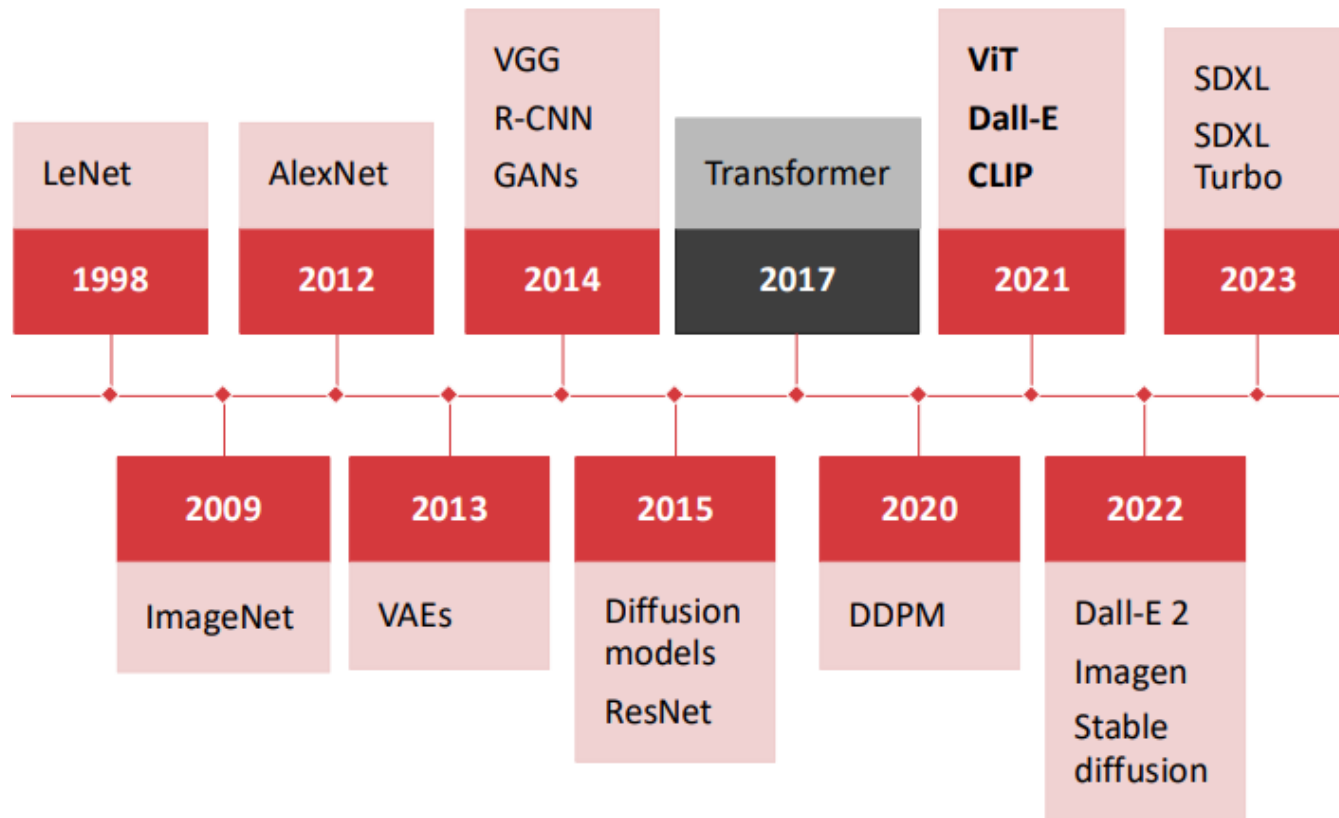


# METODE INTELIGENTE DE REZOLVARE A PROBLEMELOR REALE



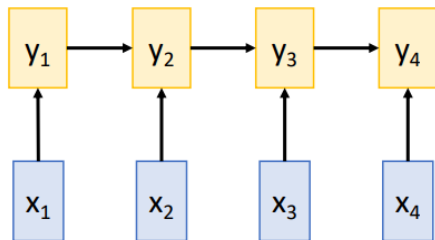
Laura Dioşan  
Vision Transformers

# Istoric al modelelor de CV



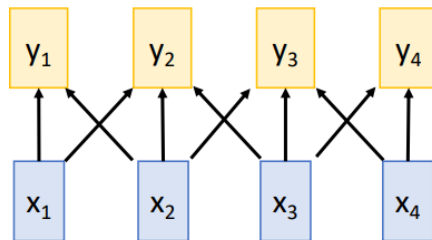
# Procesarea unei secvente

Recurrent Neural Network



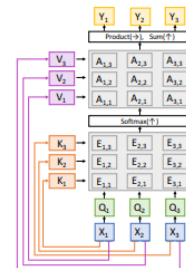
Works on **Ordered Sequences**  
(+) Good at long sequences: After one RNN layer,  $h_T$  "sees" the whole sequence  
(-) Not parallelizable: need to compute hidden states sequentially

1D Convolution



Works on **Multidimensional Grids**  
(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence  
(+) Highly parallel: Each output can be computed in parallel

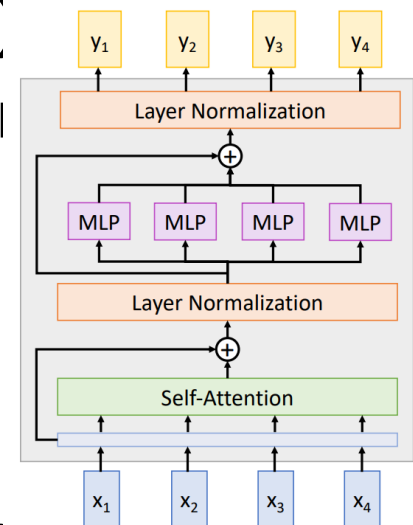
Self-Attention



Works on **Sets of Vectors**  
(-) Good at long sequences: after one self-attention layer, each output "sees" all inputs!  
(+) Highly parallel: Each output can be computed in parallel  
(-) Very memory intensive

# Transformer

- ❑ Blocurile unui transformer au
  - Input – vectori (multimi de vectori)
  - Output – vectori (multimi de vectori)
  - Hyper-parametrii:
    - ❑ # bloks
    - ❑ # heads / block
    - ❑ width
- ❑ Vectorii comunica intre ei prin mecanismul de *self-attention* al
  - Unui singur head de procesare
  - A mai multor head-uri de procesare



# Mecanismul de “atenție” (attention)

## □ Remember – embeddings

- Imagini / patch-uri izolate precum:

- Presupunem 2 atribute:

  - *caracterul tehnologic*

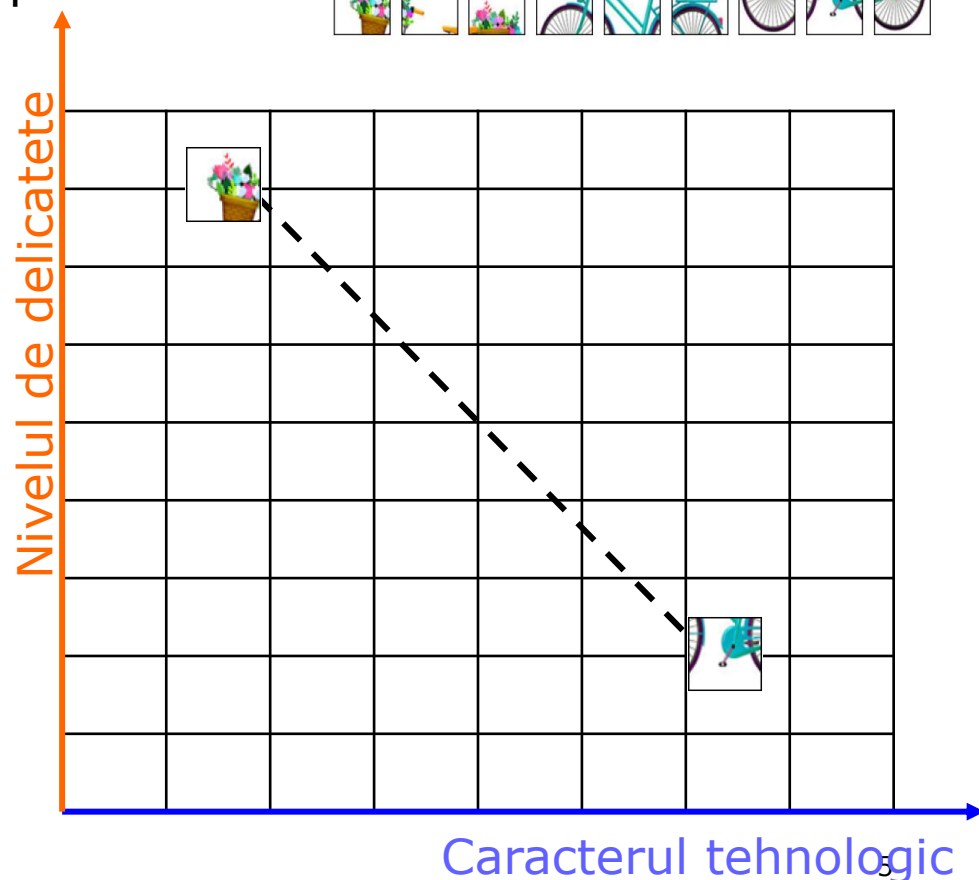
  - *nivelul de delicatete*



## □ Attention




- Contextul e ca un magnet!

- Atrage elementele care se potrivesc!



# Mecanismul de “atenție” (attention)

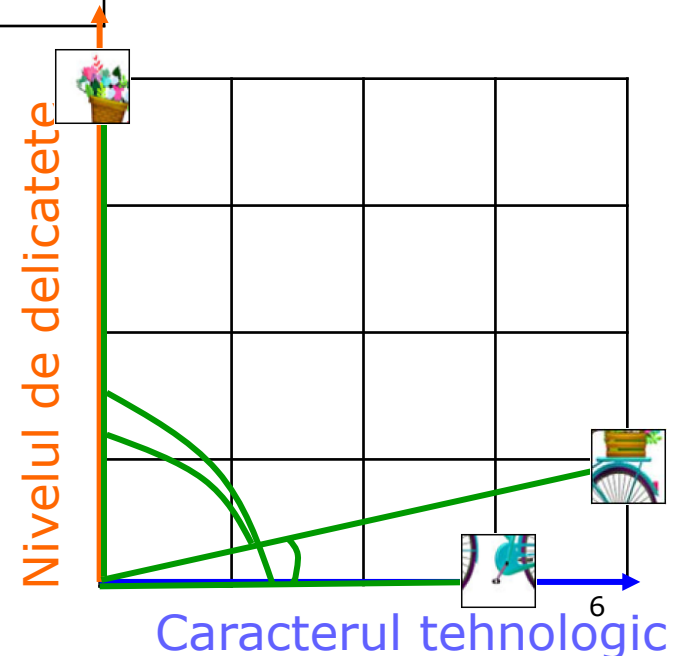
## □ Similaritatea între patch-uri

Cuvântul	<i>caracterul tehnologic</i>	<i>nivelul de delicatete</i>
	3	0
	4	1
	0	4

□  $\text{sim} \left( \begin{array}{c} \text{[patch 2]} \\ \text{[patch 1]} \end{array} \right) = 12$   
 $\text{sim}([4,1], [3,0]) =$   
 $4*3 + 1*0 = 12$

□  $\text{sim} \left( \begin{array}{c} \text{[patch 2]} \\ \text{[patch 3]} \end{array} \right) = 4$   
 $\text{sim}([4,1], [0,4]) =$   
 $4*0 + 1*4 = 4$

□  $\text{sim} \left( \begin{array}{c} \text{[patch 1]} \\ \text{[patch 3]} \end{array} \right) = 0$   
 $\text{sim}([3,0], [0,4]) =$   
 $3*0 + 0*4 = 0$



# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate

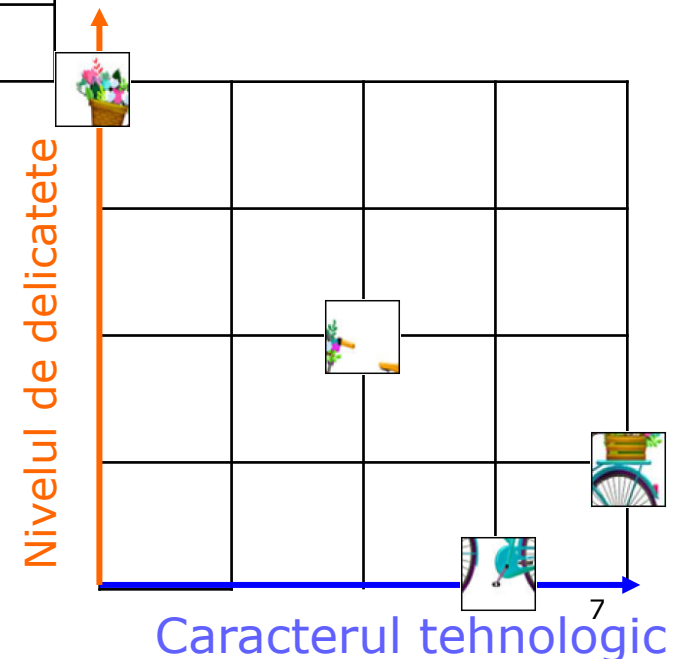


$$\square \text{sim}(\text{bird patch}, \text{bird patch}) = 1 / \sqrt{2} = 0.7$$

$$\square \text{sim}(\text{bird patch}, \text{potted plant patch}) =$$





$$\begin{aligned} \text{sim}([2,2], [0,4]) &= \\ (2*0 + 2*4) / \sqrt{2} &= \\ 8 / \sqrt{2} &= 5.65 \end{aligned}$$



# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate



						
			0.70		5.65	
			5.65		0.70	

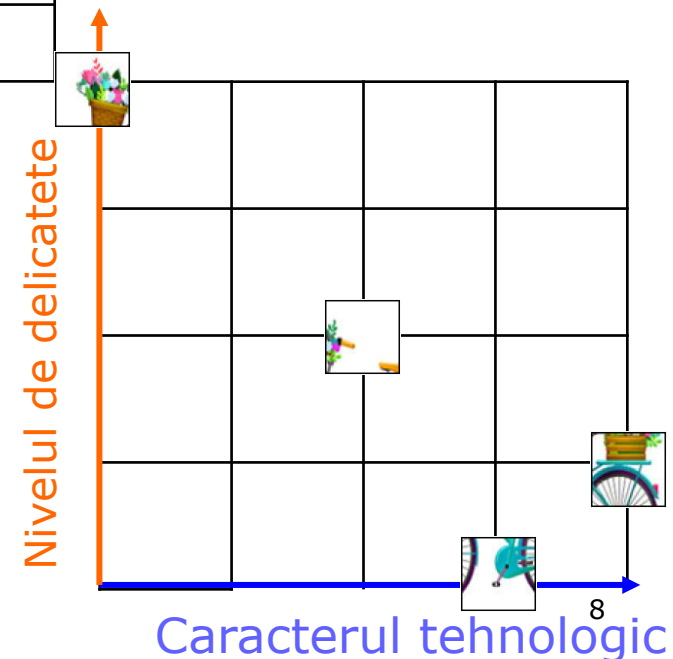
$$\square \text{sim}(\text{parrot patch}, \text{parrot patch}) = 1 / \sqrt{2} = 0.7$$

$$\square \text{sim}(\text{parrot patch}, \text{potted plant patch}) =$$

$$\text{sim}([2,2], [0,4]) =$$

$$(2*0 + 2*4) / \sqrt{2} =$$

$$8 / \sqrt{2} = 5.65$$









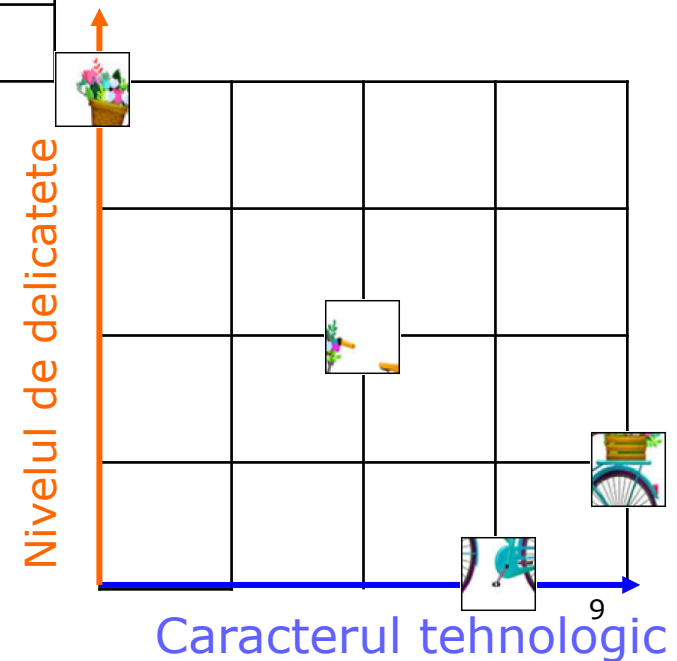
# Mecanismul de “atenție” (attention)

- Contextul patch-urilor – matricea de afinitate



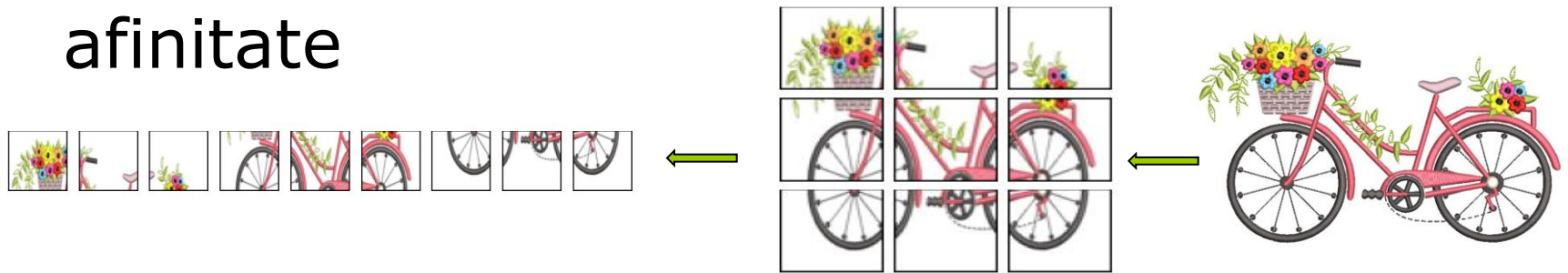
						
			0.70		5.65	
			5.65		0.70	

$$\begin{matrix} \text{parrot} \end{matrix} = 0.7 * \begin{matrix} \text{parrot} \end{matrix} + 5.65 * \begin{matrix} \text{potted plant} \end{matrix}$$



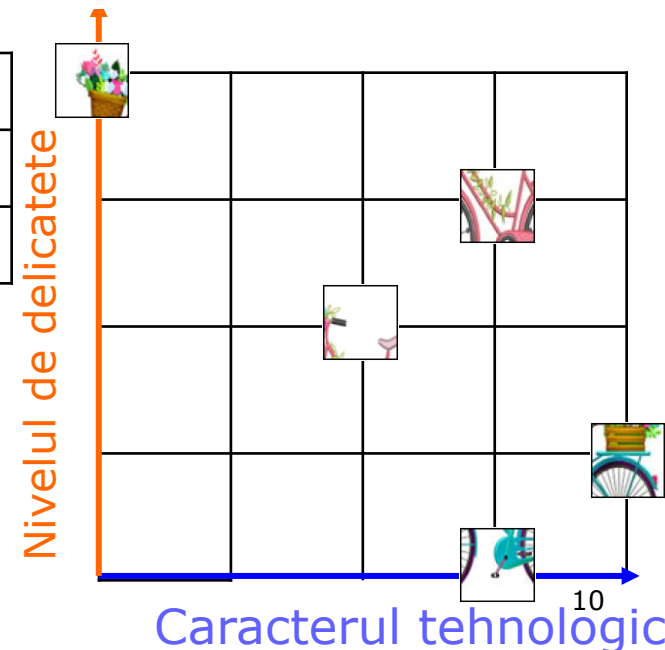
# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate



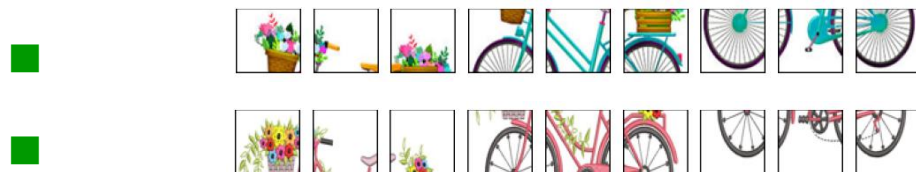
		0.70		8.48
		8.48		0.70

$$\text{Patch} = 0.7 * \text{Patch} + 8.48 * \text{Patch}$$



# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate



$$\begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} = 0.7 * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} + 5.65 * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix}$$

$$\begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} = 0.7 * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} + 8.48 * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix}$$



$$\begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} = ? * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} + ? * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix}$$

$$\begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} = ? * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix} + ? * \begin{bmatrix} \text{bird} \\ \text{bird} \end{bmatrix}$$

Normalizare – de care?

# Mecanismul de “atentie” (attention)

## Contextul patch-urilor – matricea de afinitate



$$\begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} = 0.7 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} + 5.65 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}$$

$$\begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} = 0.7 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} + 8.48 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}$$



$$\begin{bmatrix} 0.0070 \\ 0.0070 \end{bmatrix} = 0.0070 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} + 0.9929 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}$$

$$\begin{bmatrix} 0.0004 \\ 0.0004 \end{bmatrix} = 0.0004 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} + 0.9958 \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}$$

Normalizare – softmax

# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate



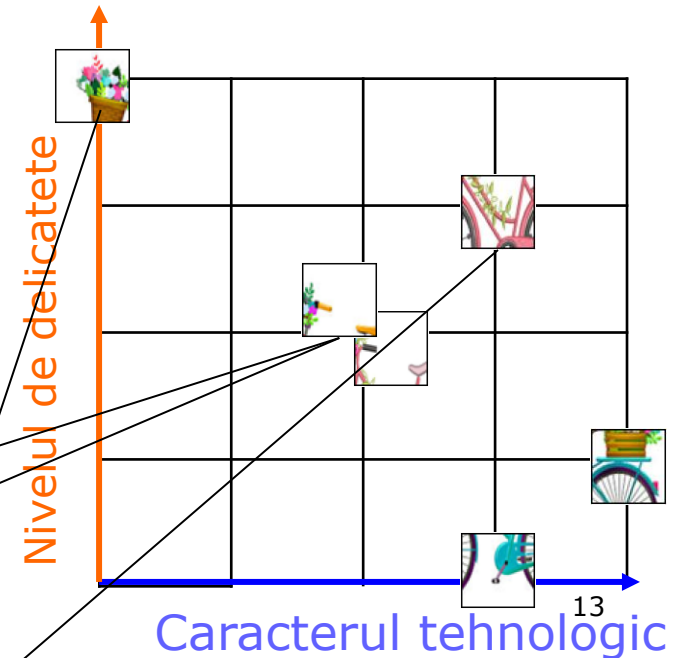
$$\begin{bmatrix} \text{flowers} \\ \text{toucan} \end{bmatrix} = 0.7 \begin{bmatrix} \text{flowers} \\ \text{toucan} \end{bmatrix} + 5.65 \begin{bmatrix} \text{flowers} \\ \text{flowers} \end{bmatrix}$$

$$\begin{bmatrix} \text{flowers} \\ \text{flowers} \end{bmatrix} = 0.7 \begin{bmatrix} \text{flowers} \\ \text{flowers} \end{bmatrix} + 8.48 \begin{bmatrix} \text{flowers} \\ \text{flowers} \end{bmatrix}$$



