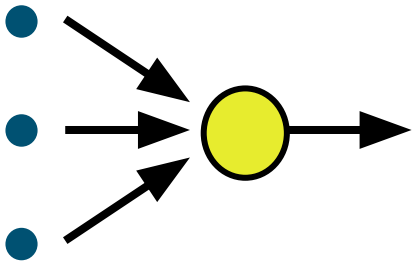
Intro to Deep Learning

1) The neuron model
2) Gradient descent
3) **Architectures (mostly for vision)**
4) Model evaluation

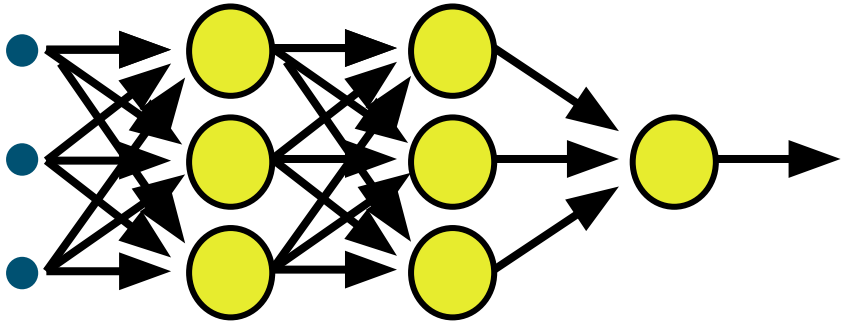
Back to the ‘ 80’s

- A single neuron (aka Perceptron) is a linear classifier.
- It needed a nonlinear version.
- The Multilayer perceptron (MLP) was formalized in 1986 by Rumelhart and colleagues

FROM the perceptron

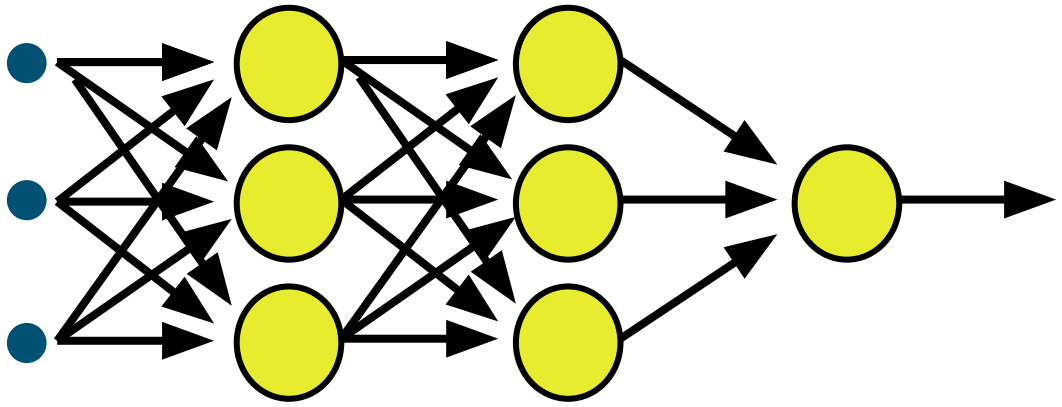


TO the MLP

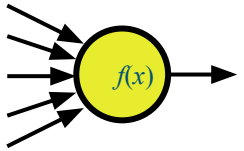


The multilayer perceptron (MLP)

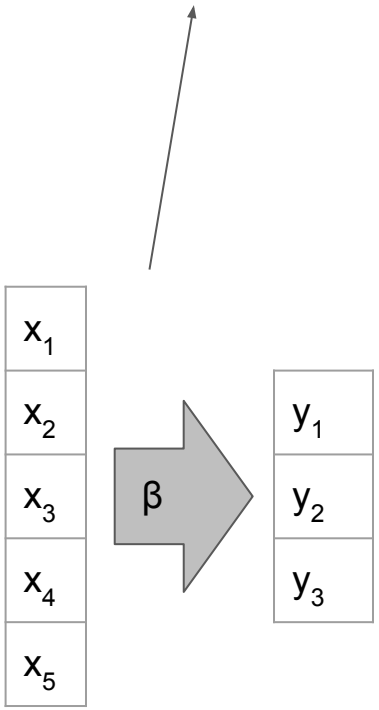
- It's a feed forward network: it goes from inputs to outputs, without loops
- Every neuron includes an activation function (e.g. sigmoid, see earlier slides)



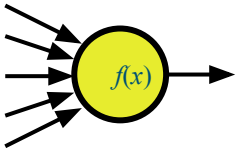
The starting point: the neuron model.

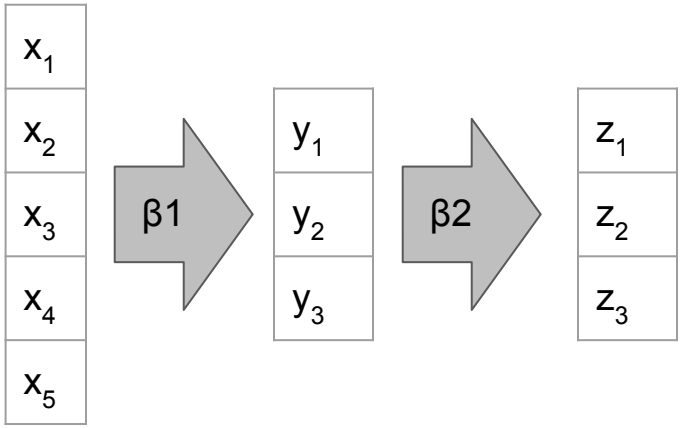


$$y = f(\boldsymbol{\beta}^\top \mathbf{x} + \beta_0)$$



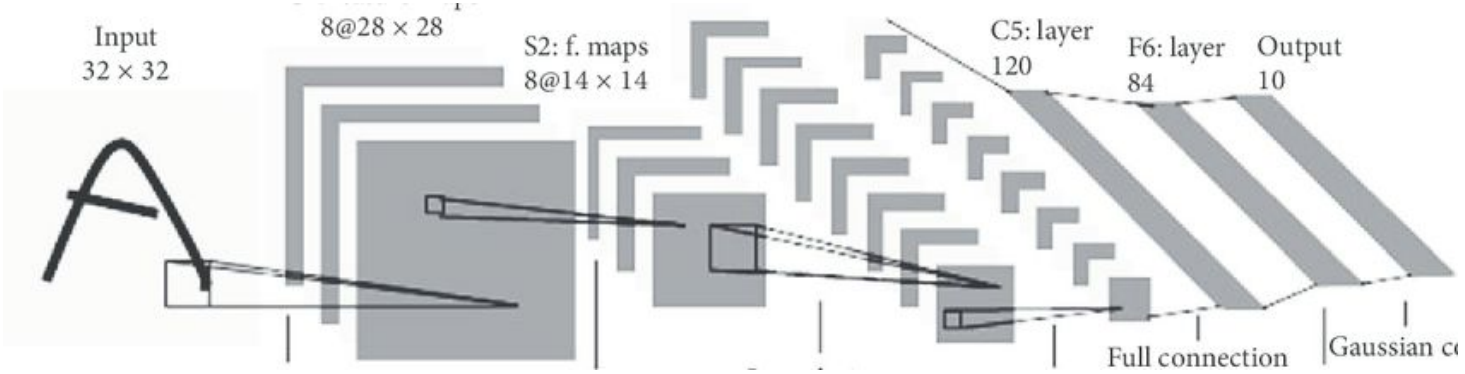
Multi-layer perceptron





Convolutional neural nets (CNN)

- In 1998, Yann LeCun proposed a network learning **spatial** filters on top of spatial filters
- They were (and are) called **convolutional neural networks**



What happened then?

- The CNN was considered interesting, but very hard to train. It needed
  - Loads of training data > nobody had them
  - A lot of computational power > same
- So it remained… a curiosity.

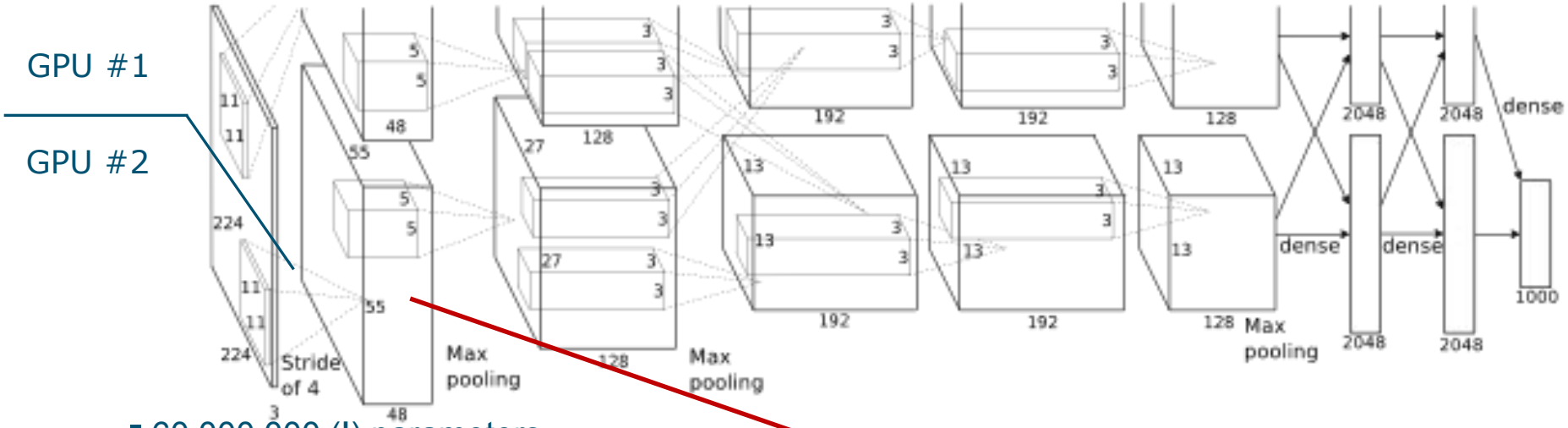
What changed everything?

- Very big datasets started to appear

- Graphic processing units made computation of massive parallel computing possible.

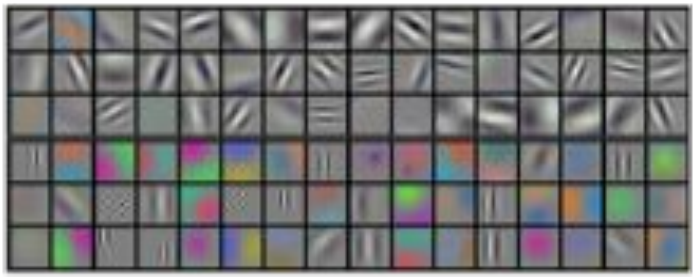
The game changer: AlexNet (2012)

- Krizhevsky *et al*., NIPS 2012:



- 60,000,000 (!) parameters

- Training: 5-6 days



And Convolutional neural networks (CNNs) started to beat all benchmarks

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

[Top-1: error when looking for the right answer in the most likely predicted class; Top-5: same, but looking among the 5 most probable classes]

So why all this fuss about CNNs?

