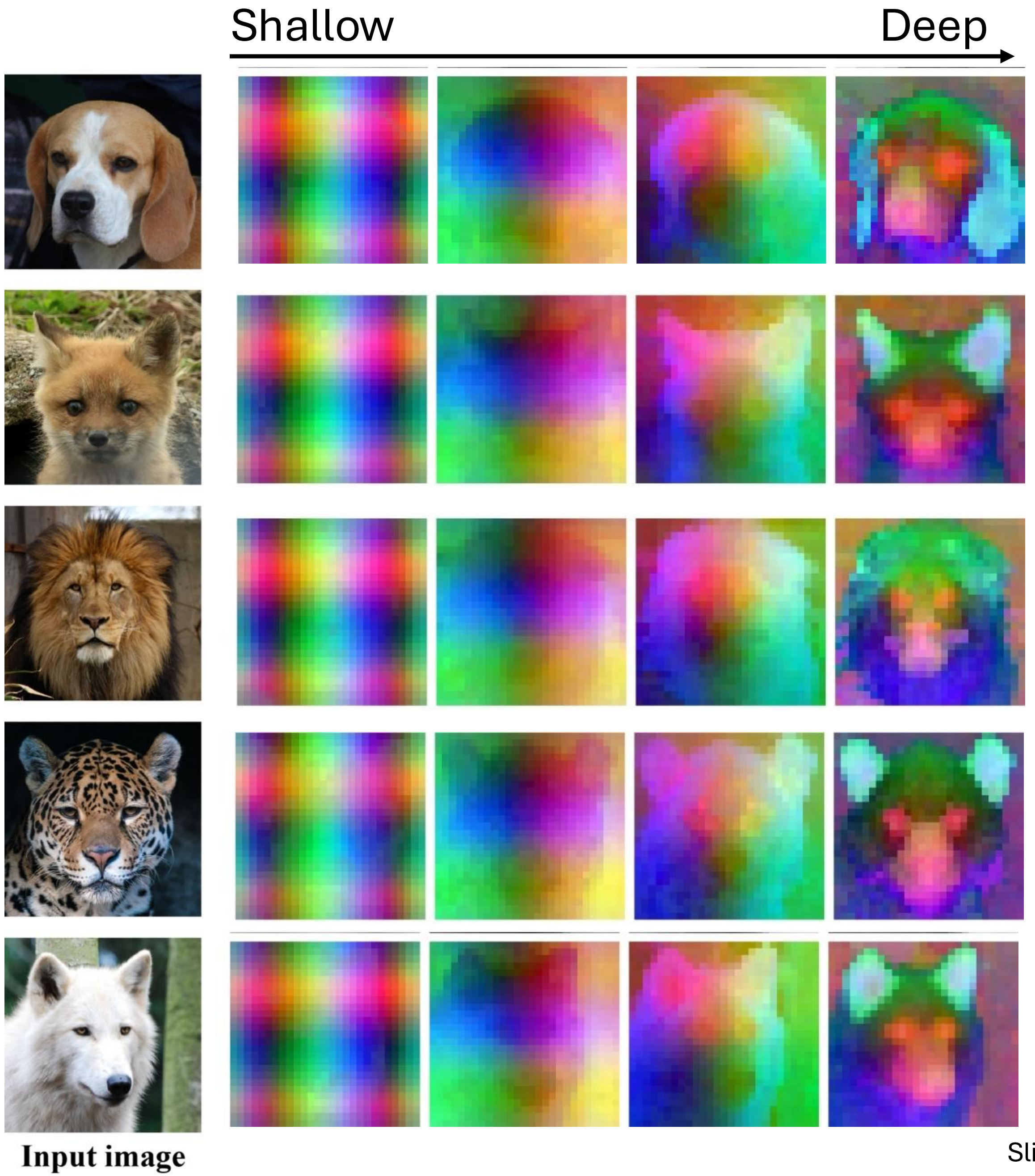
Slides from Shai Bagon

PCA (Keys) across layers

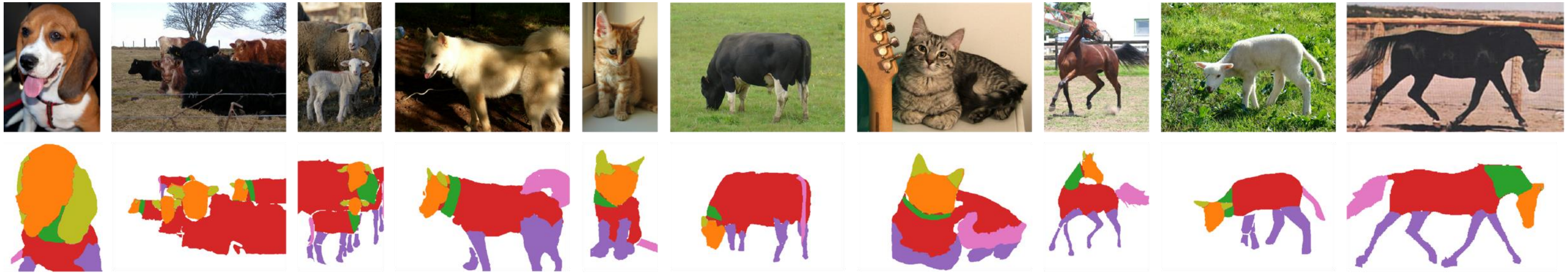


Slides from Shai Bagon

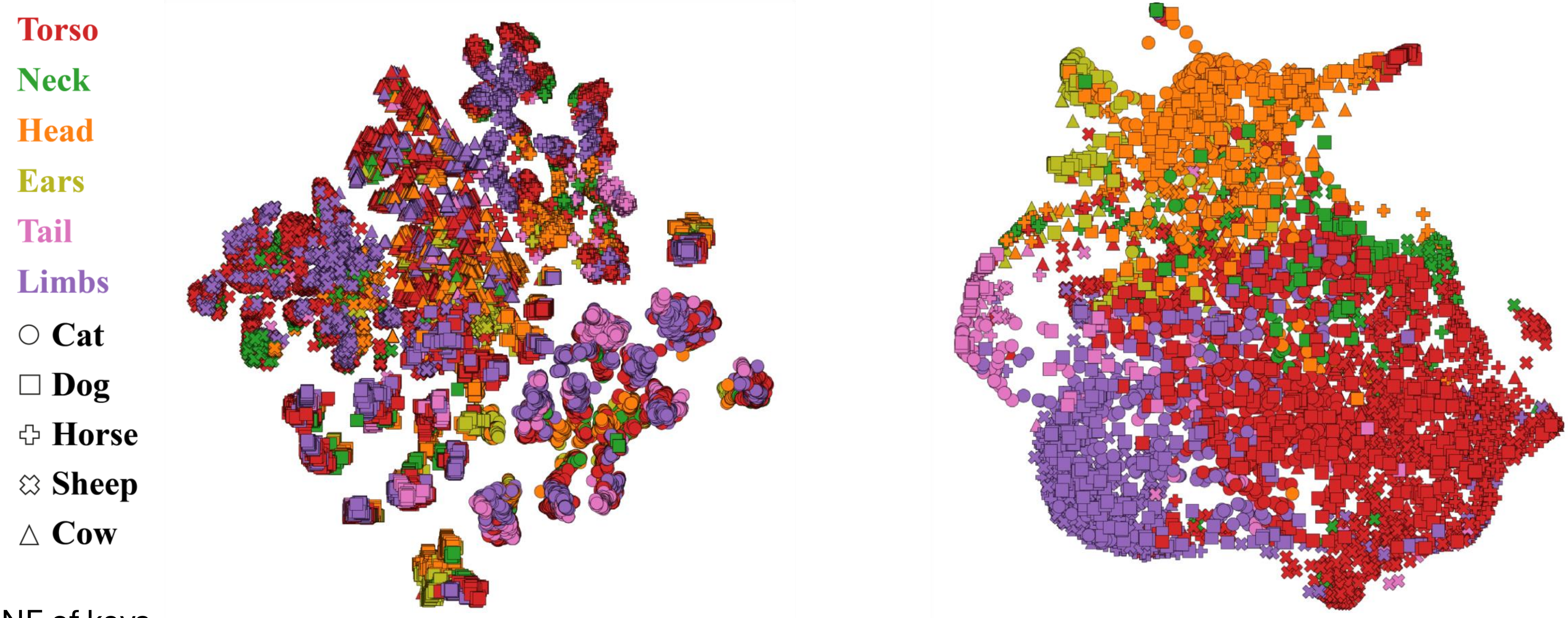
PCA – DiNO ViT



Amir, S., Gandelsman, Y., Bagon, S., Dekel, T.,
“Deep ViT Features as dense visual descriptors”



(a) Sample images and ground truth parts



(c) Supervised ViT

(b) Self-Supervised ViT (DINO-ViT)