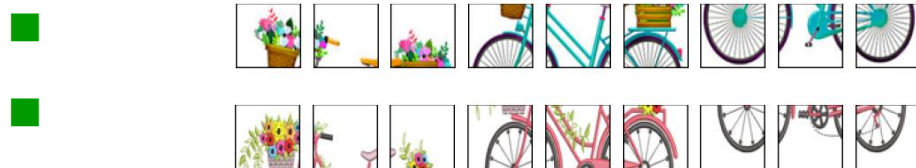
$$\begin{bmatrix} \text{flowers} \\ \text{flowers} \end{bmatrix} = 0.0070 \begin{bmatrix} 2,2 \end{bmatrix} + 0.9929 \begin{bmatrix} 0,4 \end{bmatrix}$$

$$\begin{bmatrix} \text{flowers} \\ \text{flowers} \end{bmatrix} = 0.0004 \begin{bmatrix} 2,2 \end{bmatrix} + 0.9958 \begin{bmatrix} 3,3 \end{bmatrix}$$



# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate

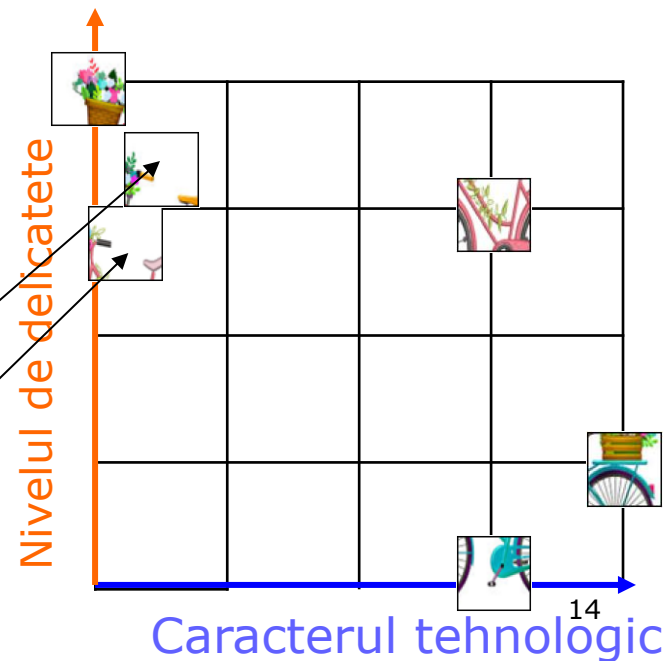


$$\begin{aligned} \text{Patch 1} &= 0.7 * \text{Patch 1} + 5.65 * \text{Patch 2} \\ \text{Patch 2} &= 0.7 * \text{Patch 2} + 8.48 * \text{Patch 3} \end{aligned}$$



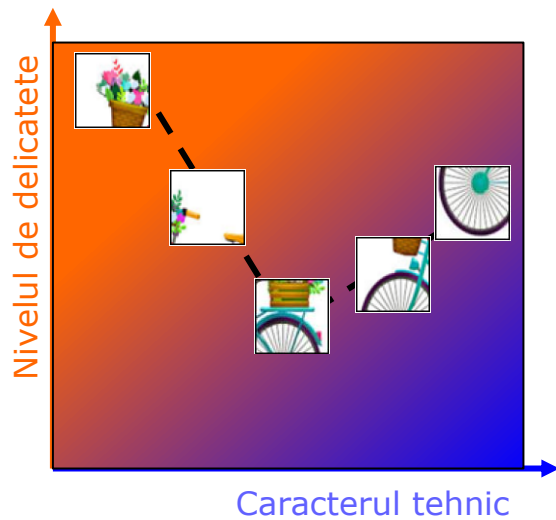
$$\text{Patch 1} = [0.1400, 3.9850]$$

$$\text{Patch 2} = [0.0008, 3.9840]$$

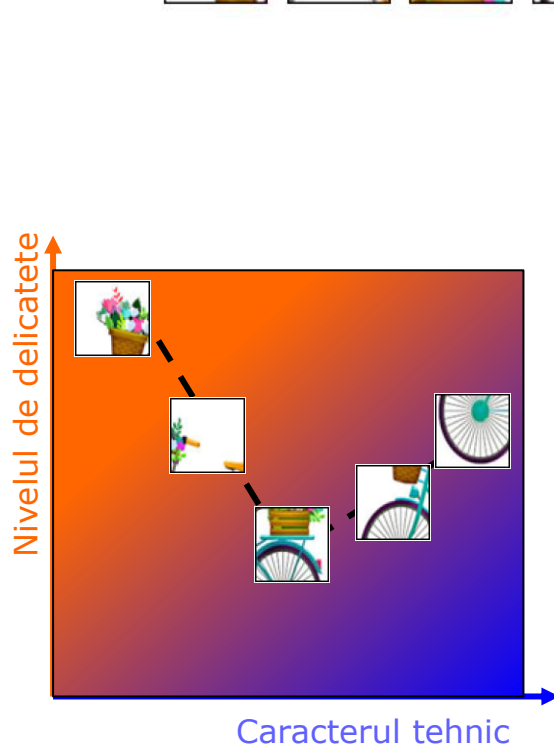


14

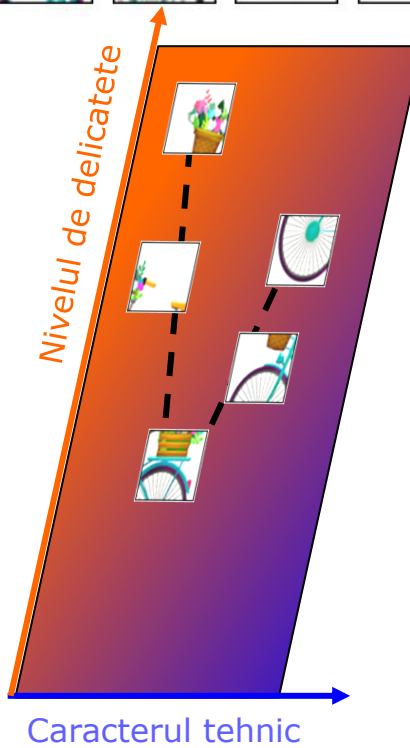
# Mecanismul de “atenție” (attention)



# Mecanismul de “atenție” (attention)



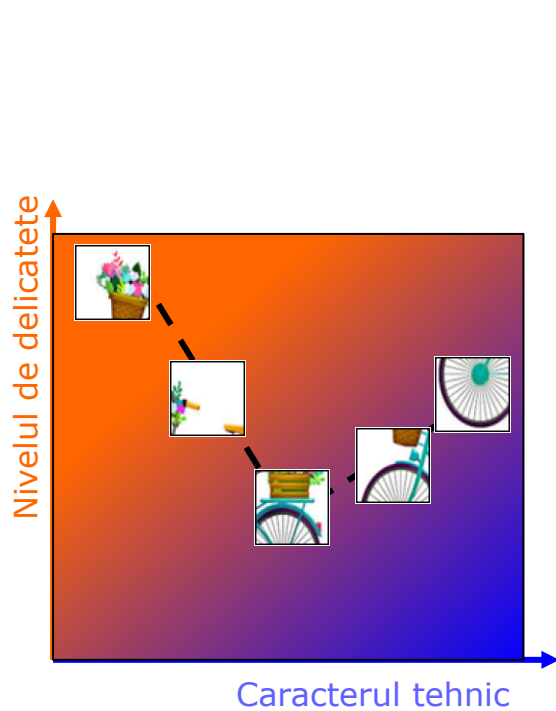
a) reprez1



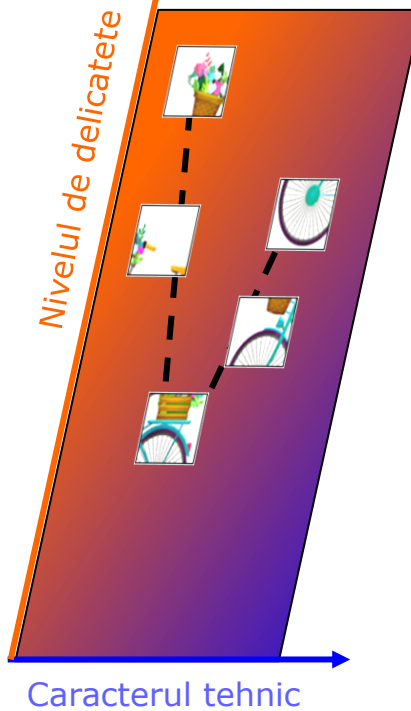
b) reprez2



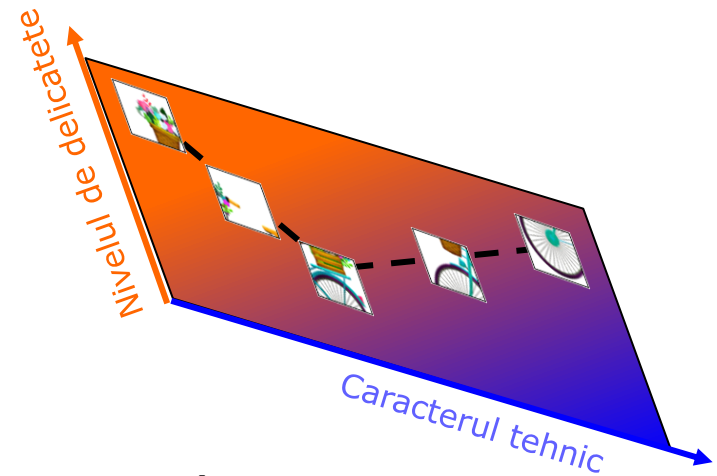
# Mecanismul de “atenție” (attention)