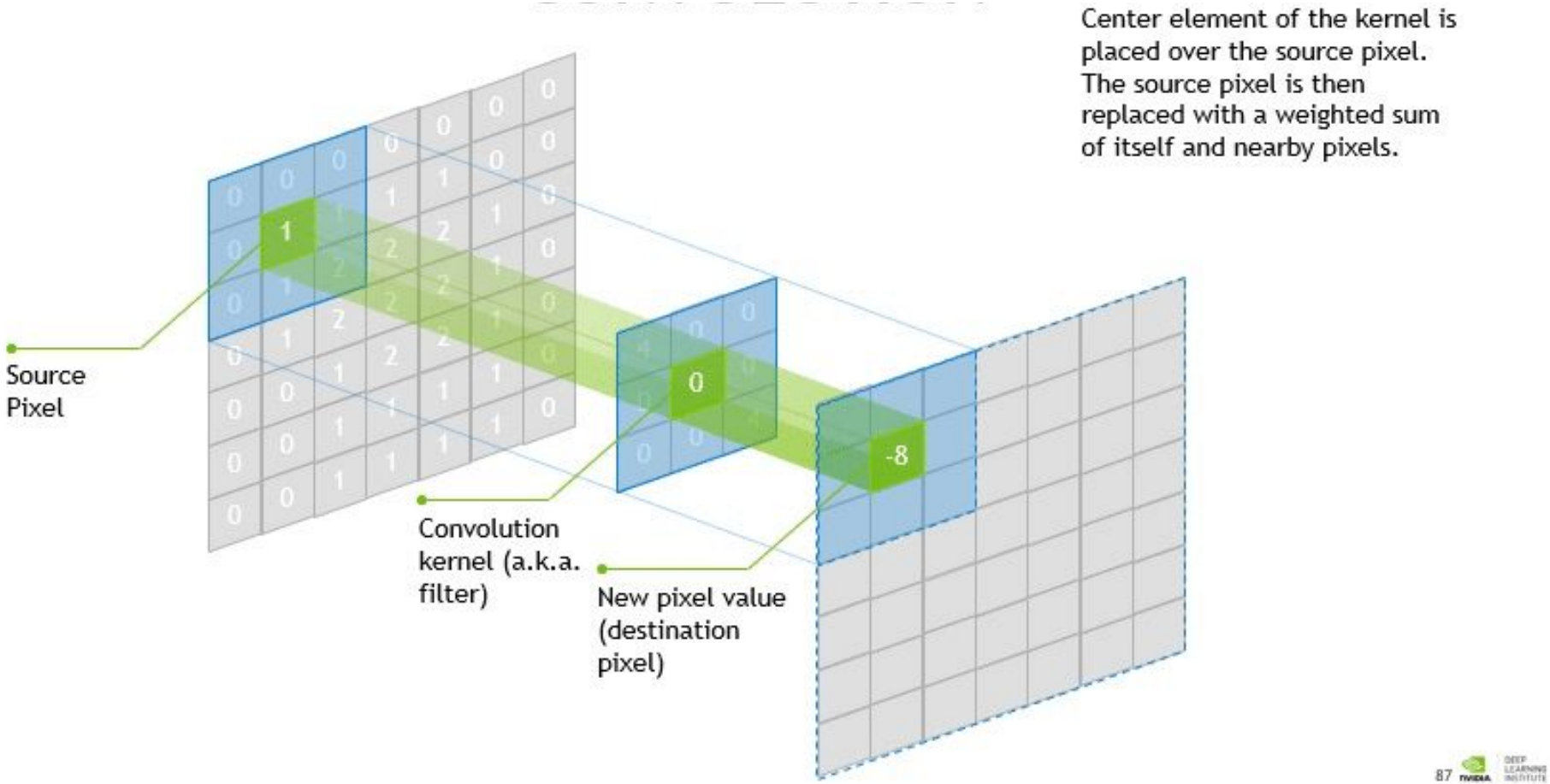
https://devblogs.nvidia.com/parallelforall/low-power-image-recognition-challenge-jets/

But how do CNNs work?

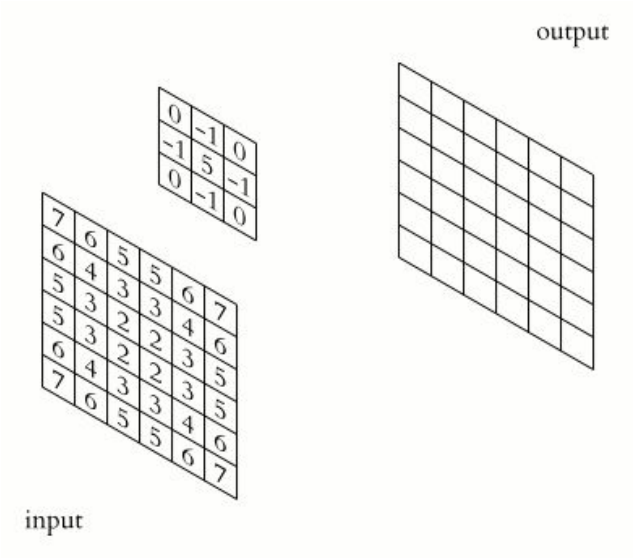
Convolutional neural networks (CNN)

- They are conceptually very similar to MLPs
- But their weights (β) are 2D convolutional filters
- For this, they are very well suited for images

Convolutions (kernels in image processing)



The convolution operator

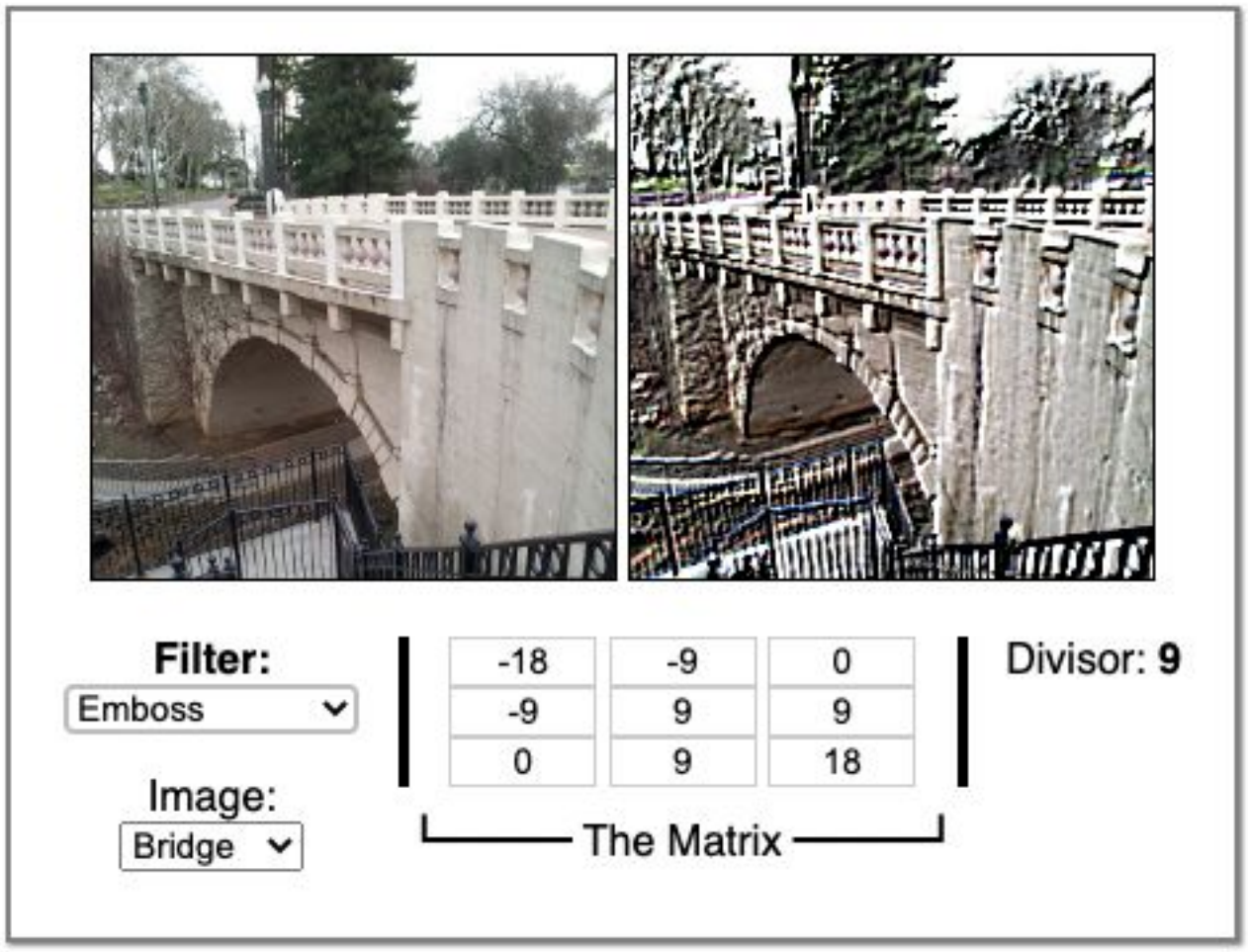


[]Wikimedia]

Convolutions (kernels in image processing)

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

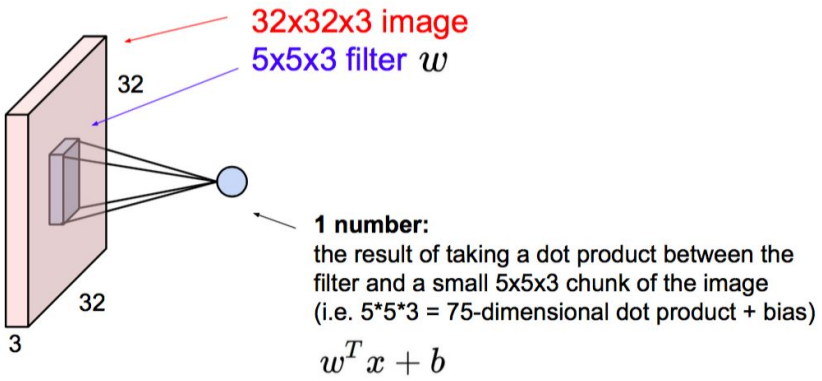
Interactive exercise



https://beej.us/blog/data/convolution-image-processing/

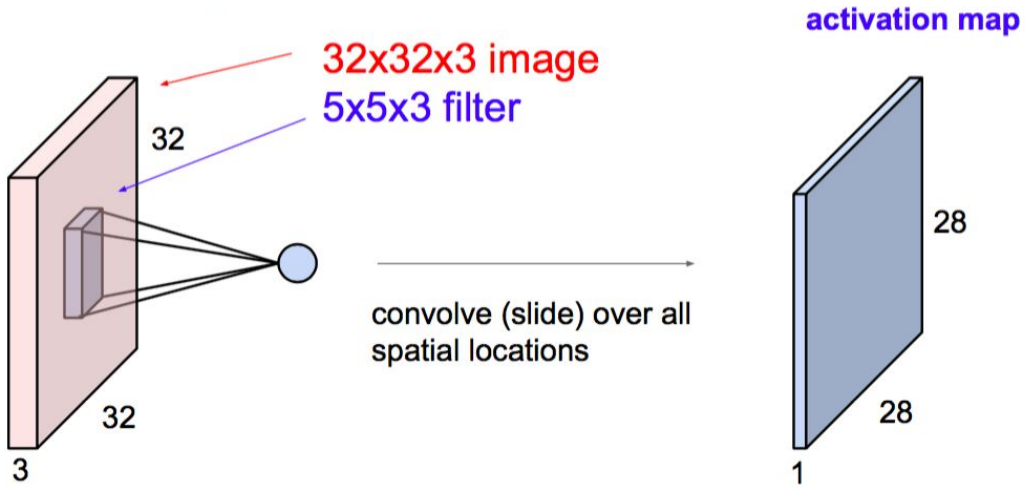
Convolutional neural networks (CNN)

- In convolutional neural networks, the filters are spatial (on 2D grids).
  - **local** : they convolve the values of the image in a local window



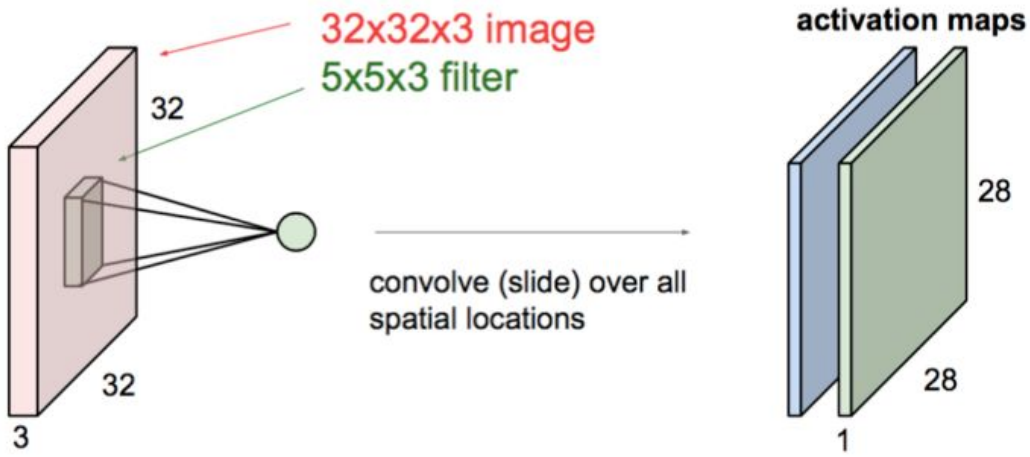
  - **shared**: the same filter is applied everywhere in the image

Convolutional neural networks (CNN)



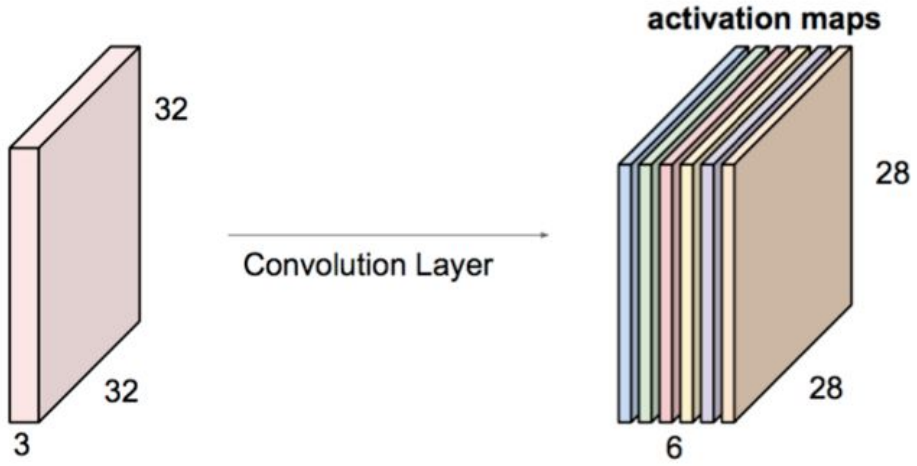
[From Wegner, ETH]

Convolutional neural networks (CNN)



[From Wegner, ETH]

Convolutional neural networks (CNN)



Shared?