T-SNE of keys visualization

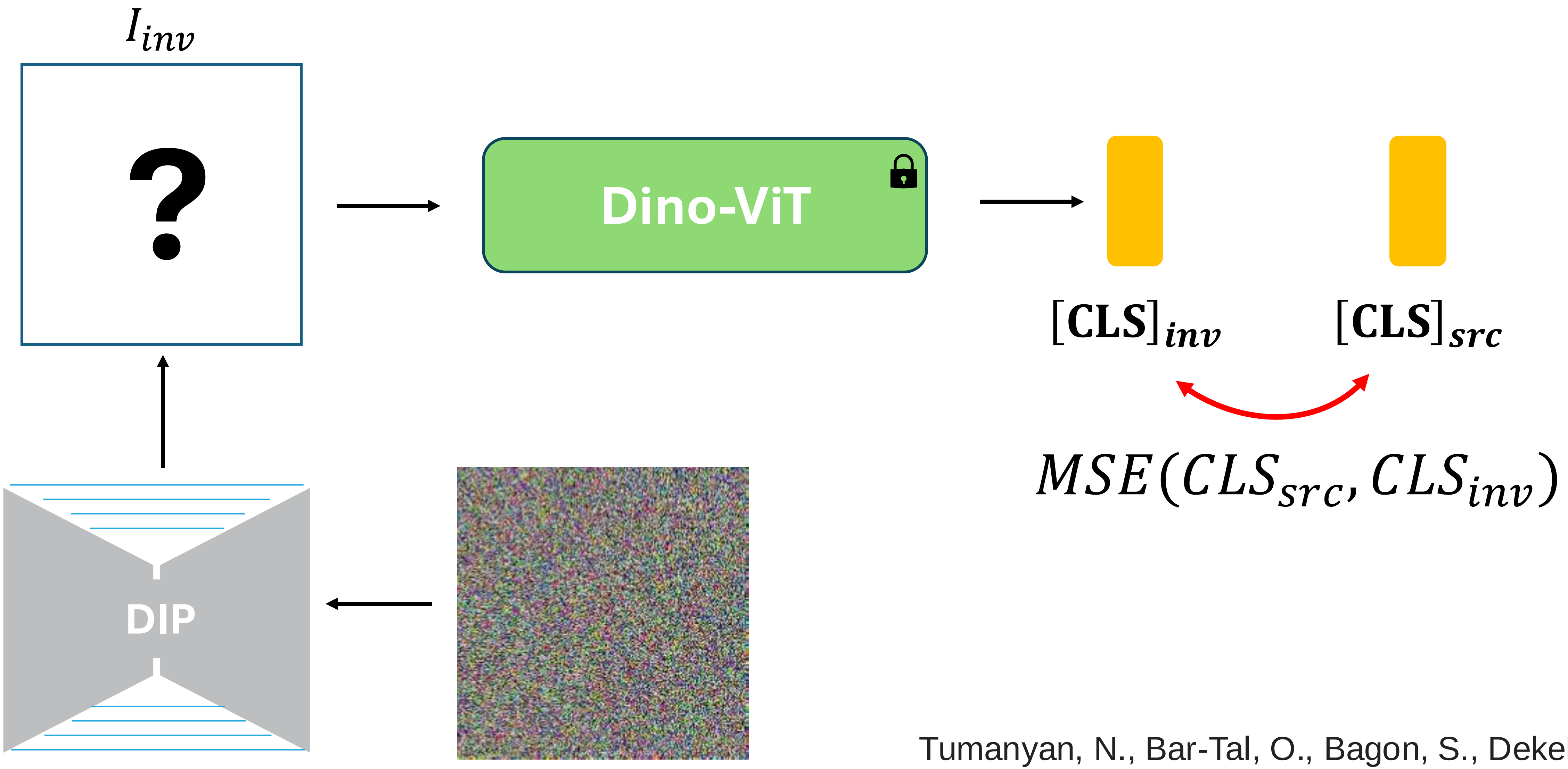
Slides from Shai Bagon

[CLS] as a **global appearance** representation

Tumanyan, N., Bar-Tal, O., Bagon, S., Dekel, T.
Splicing vit features for semantic appearance transfer (CVPR 2022)