a) reprez1

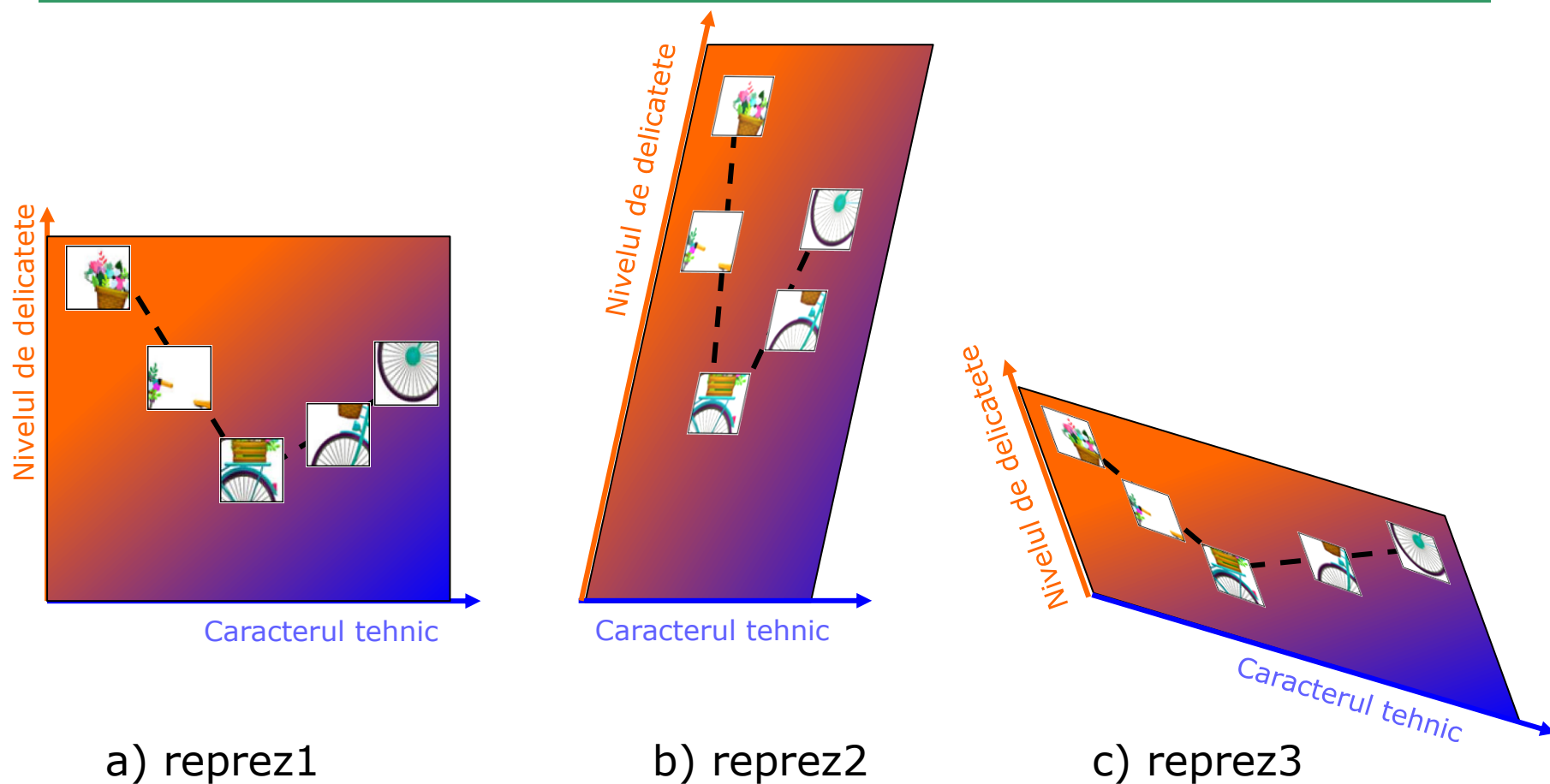


b) reprez2



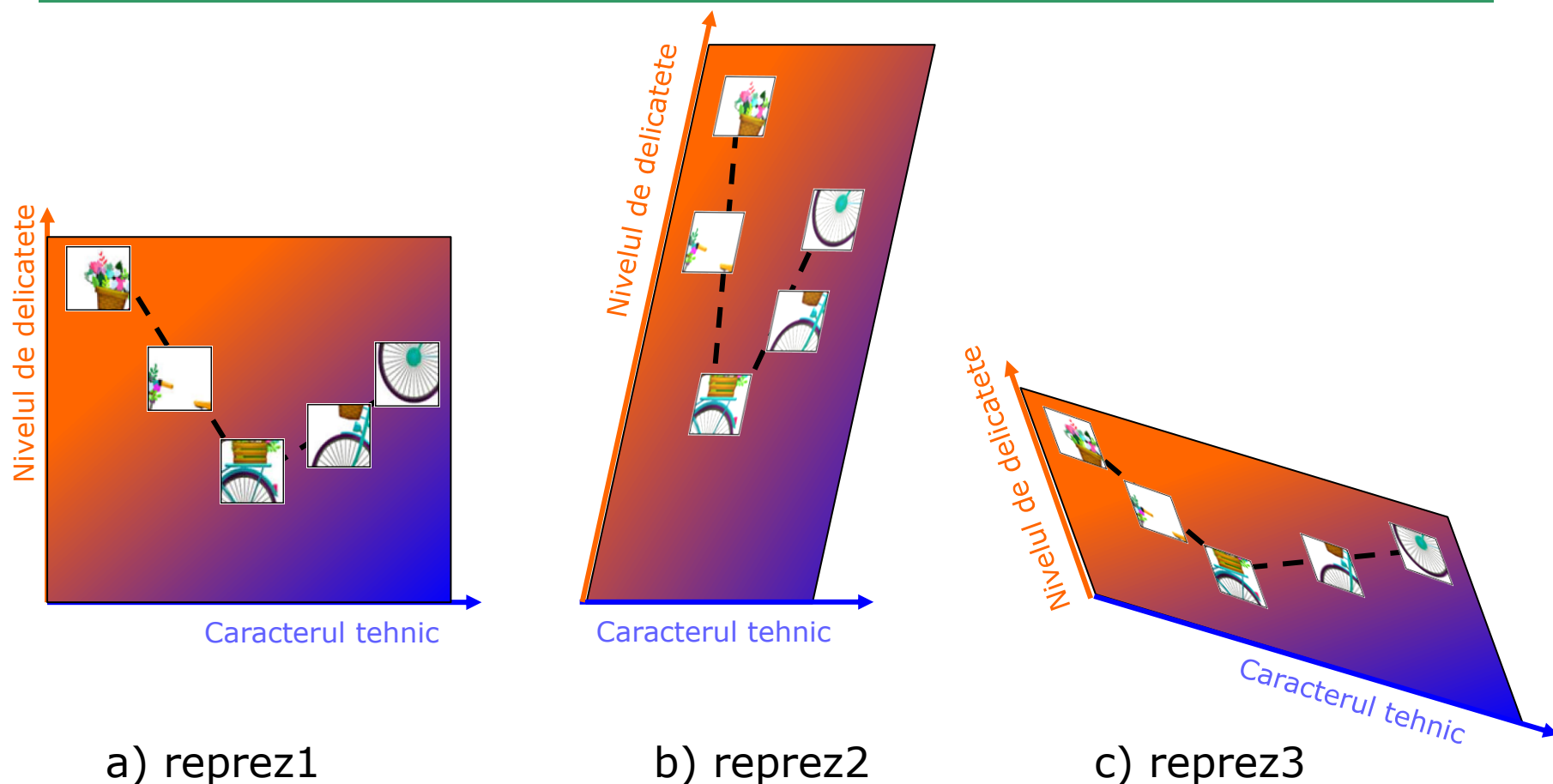
c) reprez3

# Mecanismul de “atenție” (attention)



**Q1: reprez1 = f(reprez2) = g(reprez3) ???**  
**Da!!! Transformare liniară!!!**

# Mecanismul de “atenție” (attention)



a) reprez1

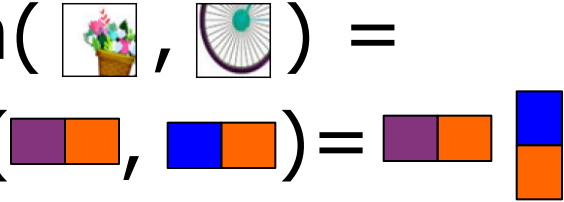
b) reprez2

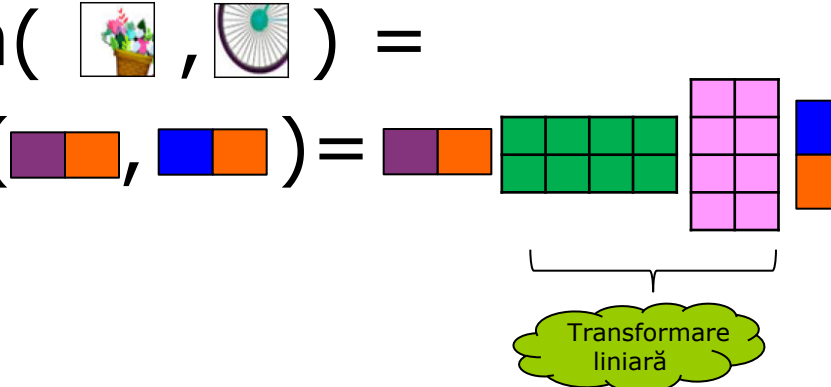
c) reprez3

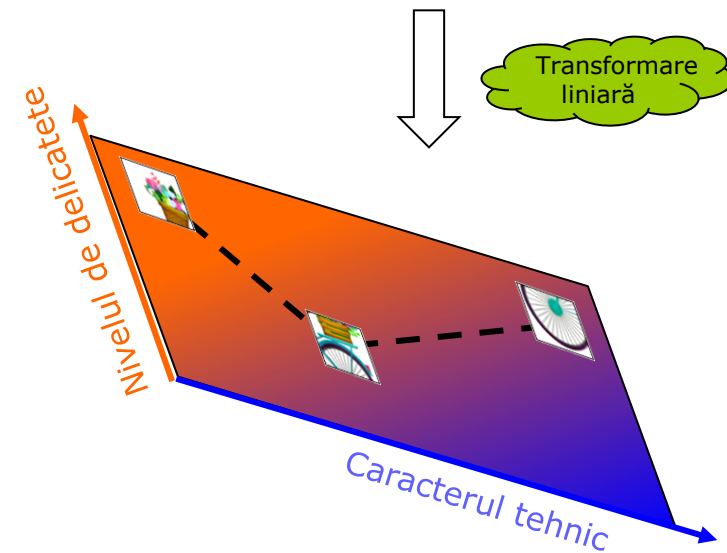
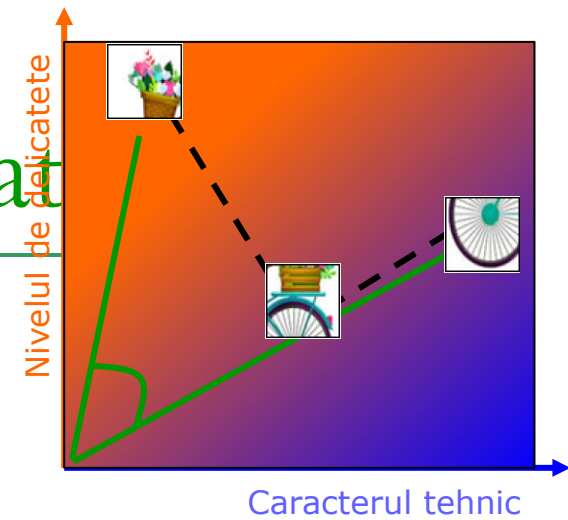
**Q2: Care reprezentare e mai bună pentru a stabili similaritatea (deci a face diferența între caracteristicile patch-ului  )?**

**Reprezentarea 1 sau 3!**

# Mecanismul de “atenție” (at

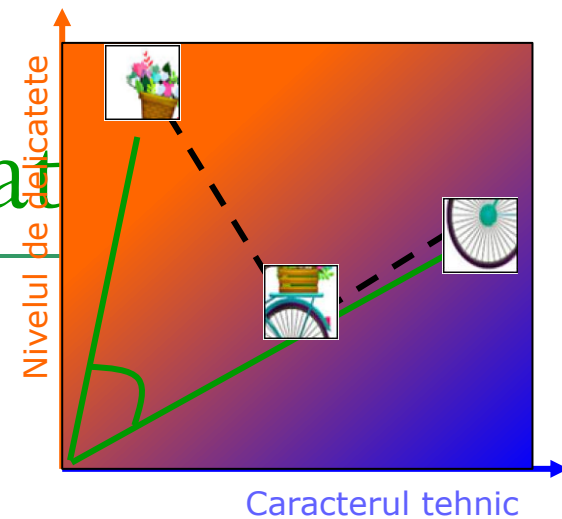
$$\square \text{ Sim}(\text{img}_1, \text{img}_2) = \text{sim}(\text{vec}_1, \text{vec}_2) = \text{vec}_3$$


$$\square \text{ Sim}(\text{img}_1, \text{img}_2) = \text{sim}(\text{vec}_1, \text{vec}_2) = \text{vec}_3$$