Each filter (each neuron) is kept the same throughout the image.
We apply the same filter everywhere.

This is one of the keys of the success of modern CNN

You see why?

Shared?

Each filter (each neuron) is kept the same throughout the image.
We apply the same filter everywhere.

This is one of the keys of the success of modern CNN

You see why?

"Recycling" the same operation allows to have much less parameters to learn!

And the filters are **learned**

Traditional model (e.g. RF, SVM)

Extract feature representations

Train a classifier

Prediction

True labels Y

Convolutional neural net

Learn both feature representations AND a classifier

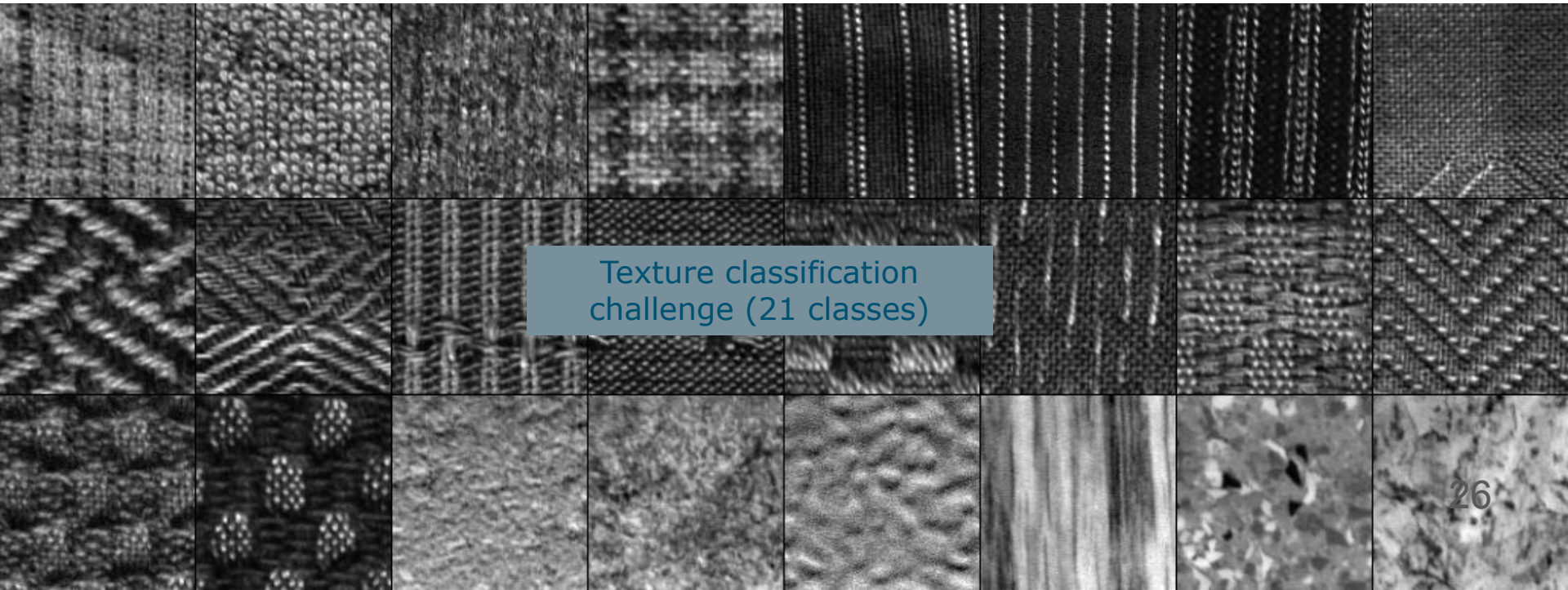
Prediction

True labels Y

- Keeps the representations simple (convolutional filters, i.e. weighted averages)
- Learns a lot of them, cascading and multiscale
- By “learns”, I mean adapt filters weights to minimize prediction error (i.e. backpropagation)

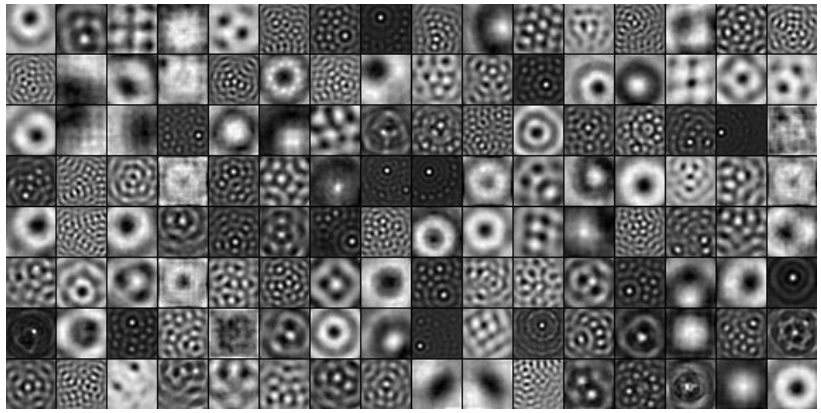
And the filters are **learned**

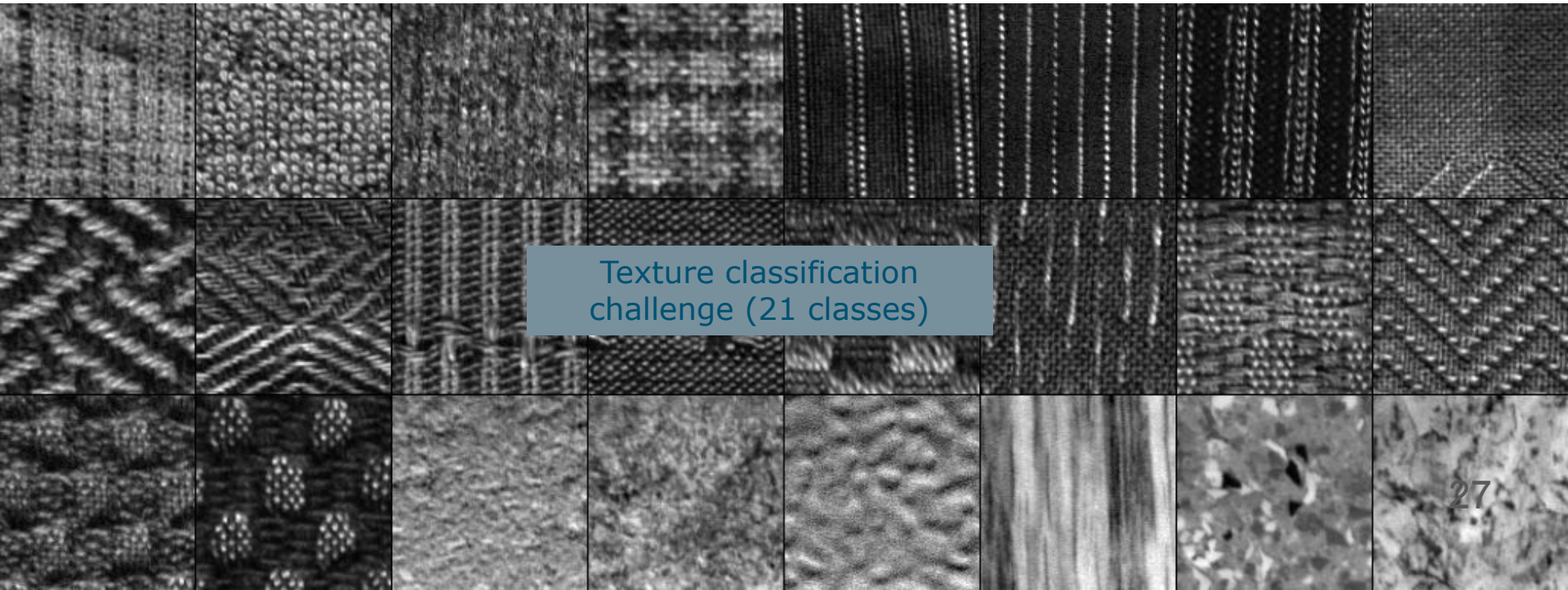
- Convolutional filters start with random numbers
- Iteratively they are improved: each coefficient is updated in the direction of largest gradient of the cost function