Slides from Shai Bagon

[CLS] as a **global appearance** representation



Tumanyan, N., Bar-Tal, O., Bagon, S., Dekel, T.
Splicing vit features for semantic appearance transfer (CVPR 2022)

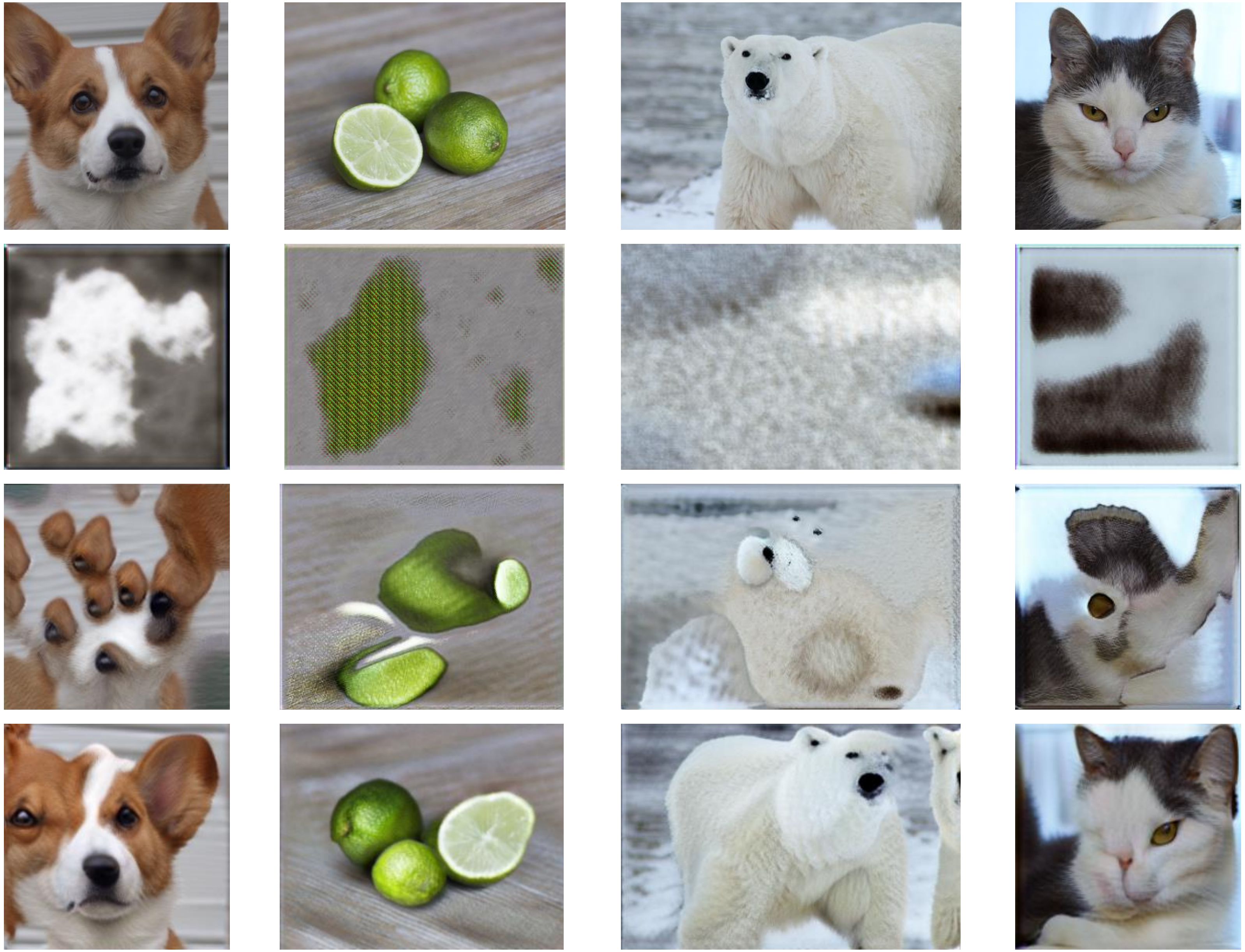
[CLS] as a **global appearance** representation