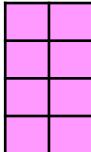
# Mecanismul de “atenție” (at

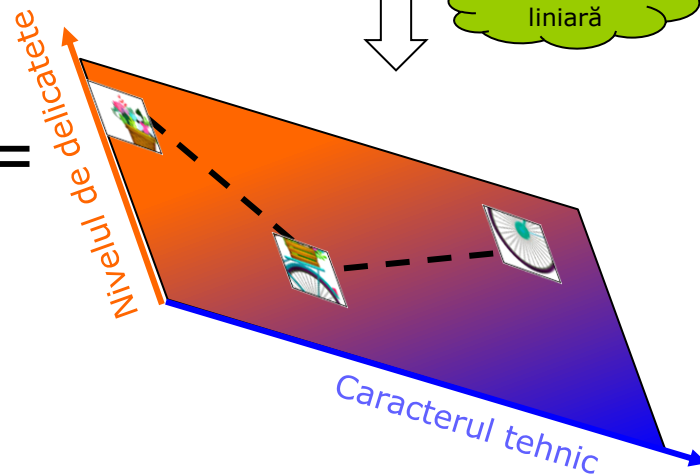
$$\square \text{sim}(\text{img1}, \text{img2}) = \text{sim}(\text{vec1}, \text{vec2}) = \text{vec3}$$



$$\square \text{sim}(\text{img1}, \text{img2}) = \text{sim}(\text{vec1}, \text{vec2}) = \text{vec3}$$

K (Key) =  

Q (Queries) = 



Valorile optime - ANN

# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate



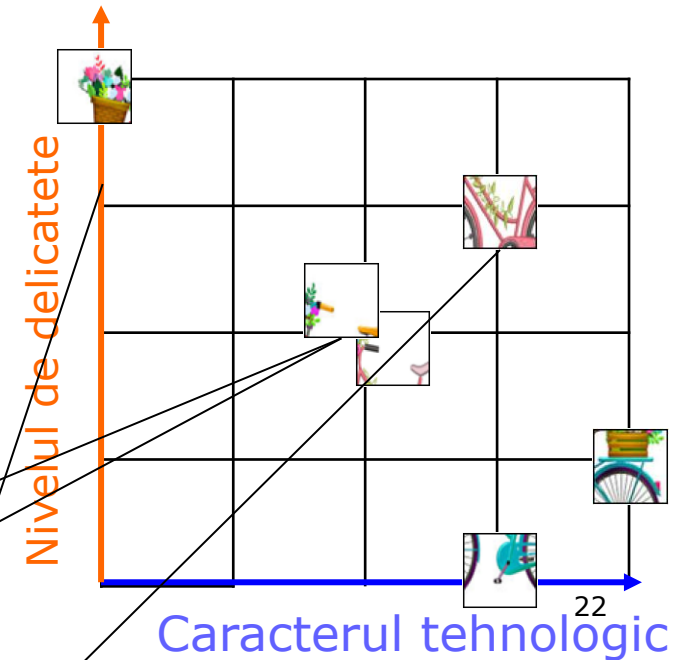
$$\begin{bmatrix} \text{bird} \end{bmatrix} = 0.7 \begin{bmatrix} \text{bird} \end{bmatrix} + 5.65 \begin{bmatrix} \text{flower pot} \end{bmatrix}$$

$$\begin{bmatrix} \text{birdhouse} \end{bmatrix} = 0.7 \begin{bmatrix} \text{birdhouse} \end{bmatrix} + 8.48 \begin{bmatrix} \text{flower pot} \end{bmatrix}$$



$$\begin{bmatrix} \text{bird} \end{bmatrix} = 0.0070 * [2,2] + 0.9929 * [0,4] = [0.1400, 3.9850]$$

$$\begin{bmatrix} \text{birdhouse} \end{bmatrix} = 0.0004 * [2,2] + 0.9958 * [3,3] = [0.0008, 3.9840]$$



# Mecanismul de “atenție” (attention)

## Contextul patch-urilor – matricea de afinitate



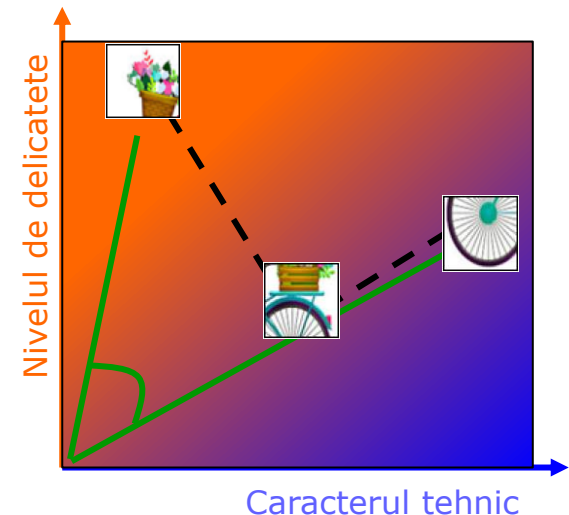
$$\begin{bmatrix} \text{toucan} \end{bmatrix} = 0.7 \begin{bmatrix} \text{toucan} \end{bmatrix} + 5.65 \begin{bmatrix} \text{flowers} \end{bmatrix}$$

$$\begin{bmatrix} \text{bird} \end{bmatrix} = 0.7 \begin{bmatrix} \text{bird} \end{bmatrix} + 8.48 \begin{bmatrix} \text{bicycle} \end{bmatrix}$$



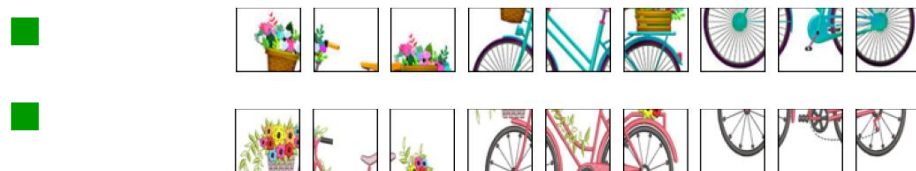
$$\begin{bmatrix} \text{toucan} \end{bmatrix} = 0.0070 * [2,2] + 0.9929 * [0,4] = [0.1400, 3.9850]$$

$$\begin{bmatrix} \text{bird} \end{bmatrix} = 0.0004 * [2,2] + 0.9958 * [3,3] = [0.0008, 3.9840]$$



# Mecanismul de “atentie” (attention)

## Contextul patch-urilor – matricea de afinitate



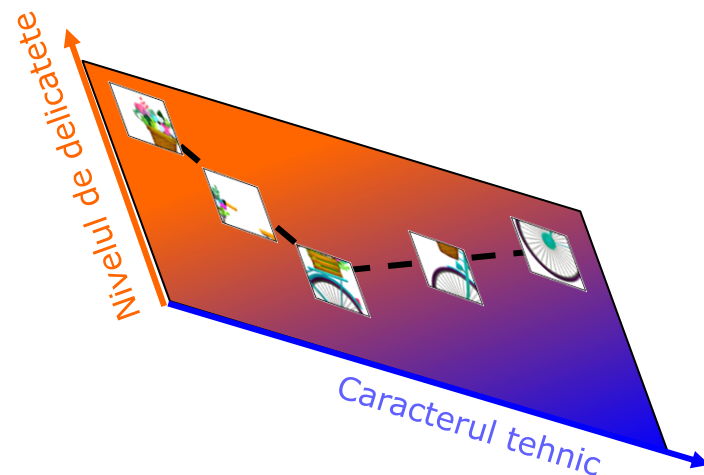
$$\begin{bmatrix} 0.7 & 0.3 \end{bmatrix} = 0.7 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} + 5.65 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix}$$

$$\begin{bmatrix} 0.7 & 0.3 \end{bmatrix} = 0.7 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} + 8.48 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix}$$



$$\begin{bmatrix} 0.7 & 0.3 \end{bmatrix} = 0.0070 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} + 0.9929 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix}$$

$$\begin{bmatrix} 0.7 & 0.3 \end{bmatrix} = 0.0004 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} + 0.9958 \begin{bmatrix} 0.7 & 0.3 \end{bmatrix}$$



in spatiul proiectiei  
definite de K si Q










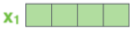
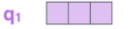
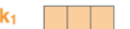



# Mecanismul de “atentie” (attention)

- Input - vectori de dimensiune (N, #patches x #patches + 1, De)
- Output - vectori de dimensiune (N, #patches x #patches + 1, D)

$$\text{softmax} \left( \frac{\begin{matrix} \text{Q} \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix}}{\sqrt{d_k}} \right) \begin{matrix} \text{V} \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix} \\ = \begin{matrix} \text{Z} \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix}$$

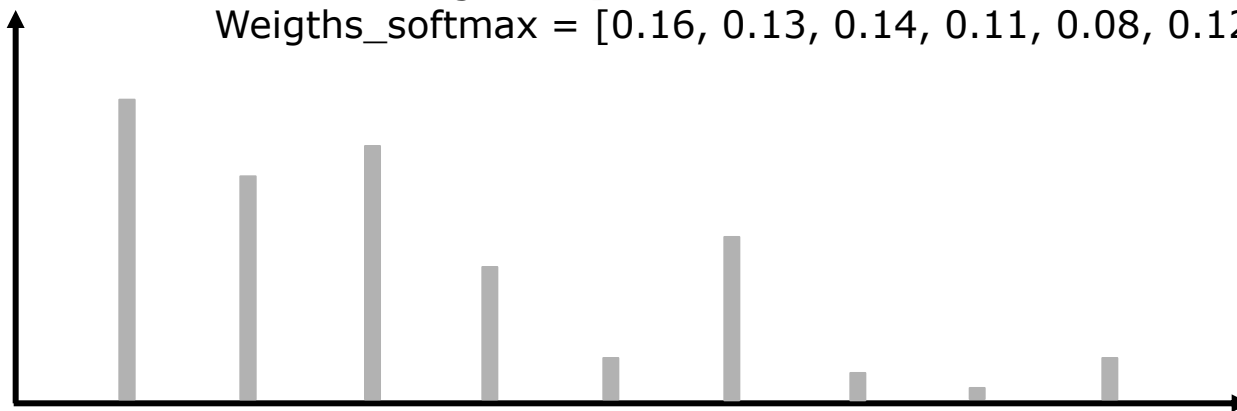
# Mecanismul de “atentie” (attention)

Input									
representation	$x_1$ 								
Queries	$q_1$ 								
Keys	$k_1$ 								
Values	$v_1$ 								
Similarities	$q_1 \cdot k_1$	$q_1 \cdot k_2$	$q_1 \cdot k_3$	$q_1 \cdot k_4$	$q_1 \cdot k_5$	$q_1 \cdot k_6$	$q_1 \cdot k_7$	$q_1 \cdot k_8$	$q_1 \cdot k_9$
Scores $s = [s_1, s_2, \dots, s_9]$	$\frac{q_1 \cdot k_1}{\sqrt{D}}$	$\frac{q_1 \cdot k_2}{\sqrt{D}}$	$\frac{q_1 \cdot k_3}{\sqrt{D}}$	$\frac{q_1 \cdot k_4}{\sqrt{D}}$	$\frac{q_1 \cdot k_5}{\sqrt{D}}$	$\frac{q_1 \cdot k_6}{\sqrt{D}}$	$\frac{q_1 \cdot k_7}{\sqrt{D}}$	$\frac{q_1 \cdot k_8}{\sqrt{D}}$	$\frac{q_1 \cdot k_9}{\sqrt{D}}$
$w = \text{Softmax}(s)$									
Weighting the values	$Z_1 = w_1 \cdot v_1 + w_2 \cdot v_2 + \dots + w_9 \cdot v_9$								