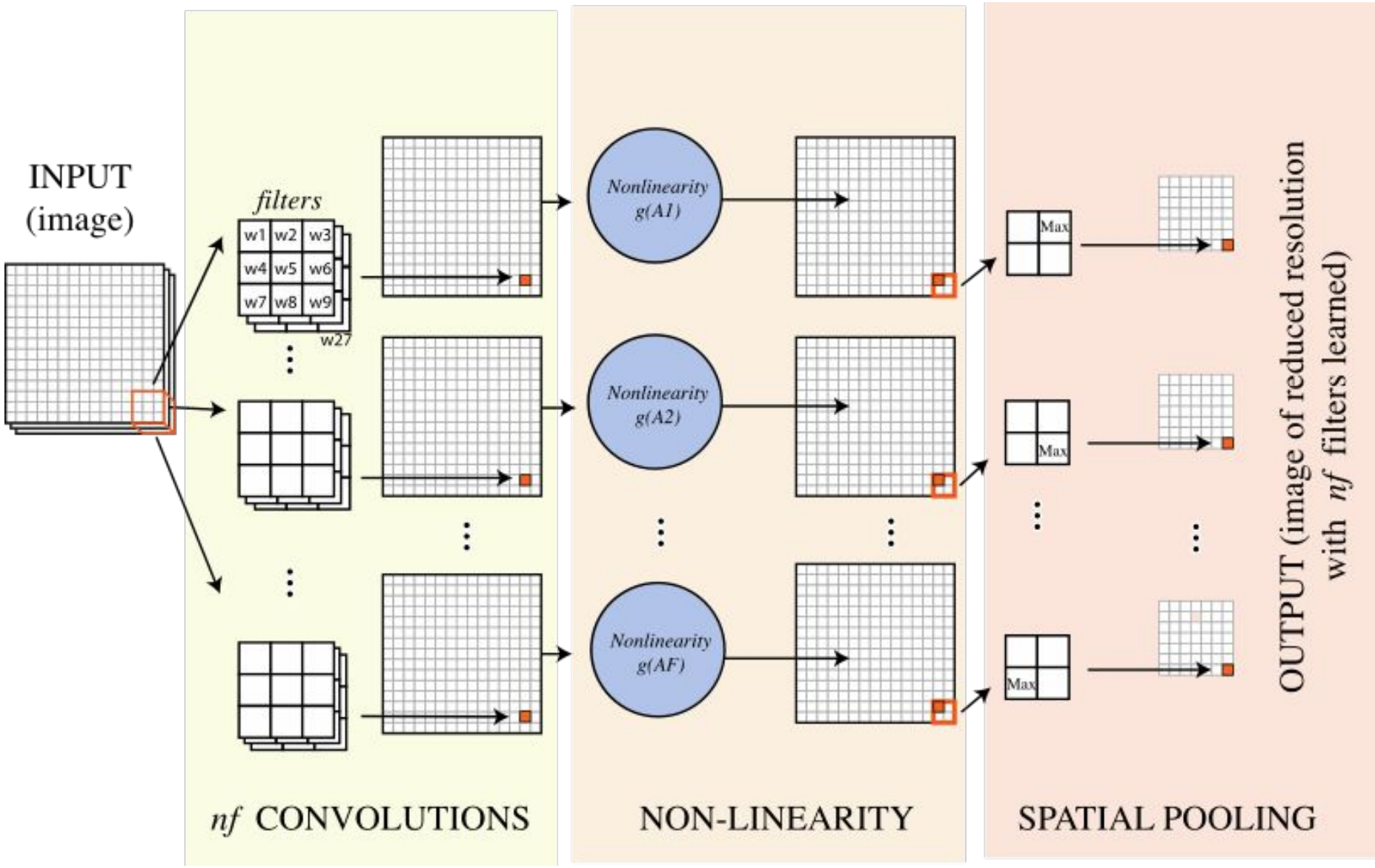


And the filters are **learned**

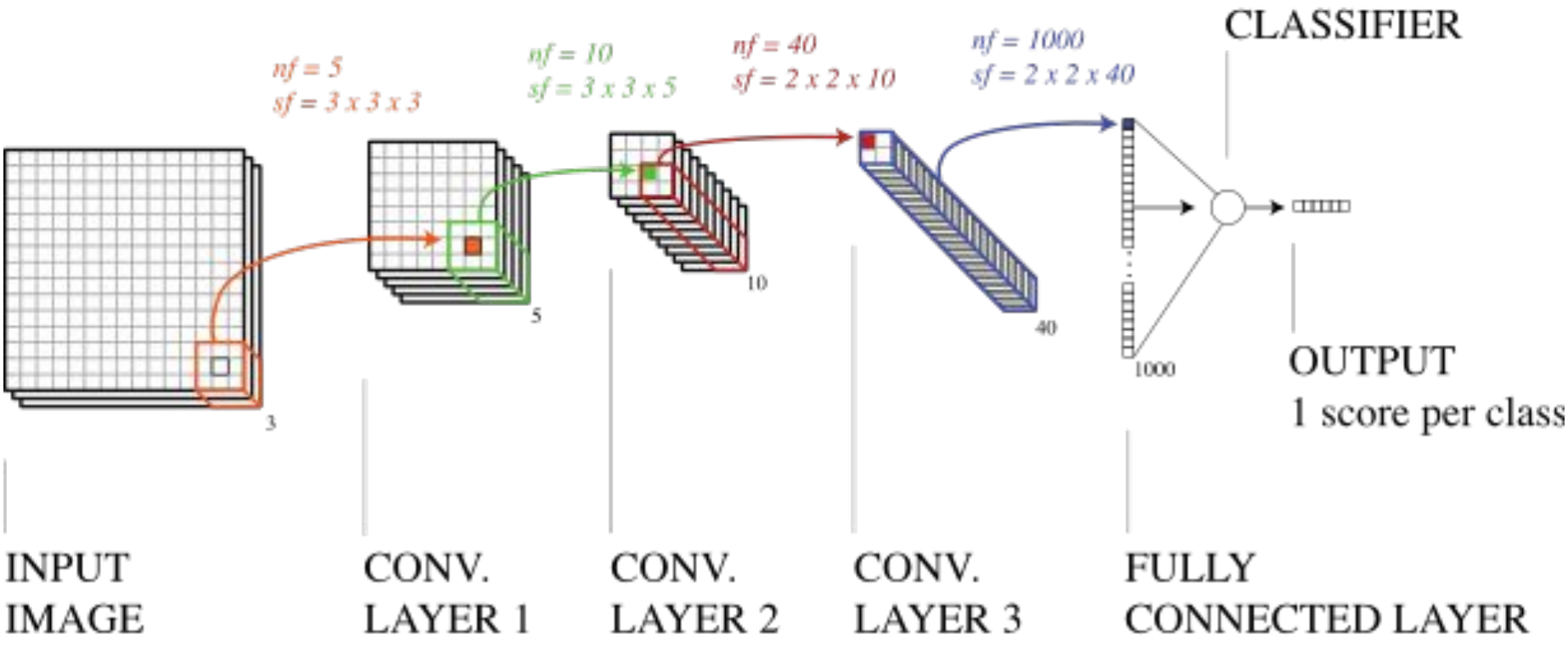
FILTERS LEARNED





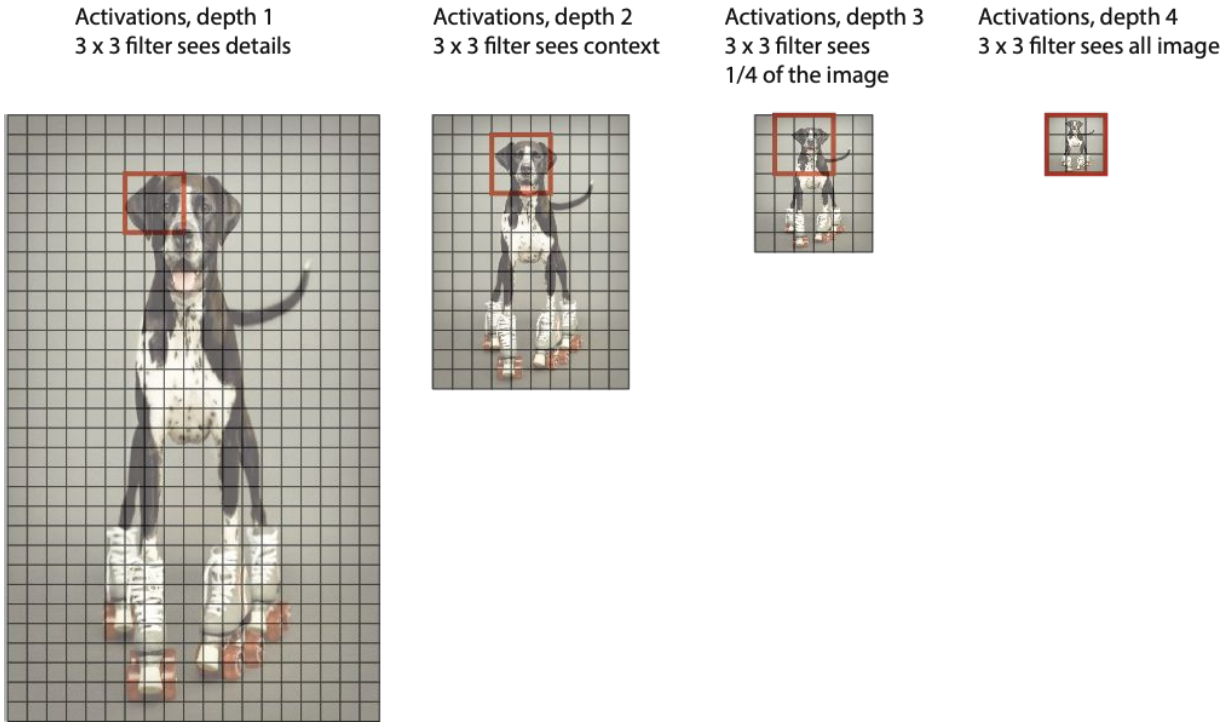

INPUT
(image)
filters
w1 w2 w3
w4 w5 w6
w7 w8 w9
w27
Nonlinearity
g(A1)
Nonlinearity
g(A2)
Nonlinearity
g(AF)
Max
Max
Max
OUTPUT (image of reduced resolution
with nf filters learned)
nf CONVOLUTIONS
NON-LINEARITY
SPATIAL POOLING

Convolutional neural networks (CNN)



Pooling increases the receptive field:

- Example: if we constantly apply 3 x 3 filters, this is what they "see"



HOW A DEEP NEURAL NETWORK SEES

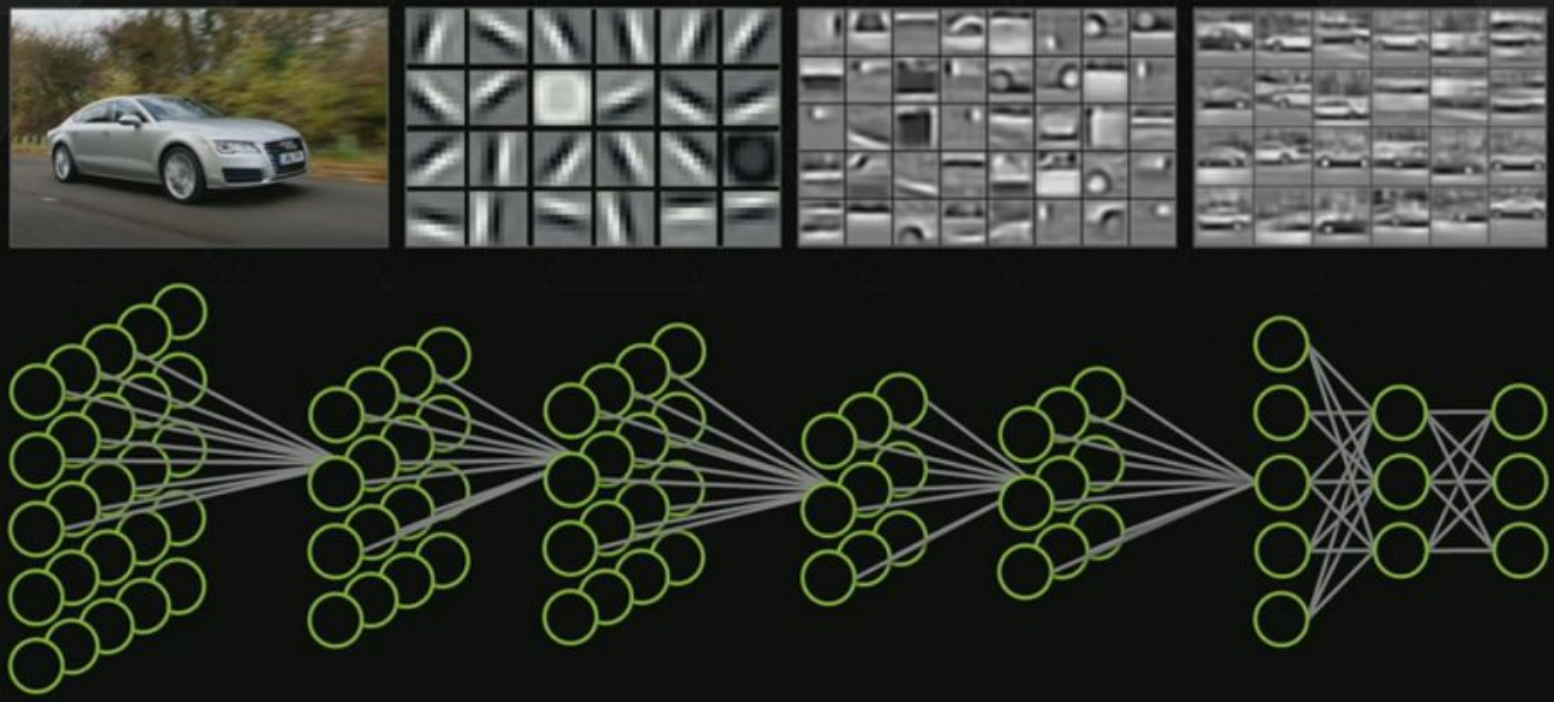
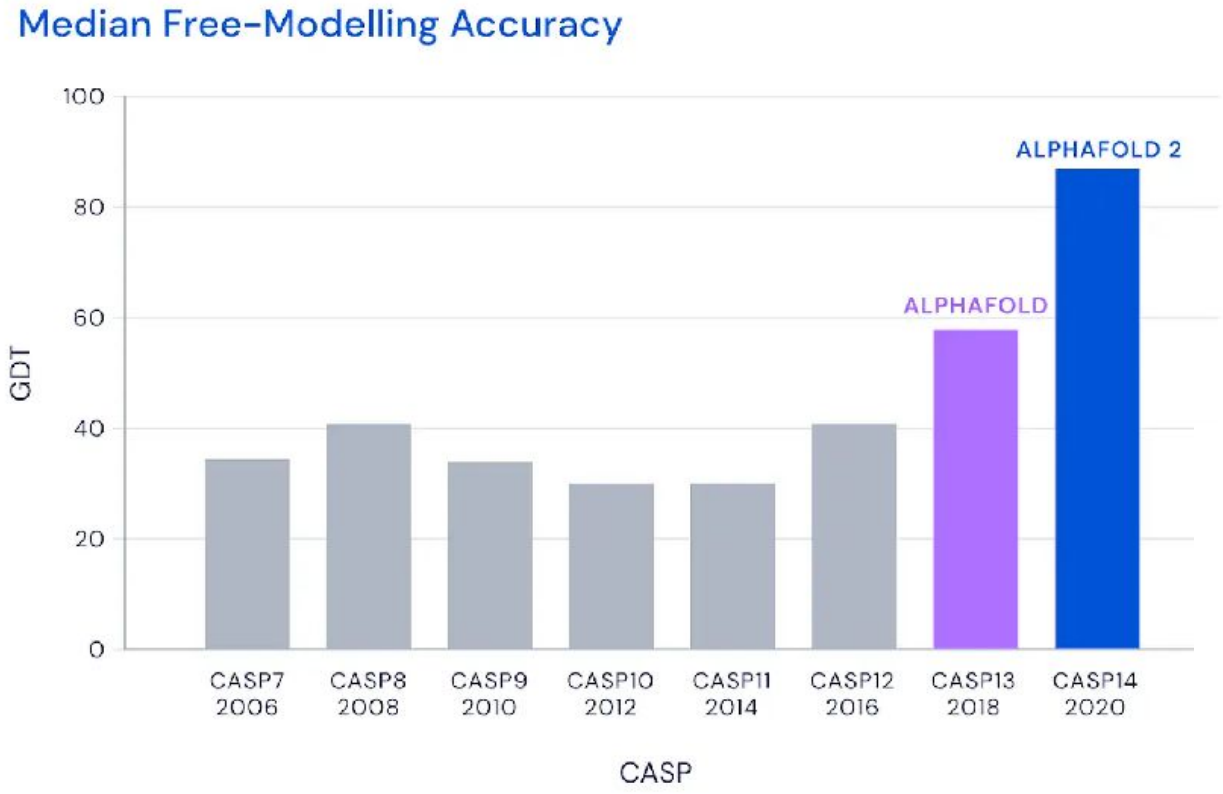
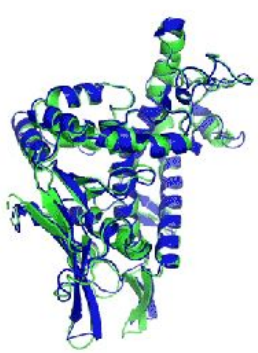


Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011.
Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.

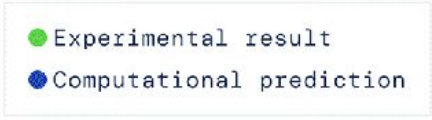
Deep Learning beyond CNNs





T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

T1049 / 6y4f
93.3 GDT
(adhesin tip)



https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

Deep Learning beyond CNNs

https://deepmind.com/blog/article/nowcasting

Deep Learning beyond CNNs

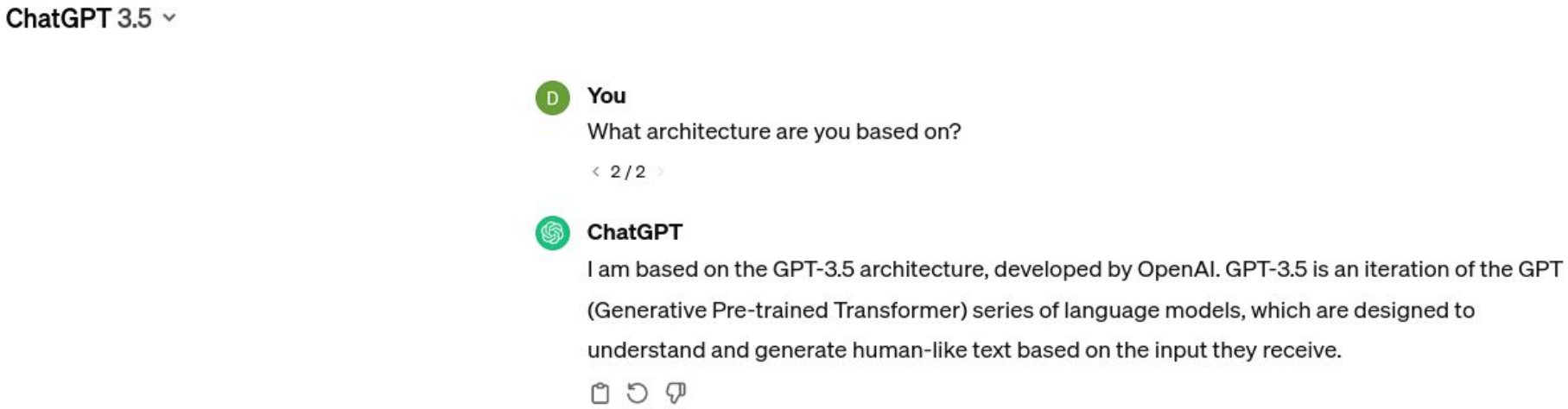
TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

https://openai.com/blog/dall-e/

Deep Learning beyond CNNs



Transformers

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

Transformers

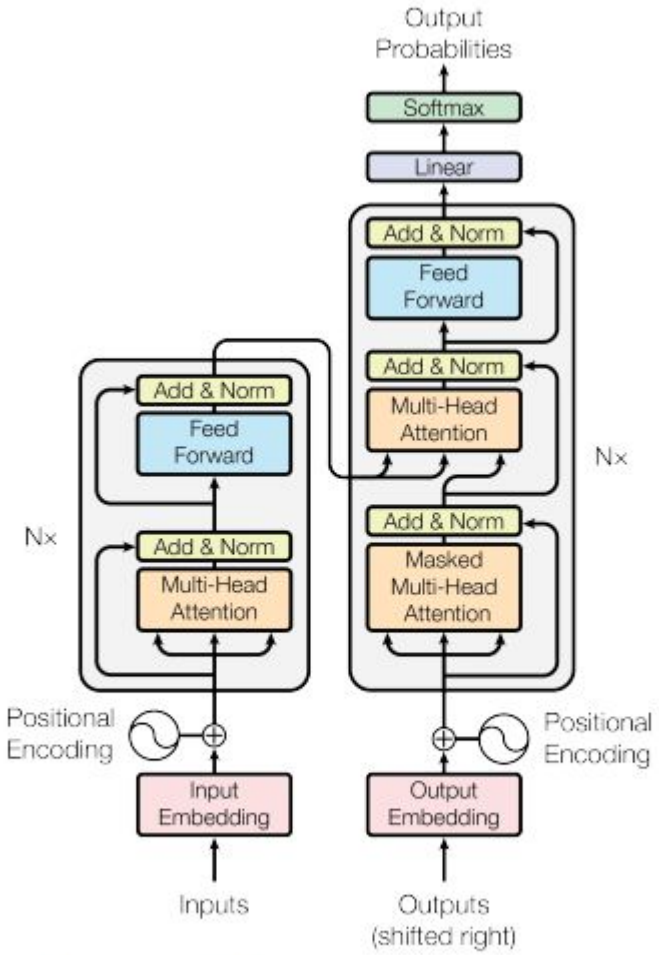


Figure 1: The Transformer - model architecture.

Transformers

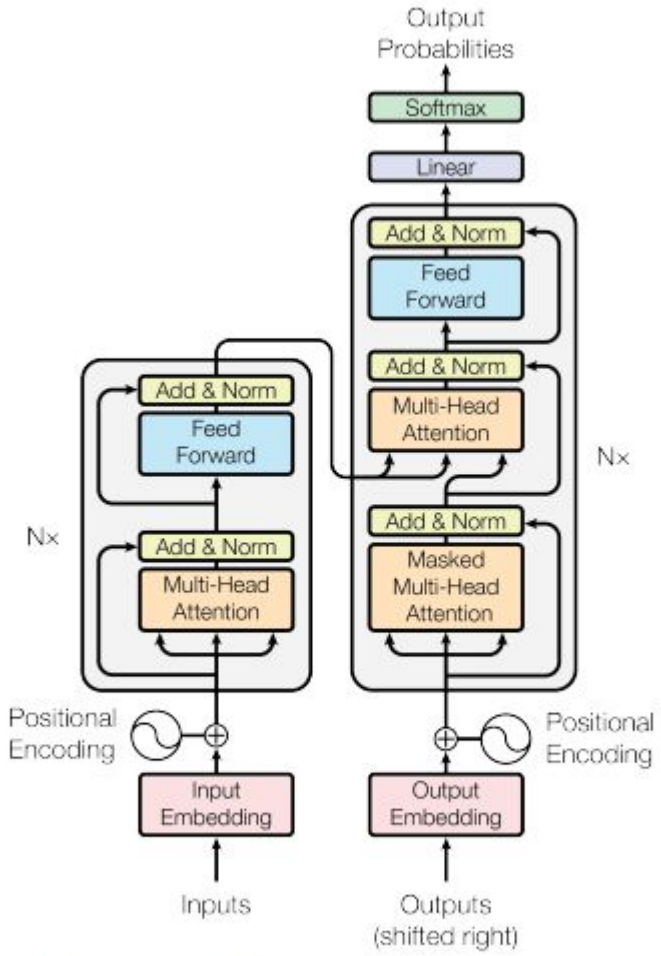
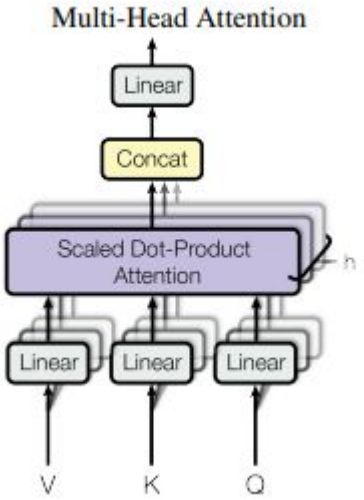


Figure 1: The Transformer - model architecture.



Transformers

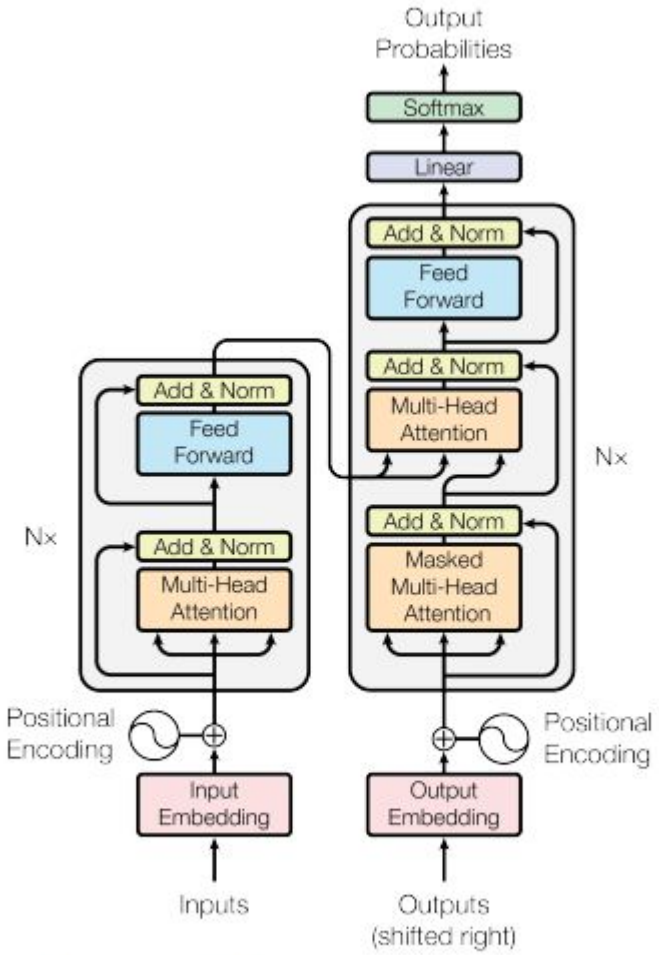
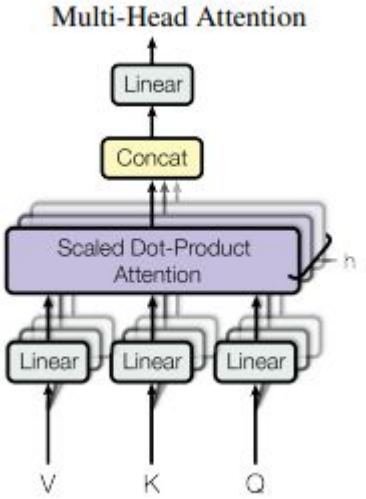
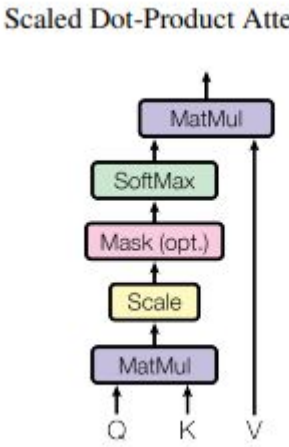