Input

layer **0**

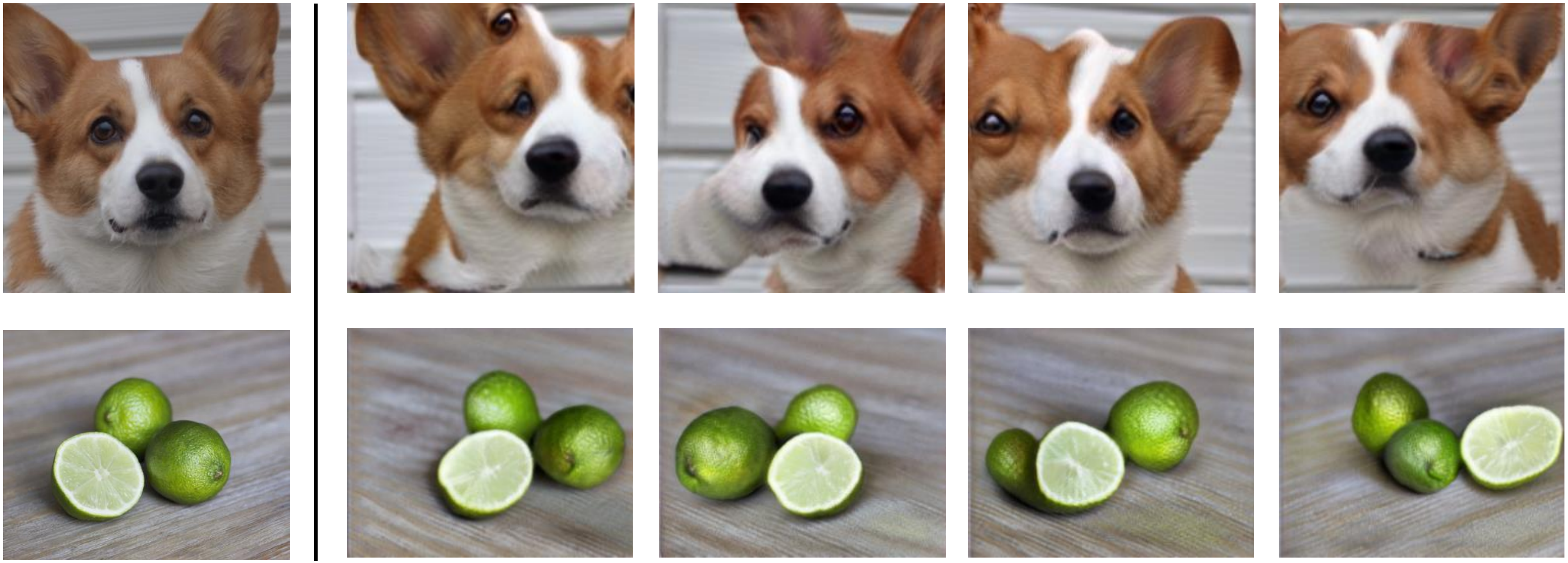
layer **3**

layer **11**



Slides from Shai Bagon

[CLS] as a **global appearance** representation



Inversion run 1 Inversion run 2 Inversion run 3 Inversion run 4

Slides from Shai Bagon