# Mecanismul de “atentie” (attention)

Weights

Weights = [0.90, 0.70, 0.80, 0.50, 0.20, 0.60, 0.15, 0.10, 0.20]  
Weights\_softmax = [0.16, 0.13, 0.14, 0.11, 0.08, 0.12, 0.078, 0.074, 0.08]



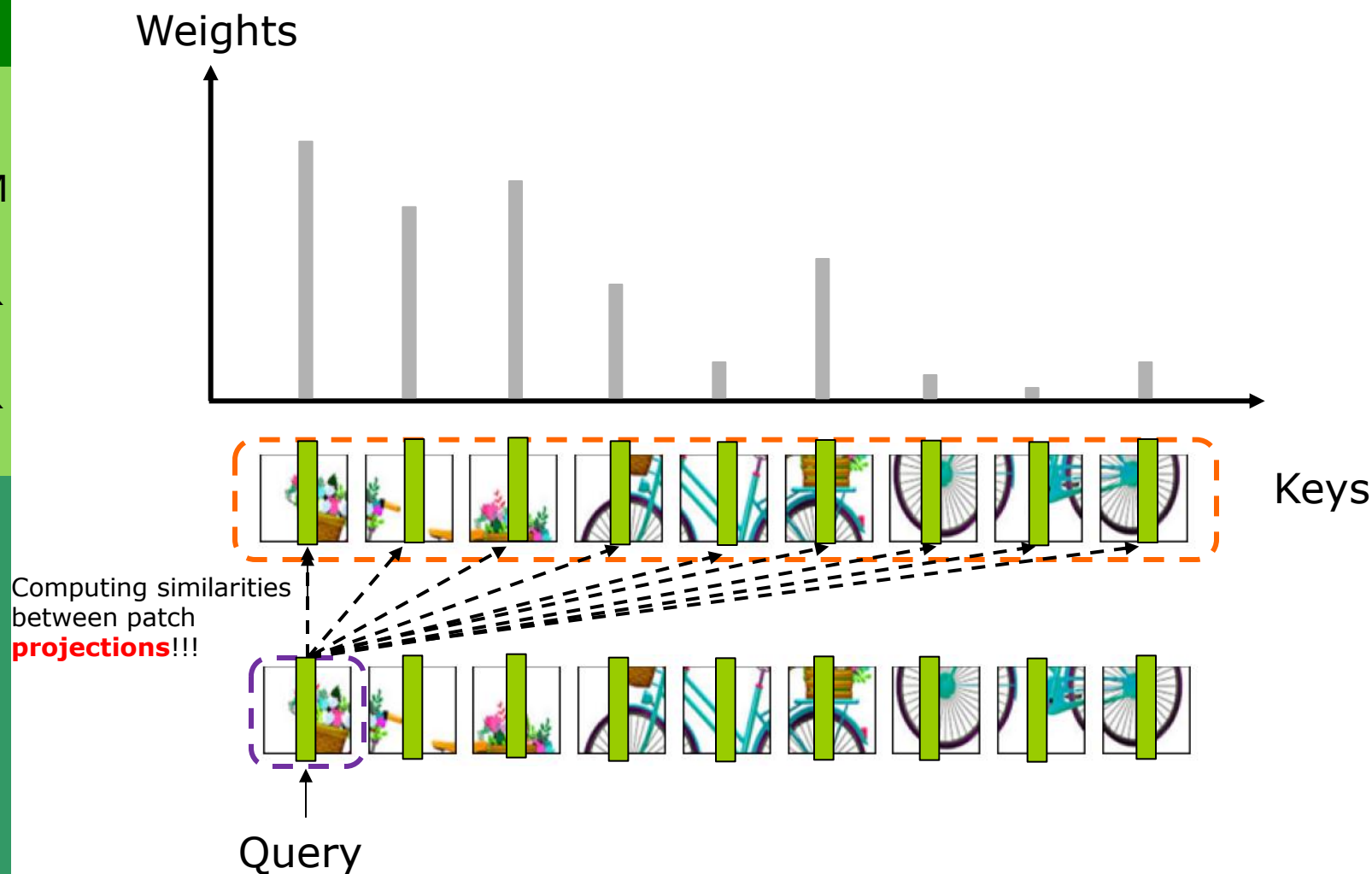
Keys

Computing similarities  
between patch  
**projections!!!**

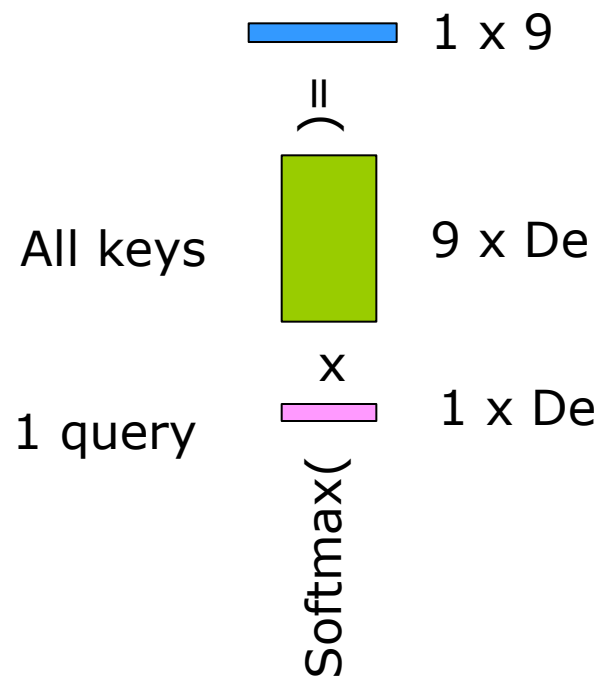
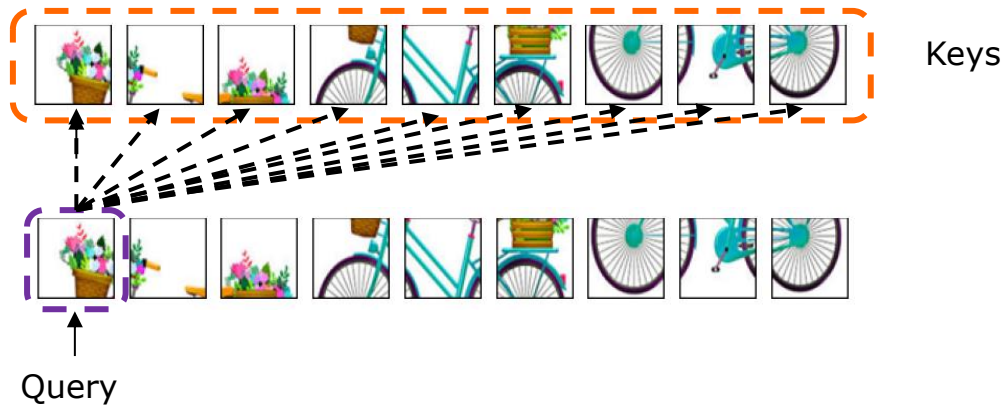
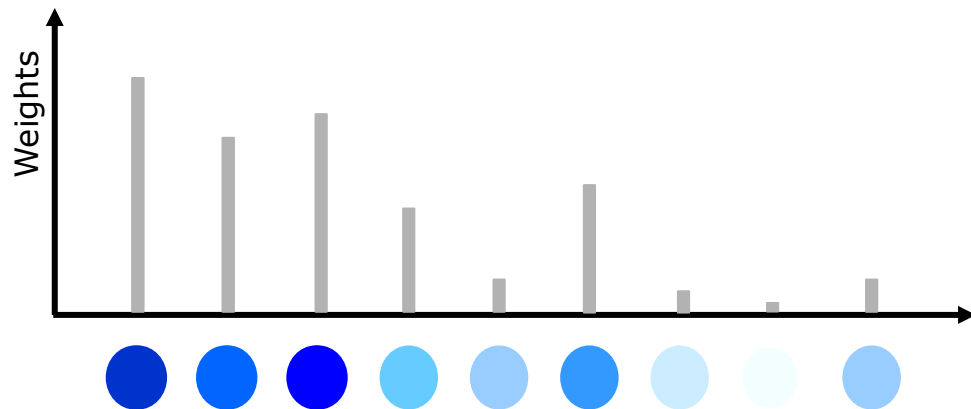


Query

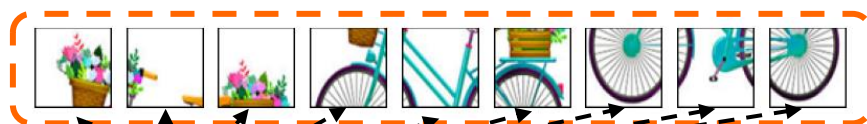
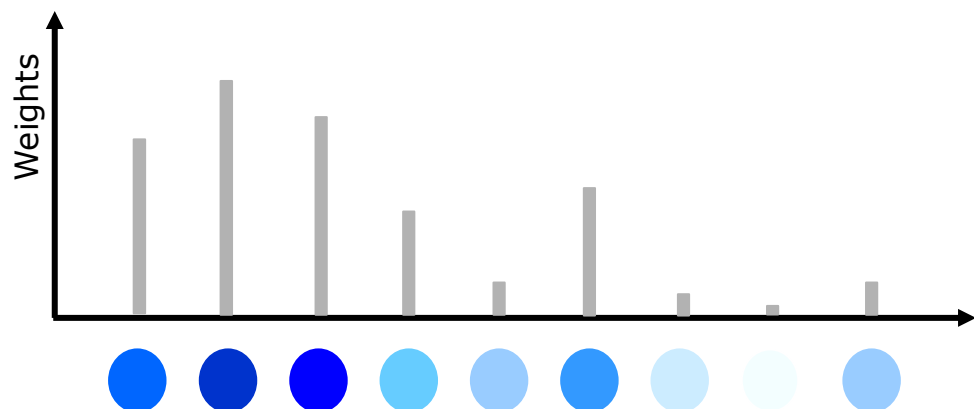
# Mecanismul de “atentie” (attention)



# Mecanismul de “atentie” (attention)



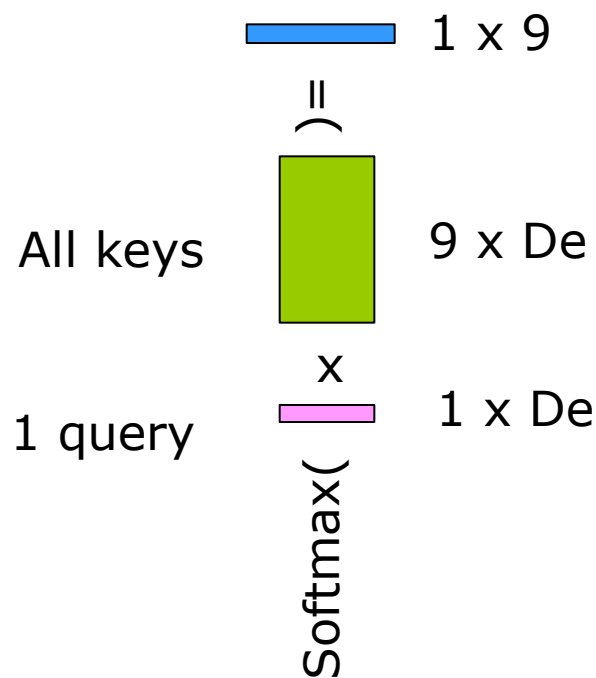
# Mecanismul de “atentie” (attention)