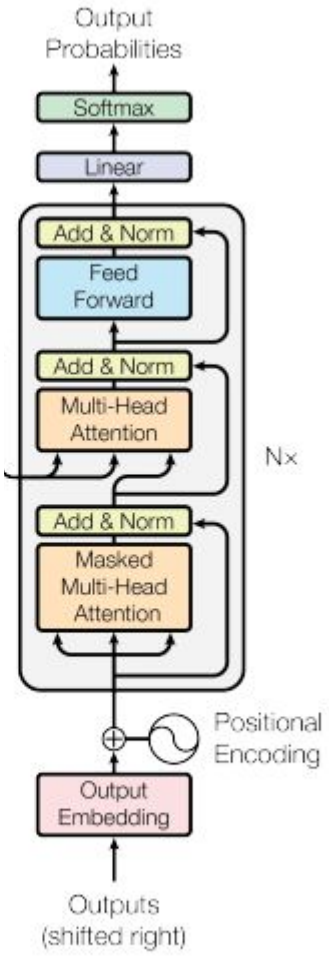
Figure 1: The Transformer - model architecture.



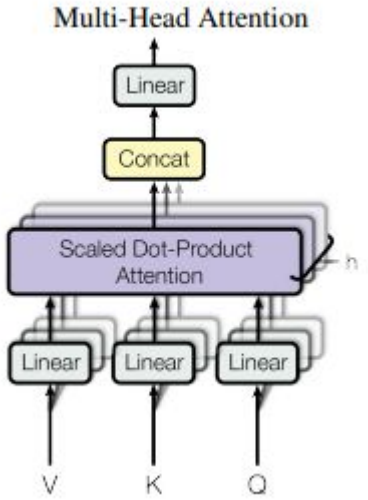


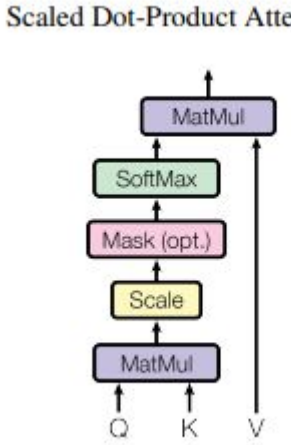
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Transformers



ıer - model architecture.





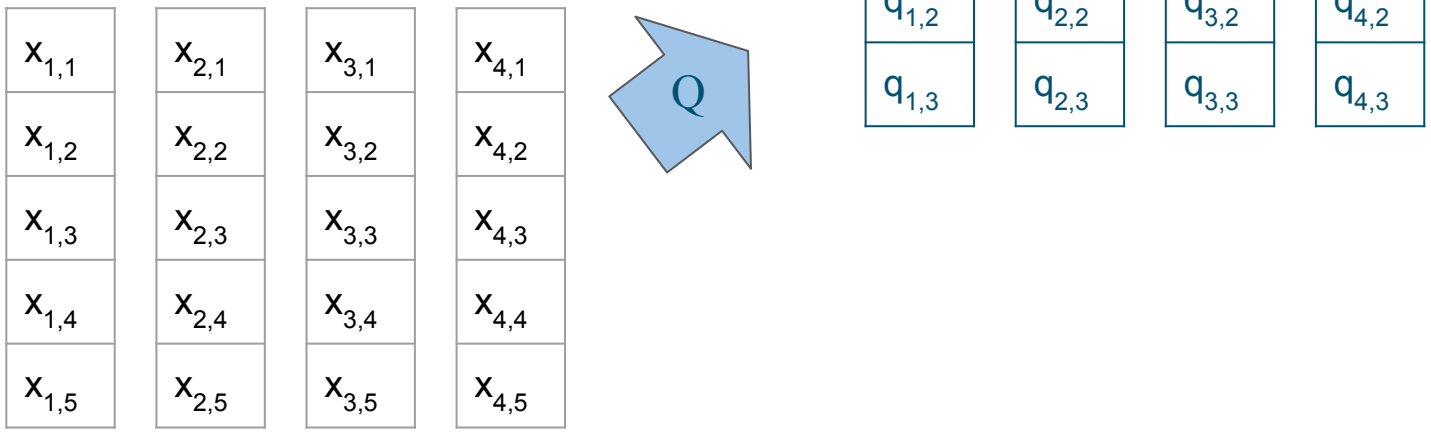
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Self-Attention

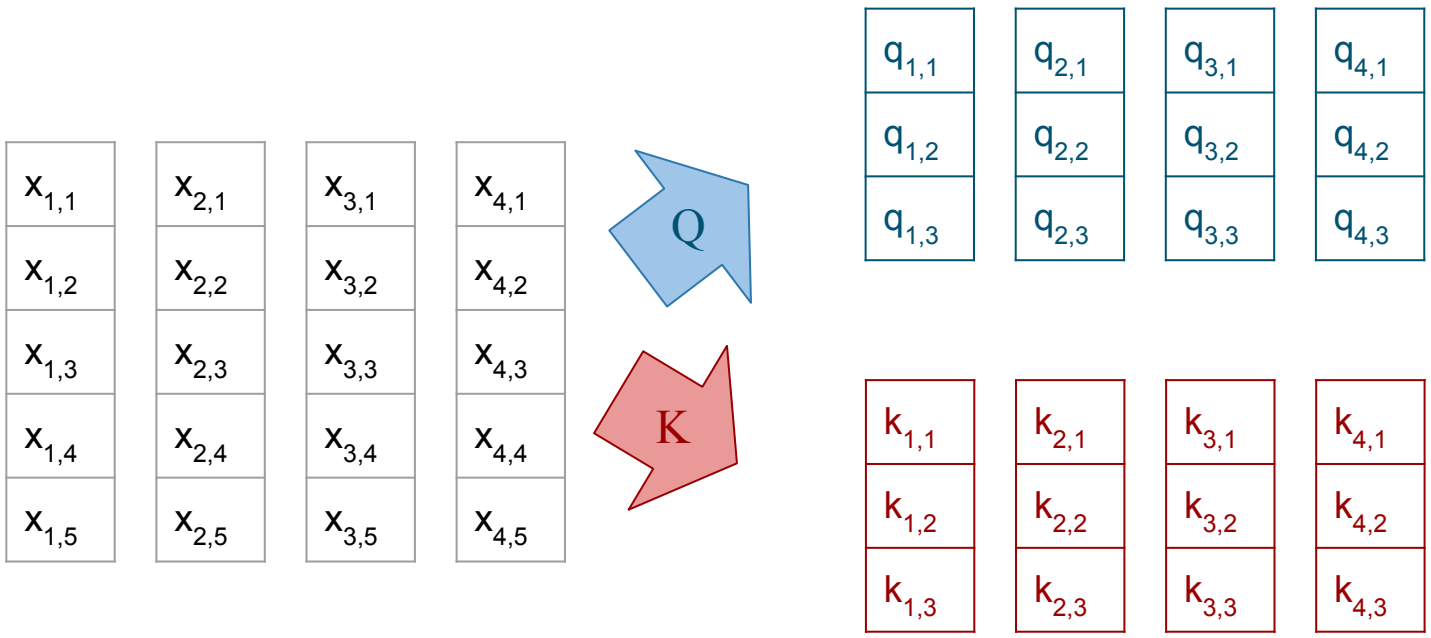
Each colum vector is a token: a word, a patch, etc.

$x_{1,1}$	$x_{2,1}$	$x_{3,1}$	$x_{4,1}$
$x_{1,2}$	$x_{2,2}$	$x_{3,2}$	$x_{4,2}$
$x_{1,3}$	$x_{2,3}$	$x_{3,3}$	$x_{4,3}$
$x_{1,4}$	$x_{2,4}$	$x_{3,4}$	$x_{4,4}$
$x_{1,5}$	$x_{2,5}$	$x_{3,5}$	$x_{4,5}$