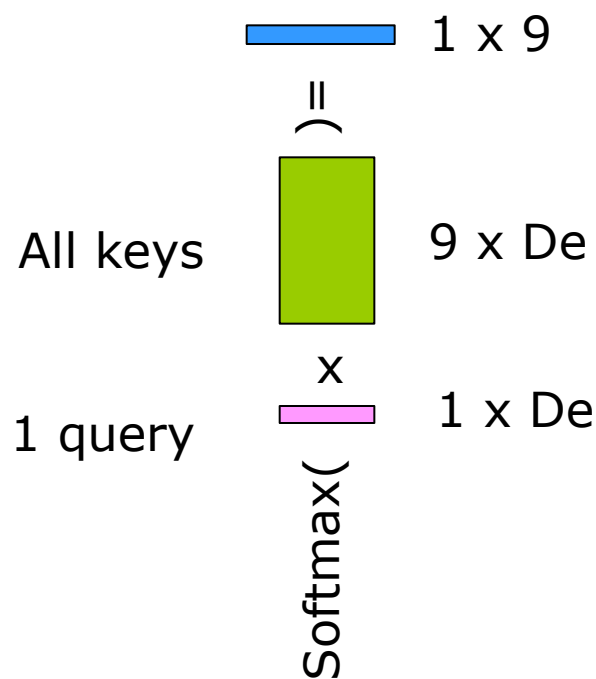
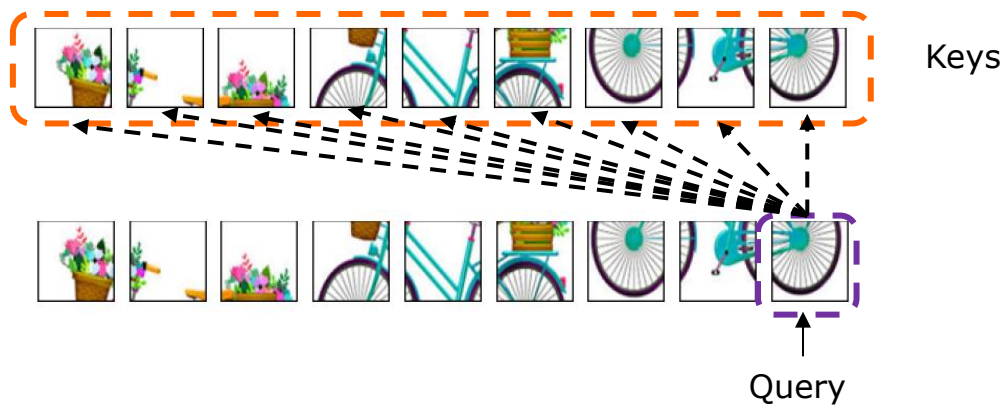
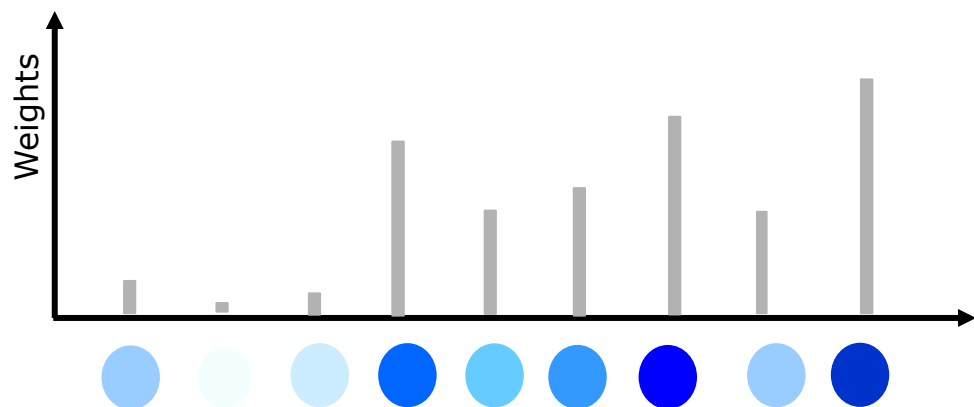
Keys



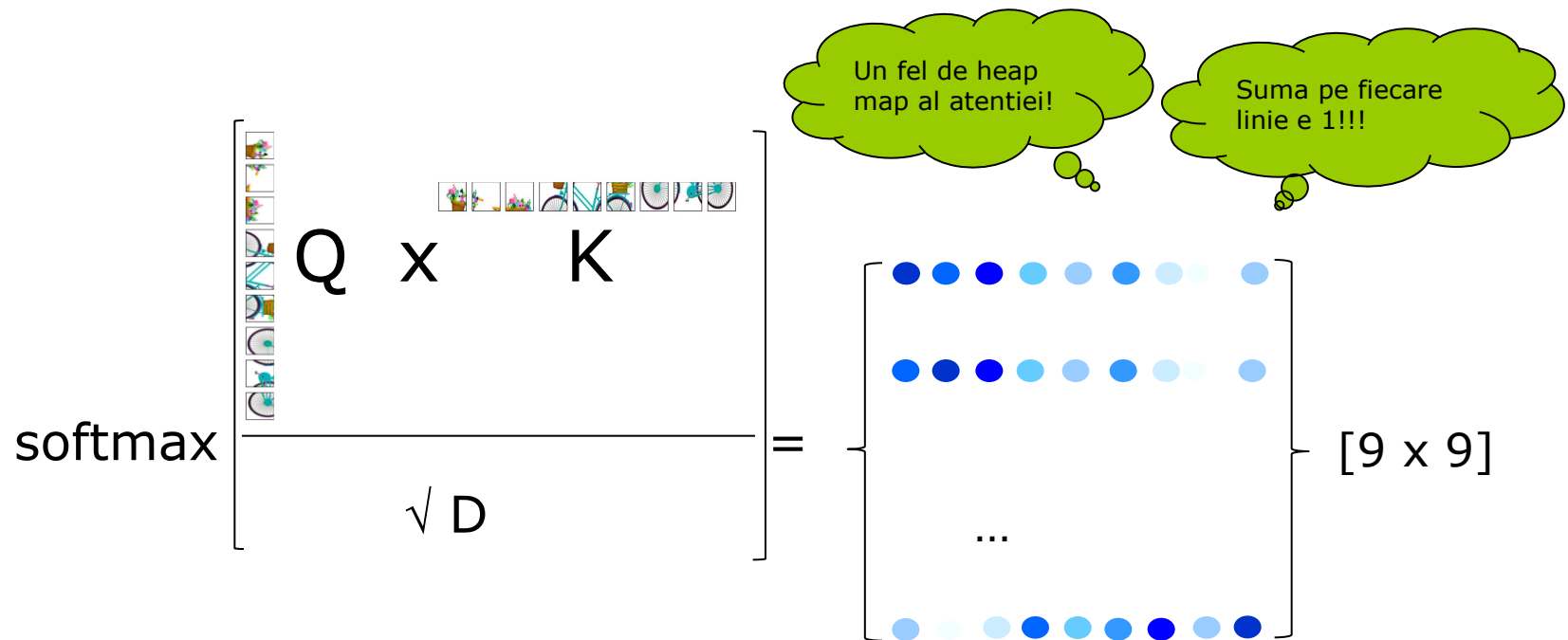
Query



# Mecanismul de “atenție” (attention)

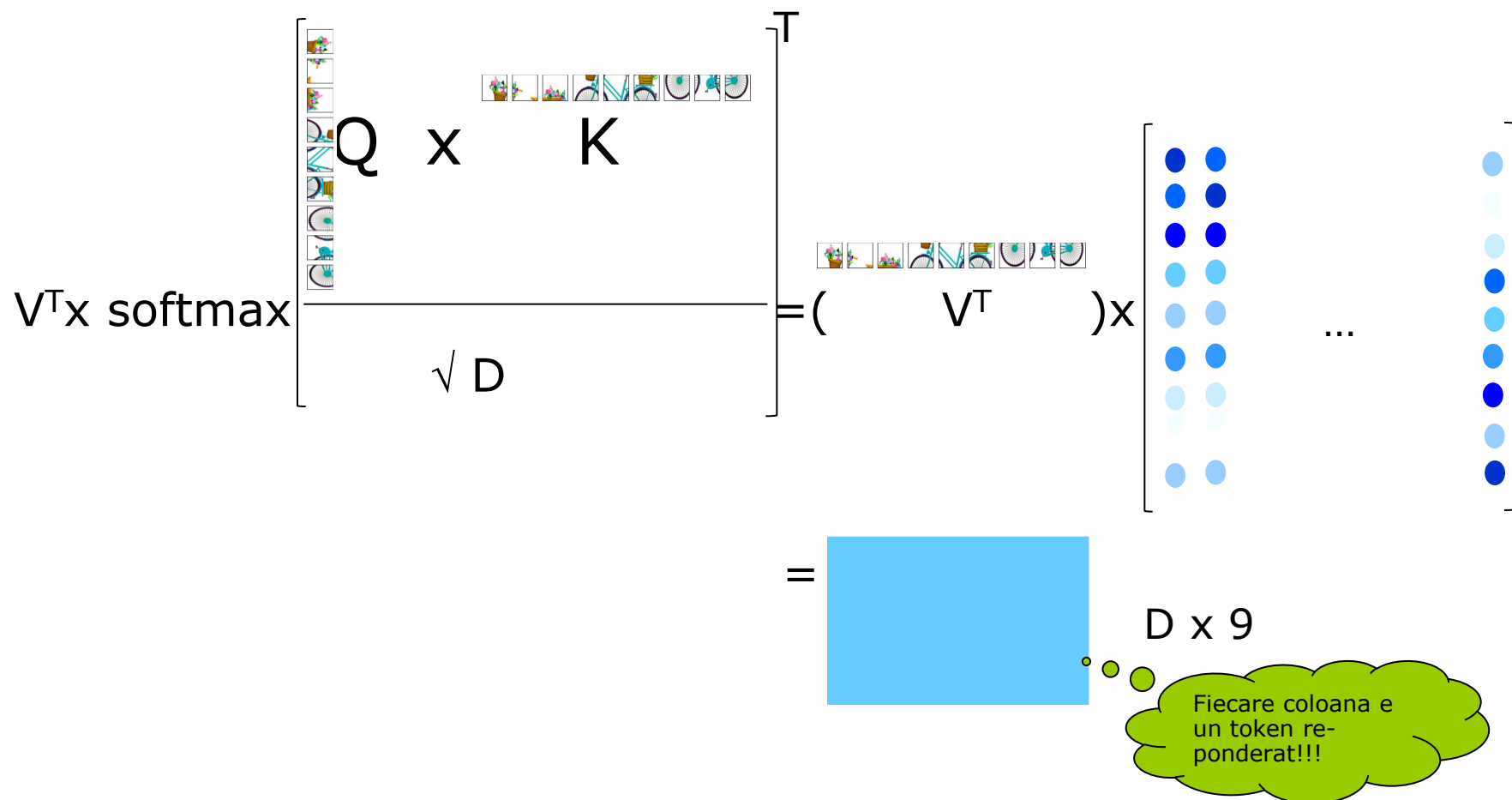


# Mecanismul de “atentie” (attention)

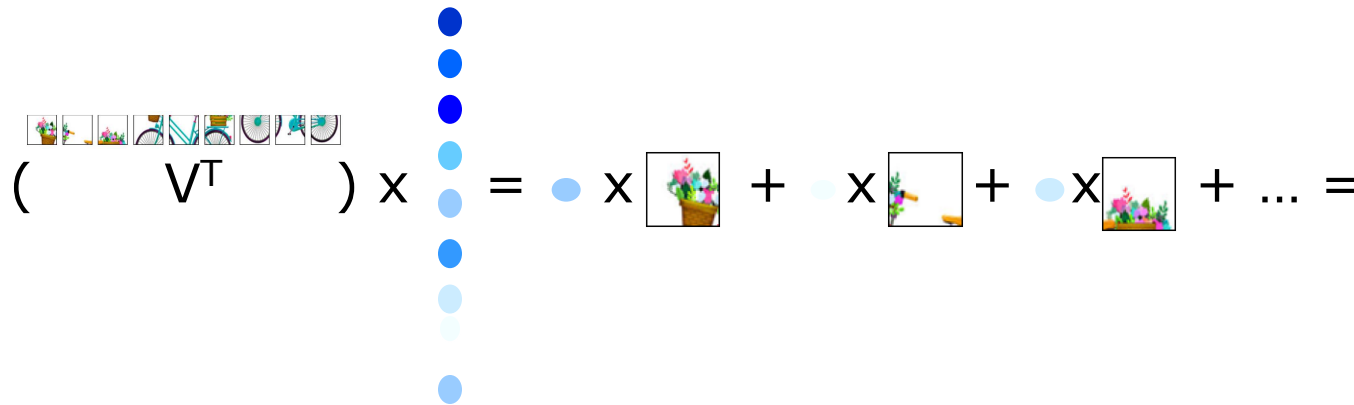




# Mecanismul de “atentie” (attention)



# Mecanismul de “atentie” (attention)


$$\begin{pmatrix} \text{flowers} & \text{bananas} & \text{wheels} & \dots \end{pmatrix} V^T \times \begin{pmatrix} \text{blue} \\ \text{blue} \\ \text{blue} \\ \text{light blue} \\ \text{light blue} \\ \text{light blue} \\ \text{light blue} \\ \text{light blue} \\ \text{light blue} \\ \text{light blue} \end{pmatrix} = \text{blue} \times \text{flowers} + \text{light blue} \times \text{bananas} + \text{medium blue} \times \text{flowers} + \dots =$$

Suma valorilor  
re-ponderate

M  
I  
R  
P  
R[illegible]

# Cum se poate integra mecanismul de atentie intr-o retea neuronală pentru procesarea imaginilor?

## ■ V2: Inlocuirea convolutiei cu o "atentie locala" ("local relation")

### ■ Cum?

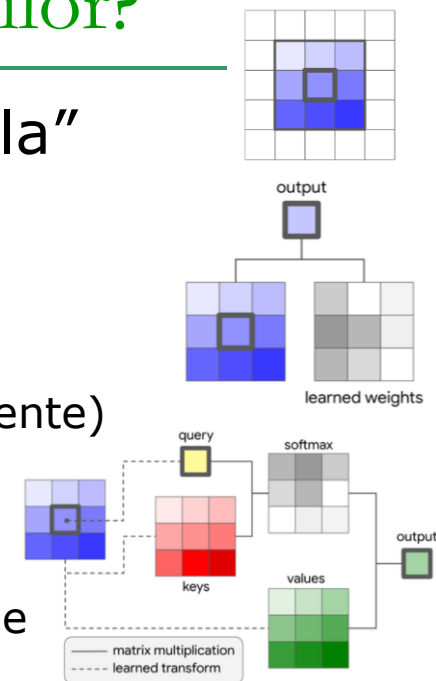
- Convolutia = produs scalar intre filtru si patch
- Local attention
  - Centrul patch-ului -> query (vector cu  $D_q$  elemente)
  - Fiecare element din patch ->
    - Keys (vector  $R \times R \times D_k$ )
    - Values (vector  $R \times R \times C'$ )
  - Outputul e calculate cu ajutorul mecanismului de atentie

### ■ Pro

- Numar mai mic de parametrii (in [Hu et al.]: ResNet-50 are  $25.5 \times 10^6$  param, iar LR-Net-50 are  $23.3 \times 10^6$  params)

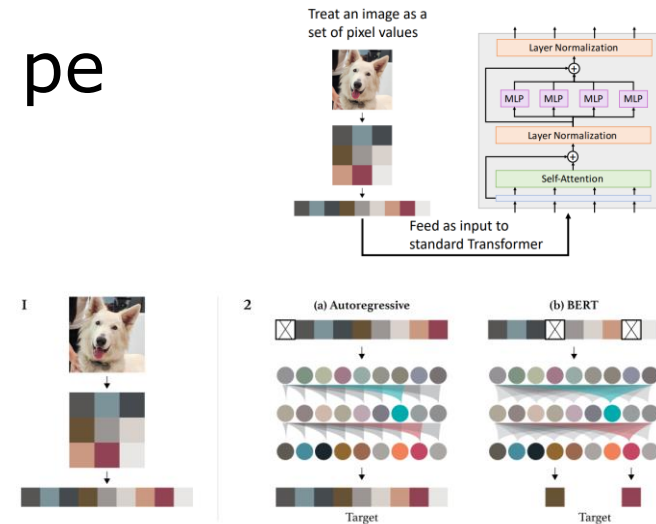
### ■ Contra

- Implementare dificila (multe detalii tricky)
- Doar putin mai buna ca ResNet



# Cum se poate integra mecanismul de atentie intr-o retea neuronală pentru procesarea imaginilor?

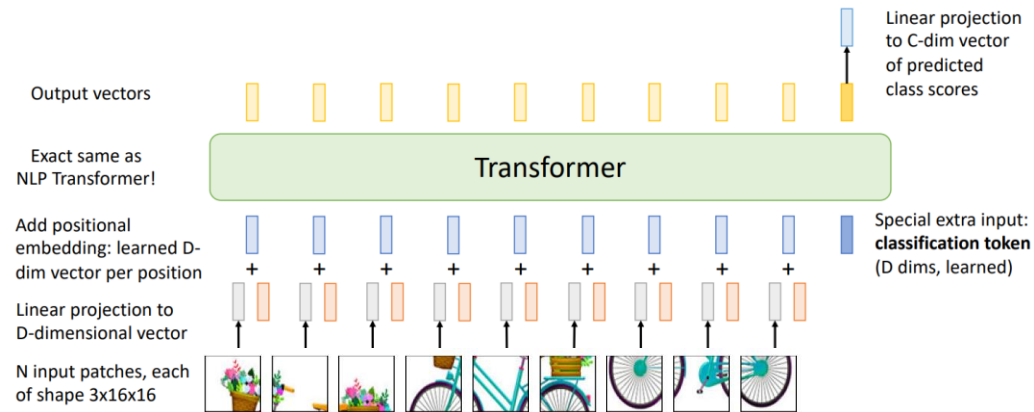
- ❑ V3: transformer aplicat direct pe pixeli
  - Cum?
    - ❑ Input-ul pt transformer este imaginea (redimensionata) si aplatizata
- ❑ Pro
  - Simplu (conceptual)
- ❑ Contra
  - O imagine de dimensiune  $n \times n$  necesita  $n^4$  elemente in fiecare matrice de atentie -> multa memorie



# Cum se poate integra mecanismul de atentie intr-o retea neuronală pentru procesarea imaginilor?

## □ V4: transformer aplicat pe patch-uri

### ■ Cum?



### ■ Pro

- Mai puțin convoluții

### ■ Contra

- Număr mare de parametrii
- Nevoie de multe date de antrenament

# Cum se poate imbunatati performanta unui ViT?

---

## □ Regularizare

- Weight decay, stochastic depth, dropout

## □ Data augmentation

- MixUp, RandAugment

# Cum se poate imbunatati performanta unui ViT?

## □ Distillation

CNN

Step 1: Train a **teacher model** on images and ground-truth labels



$P(\text{cat}) = 0.9$   
 $P(\text{dog}) = 0.1$

Cross  
Entropy  
Loss

GT label:  
Cat

Step 2: Train a **student model** to match predictions from the **teacher** (sometimes also to match GT labels)



$P(\text{cat}) = 0.1$   
 $P(\text{dog}) = 0.9$

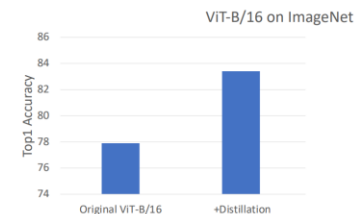
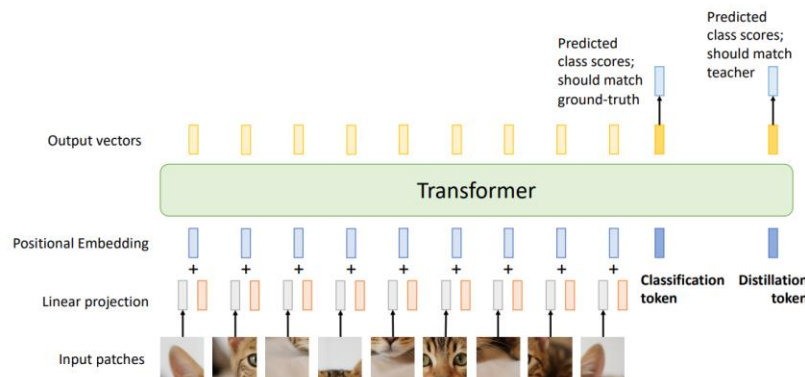
KL Divergence Loss



$P(\text{cat}) = 0.2$   
 $P(\text{dog}) = 0.8$

Cross  
Entropy  
Loss

GT label:  
Dog



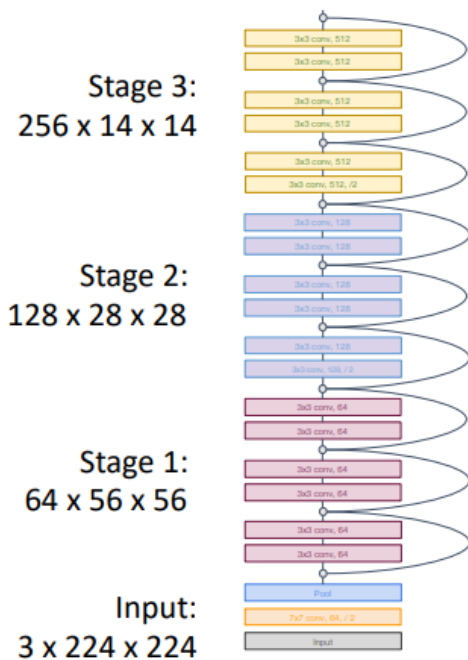
Hinton, G. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* [link](#)

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021, July). Training data-efficient image transformers & distillation through attention. In *International conference on machine learning* (pp. 10347-10357). PMLR [link](#)



# Cum se poate imbunatati performanta unui ViT?

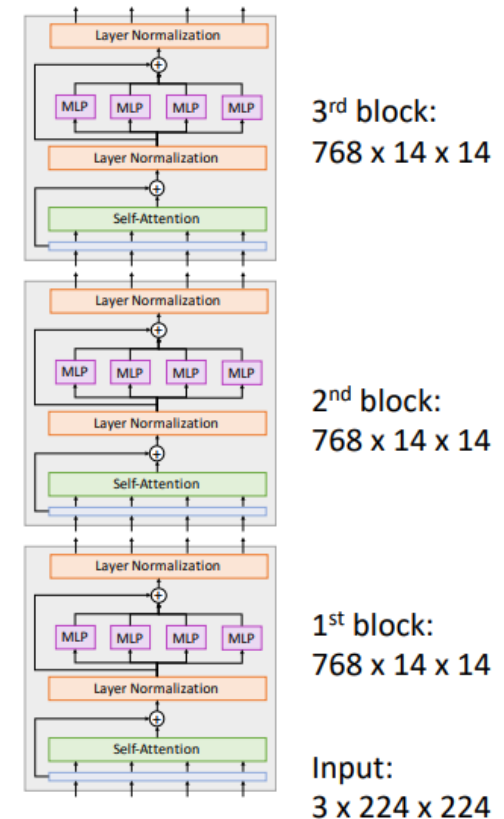
## ViT vs CNN



In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

In a ViT, all blocks have same resolution and number of channels (Isotropic architecture)



# More details

---

- ❑ [https://github.com/google-research/vision\\_transformer](https://github.com/google-research/vision_transformer)
- ❑ [https://huggingface.co/docs/transformers/model\\_doc/vit](https://huggingface.co/docs/transformers/model_doc/vit)