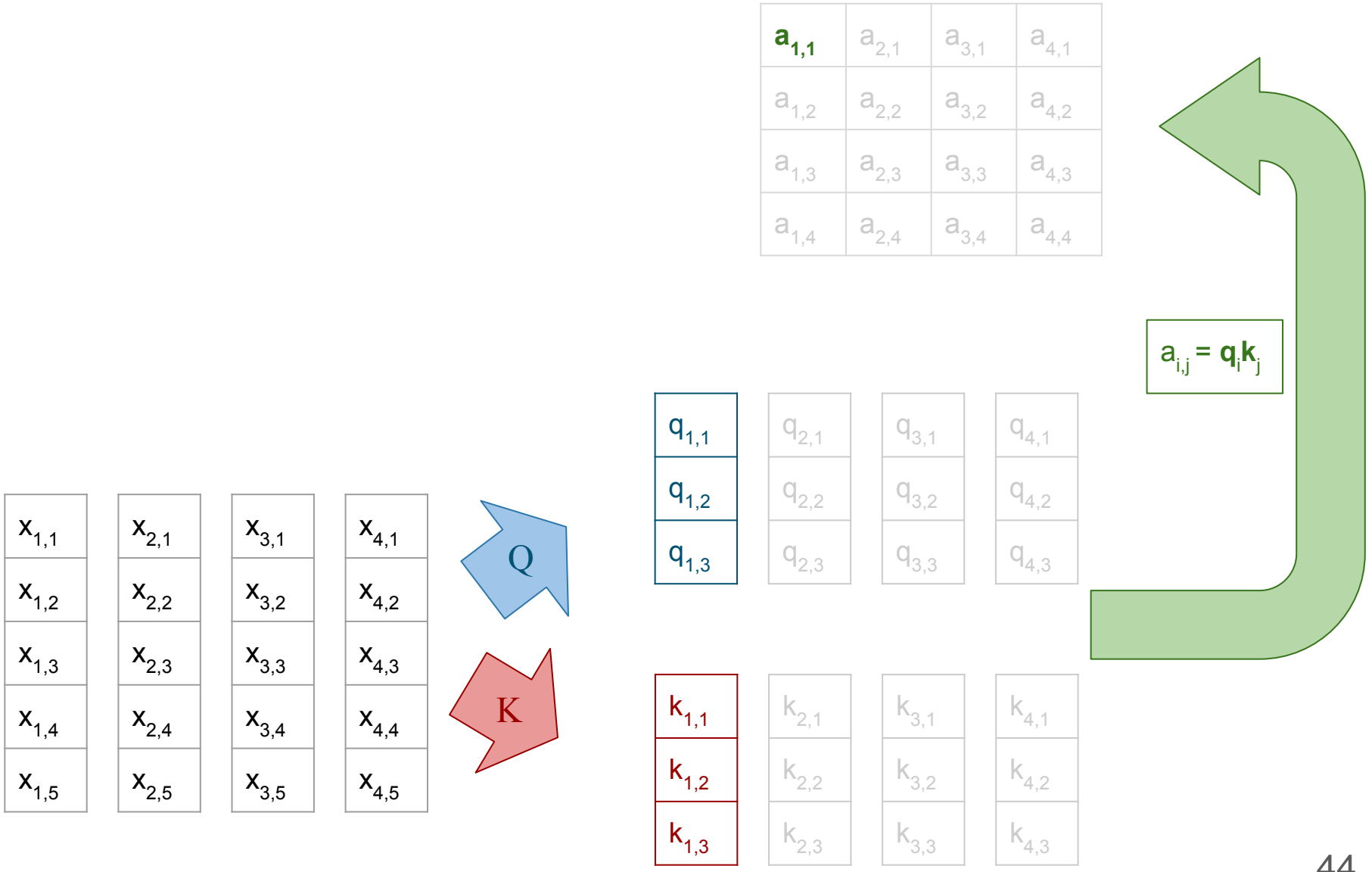
Self-Attention



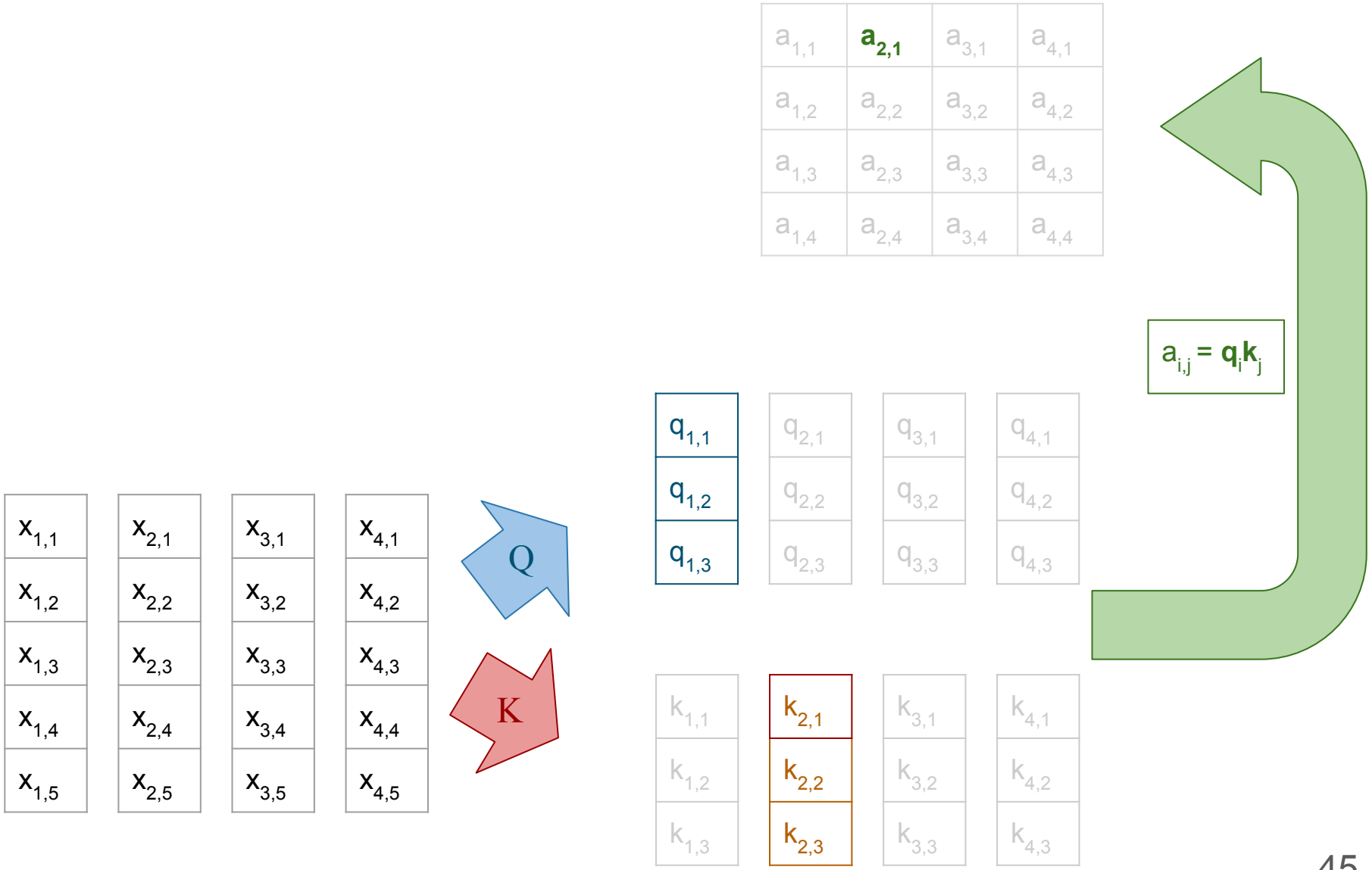
Self-Attention



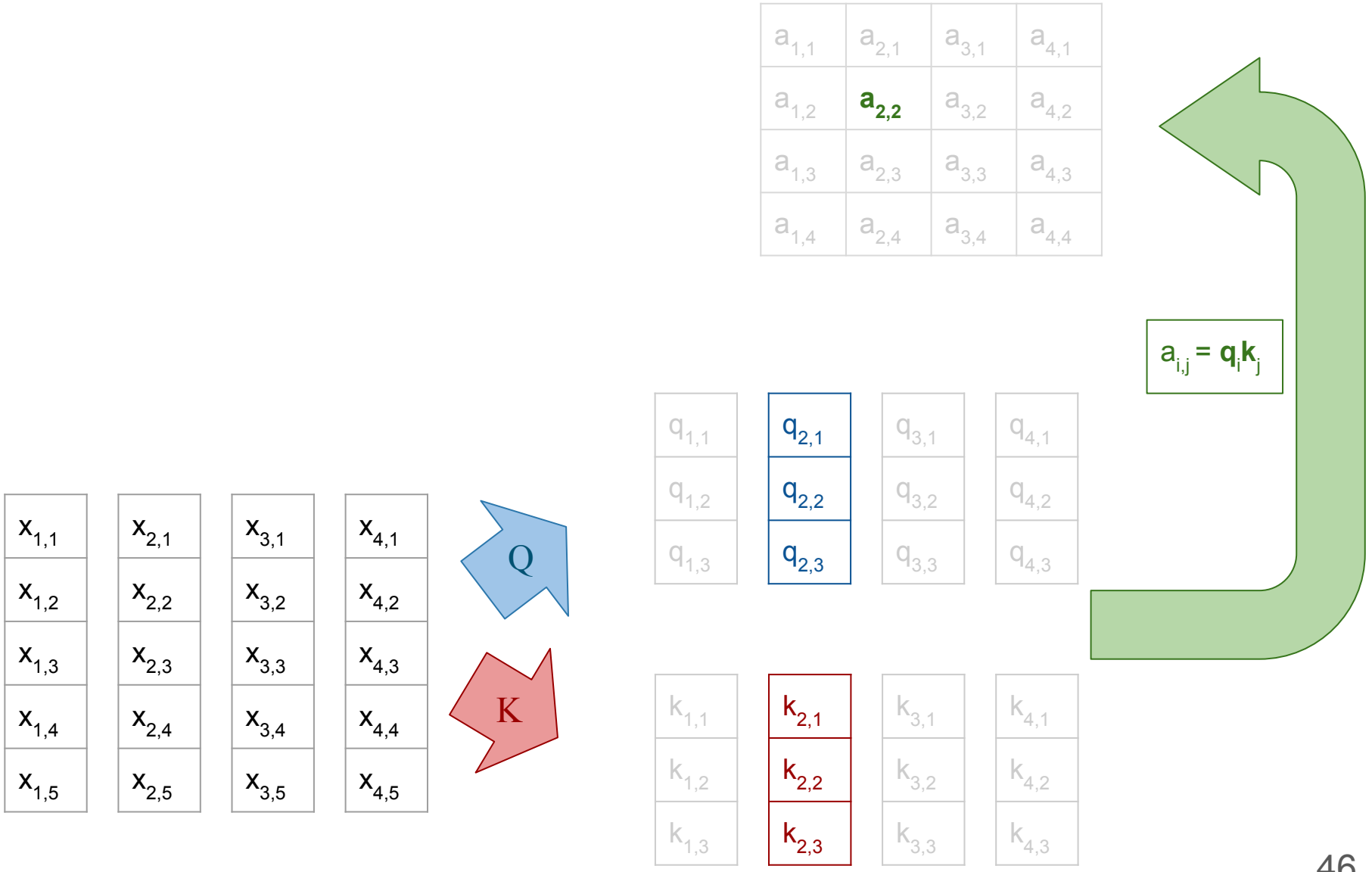
Self-Attention



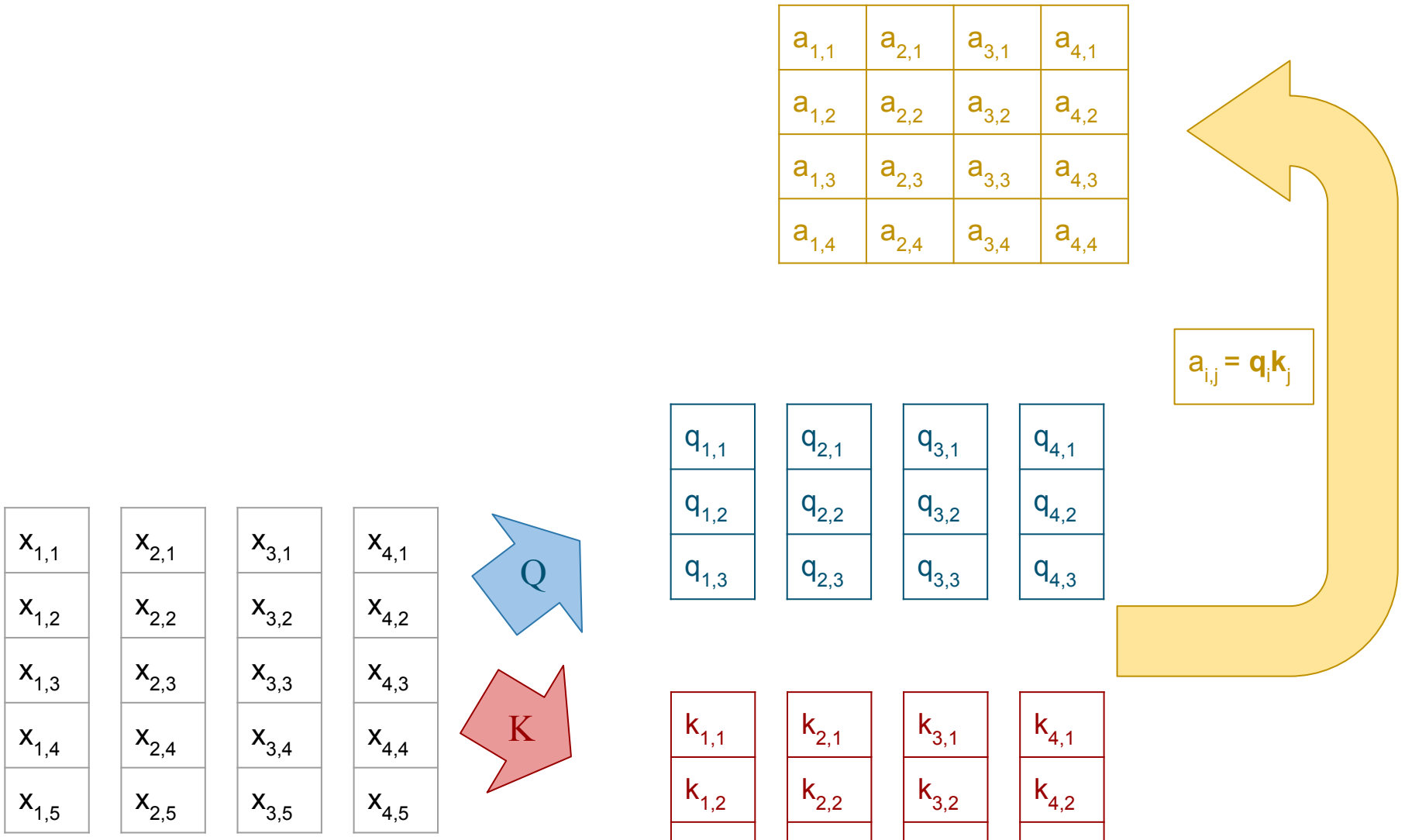
Self-Attention



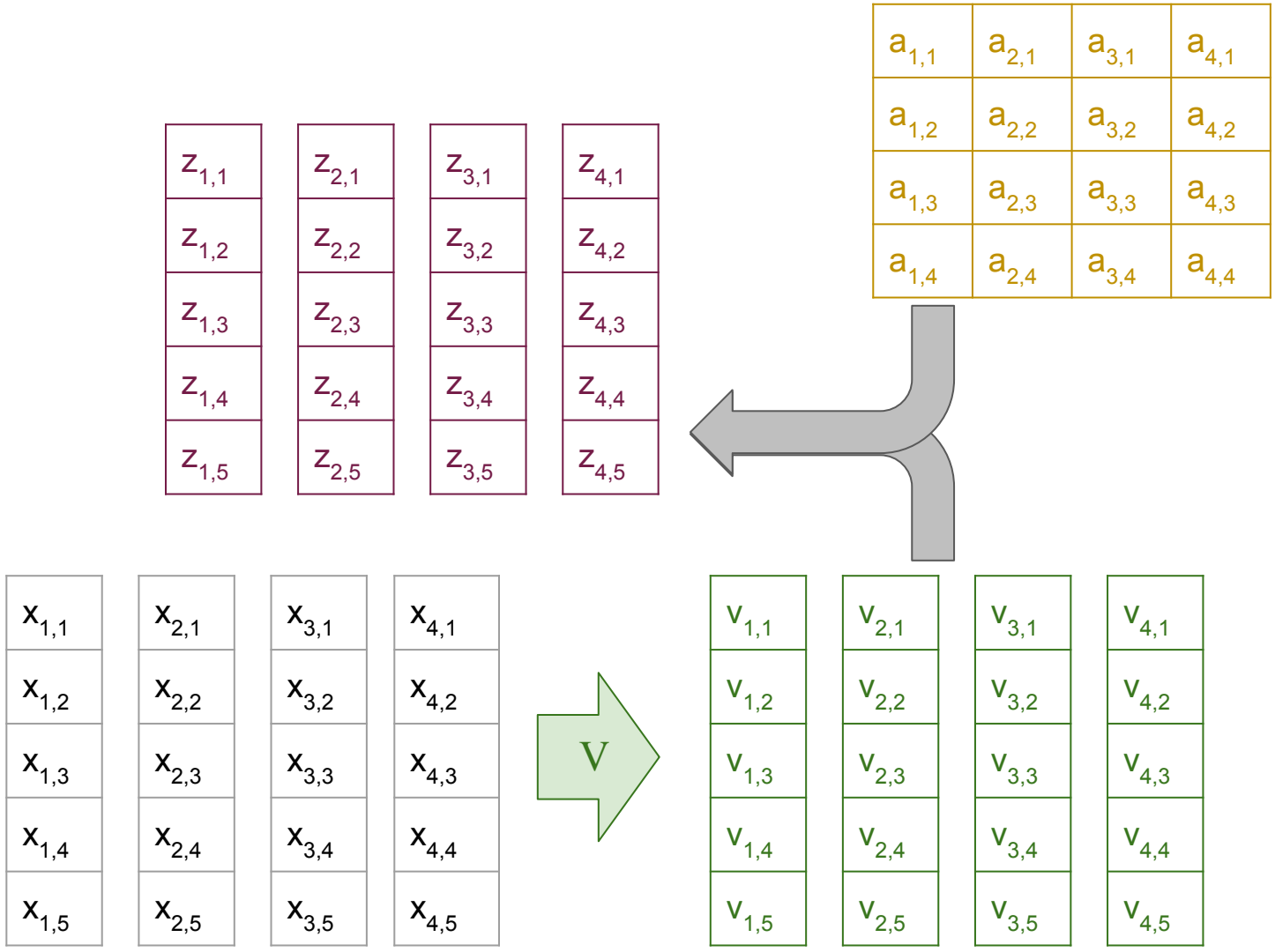
Self-Attention



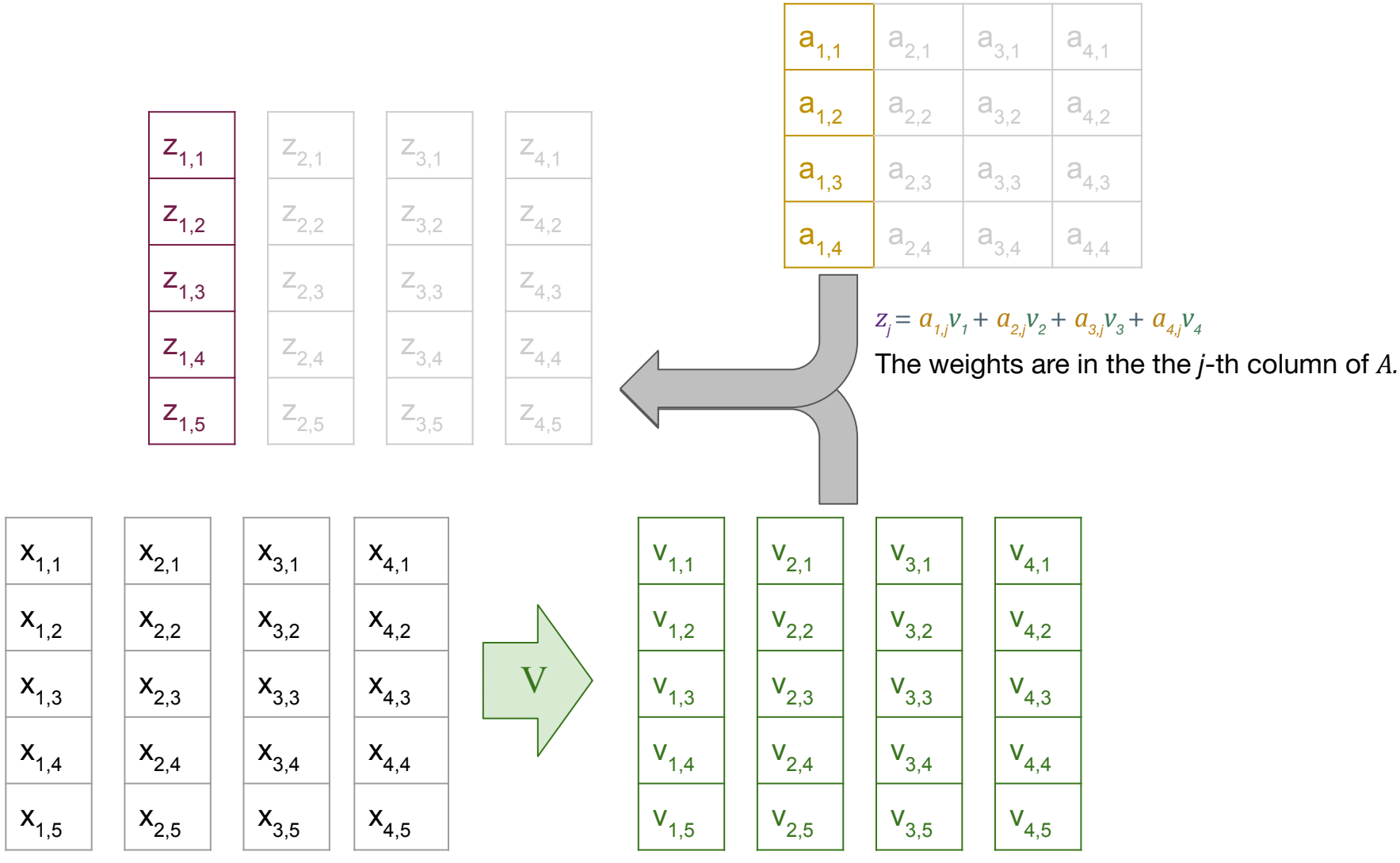
Self-Attention



Self-Attention



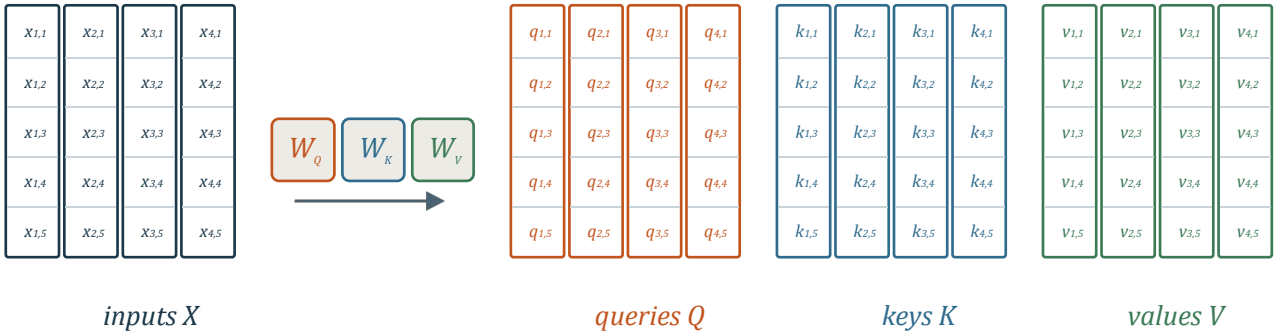
Self-Attention



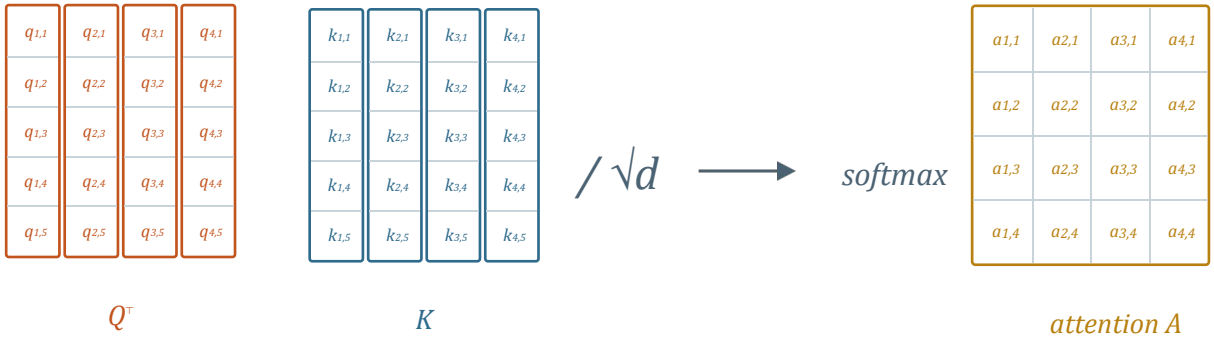
Self-Attention

$$Z = softmax(Q^T K / \sqrt{d})\, V$$

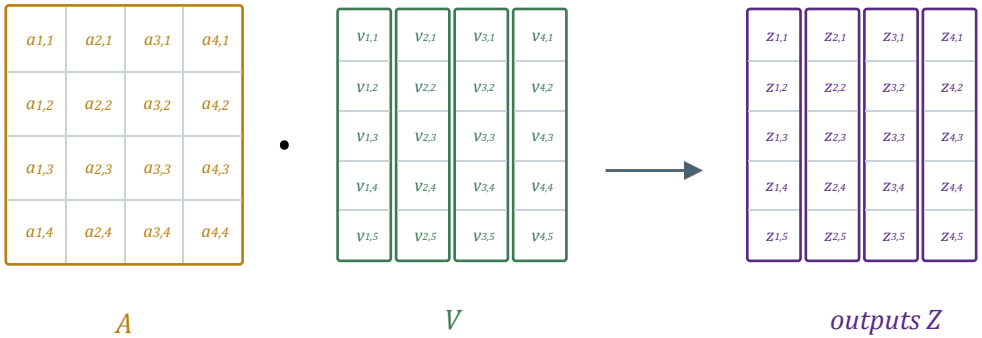
1) PROJECT



2) SCORE



3) COMBINE



Summary

- Perceptrons are “neuron-inspired” linear discriminants
- Multilayer perceptrons are trainable, nonlinear discriminants
- Feed-forward neural networks in general can be used for classification, regression and feature extraction
- There is a large body of alternative neural nets
- Key problems in the application of ANNs are choosing the right size and good training parameters

Summary (2)

- Convolutional neural networks have a constrained architecture encoding prior knowledge of the problem
- Deep learning is concerned with constructing extremely large neural networks that depend on:
  - special hardware (GPUs), to be able to train them
  - specific tricks (e.g. rectified linear units) to prevent overfitting