Raoni LOURENÇO | 02/10/2024

Master in Data Science
Advanced topics in applied Machine Learning

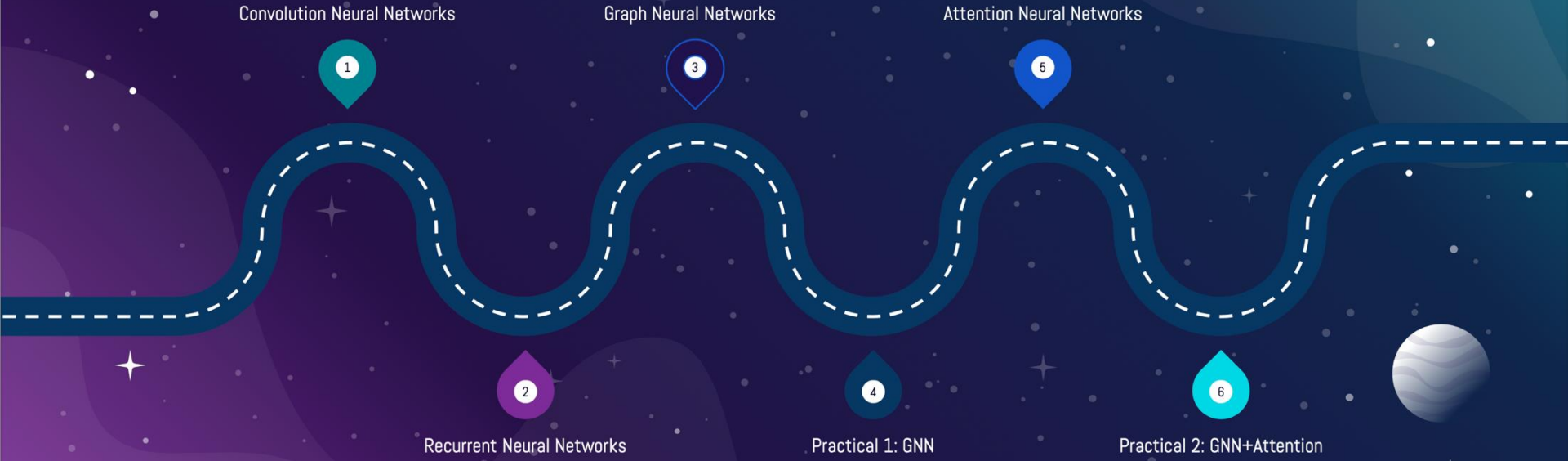
Hello!

I AM Raoni LOURENÇO

Postodoctoral researcher @ University of Luxembourg

raoni.lourenco@uni.lu

Today's session



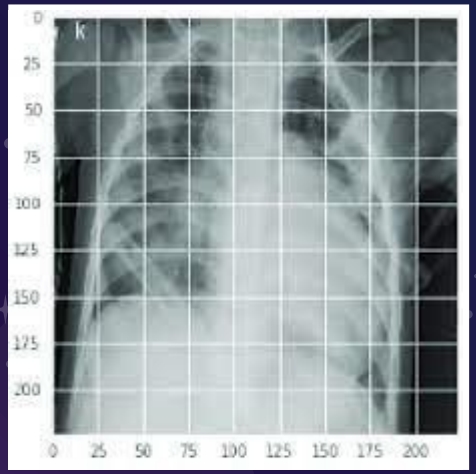

0.
Practical
https://t.ly/uLHFH

1.
Convolutional Neural Networks

Convolutional Neural Networks

Multi-layer perceptron for images?

- Consider an image of size 224 x 224



Convolutional Neural Networks

Multi-layer perceptron for images?

- Consider an image of size 224 x 224 (resized to 250^2)
- Vectorize the 2D image to a 1D vector as input feature
- For each hidden node, it requires 250x250 weights ~ 62,500

Convolutional Neural Networks

Multi-layer perceptron for images?

- Consider an image of size 224 x 224
- Vectorize the 2D image to a 1D vector as input feature
- For each hidden node, it requires 250x250 weights ~ 62,500
- How about multiple hidden layer? Bigger image?

Convolutional Neural Networks

Multi-layer perceptron for images?

- Consider an image of size 224 x 224
- Vectorize the 2D image to a 1D vector as input feature
- For each hidden node, it requires 250x250 weights ~ 62,500
- How about multiple hidden layer? Bigger image?

- Too many weights, computational and memory expensive

Convolutional Neural Networks

Multi-layer perceptron for images?

- Consider an image of size 224 x 224
- Vectorize the 2D image to a 1D vector as input feature
- For each hidden node, it requires 250x250 weights ~ 62,500
- How about multiple hidden layer? Bigger image?

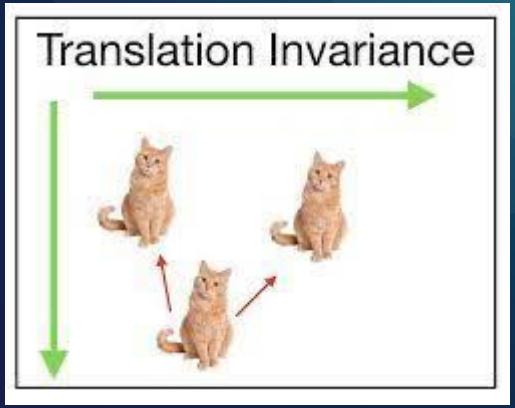
- Too many weights, computational and memory expensive

Can we better utilize the “image” features and reduce the number of weights?

CNN: Translation Invariant

Even if the input image shifts, the output (classification) stays unchanged.

- A CAT is still a CAT!



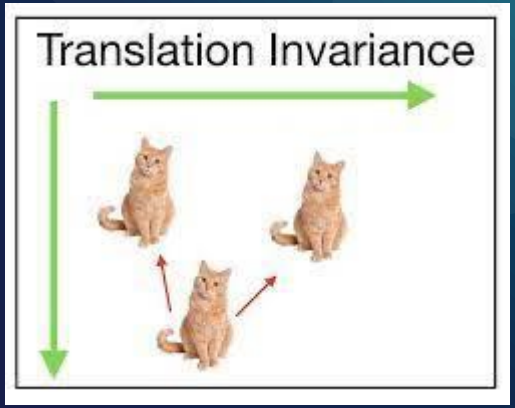
CNN: Translation Invariant

Even if the input image shifts, the output (classification) stays unchanged.

- A CAT is still a CAT!

Useful feature learned in training

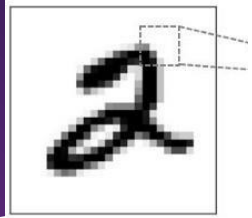
- Detectors for that feature useful in all locations during testing.



CNN: Translation Invariant

Translation Invariant: if a detector (filter) learnt a useful feature to detect 'CAT', it will capture 'CAT' wherever its location is in an image at testing time

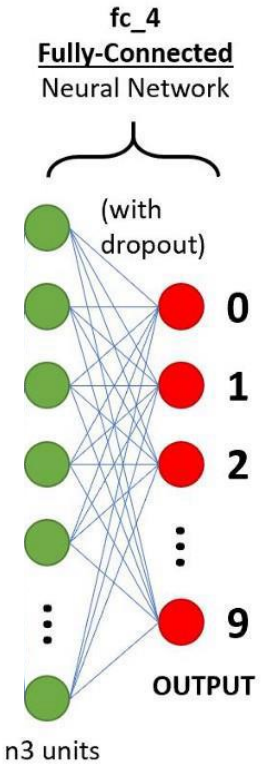
AlexNet architecture



INPUT

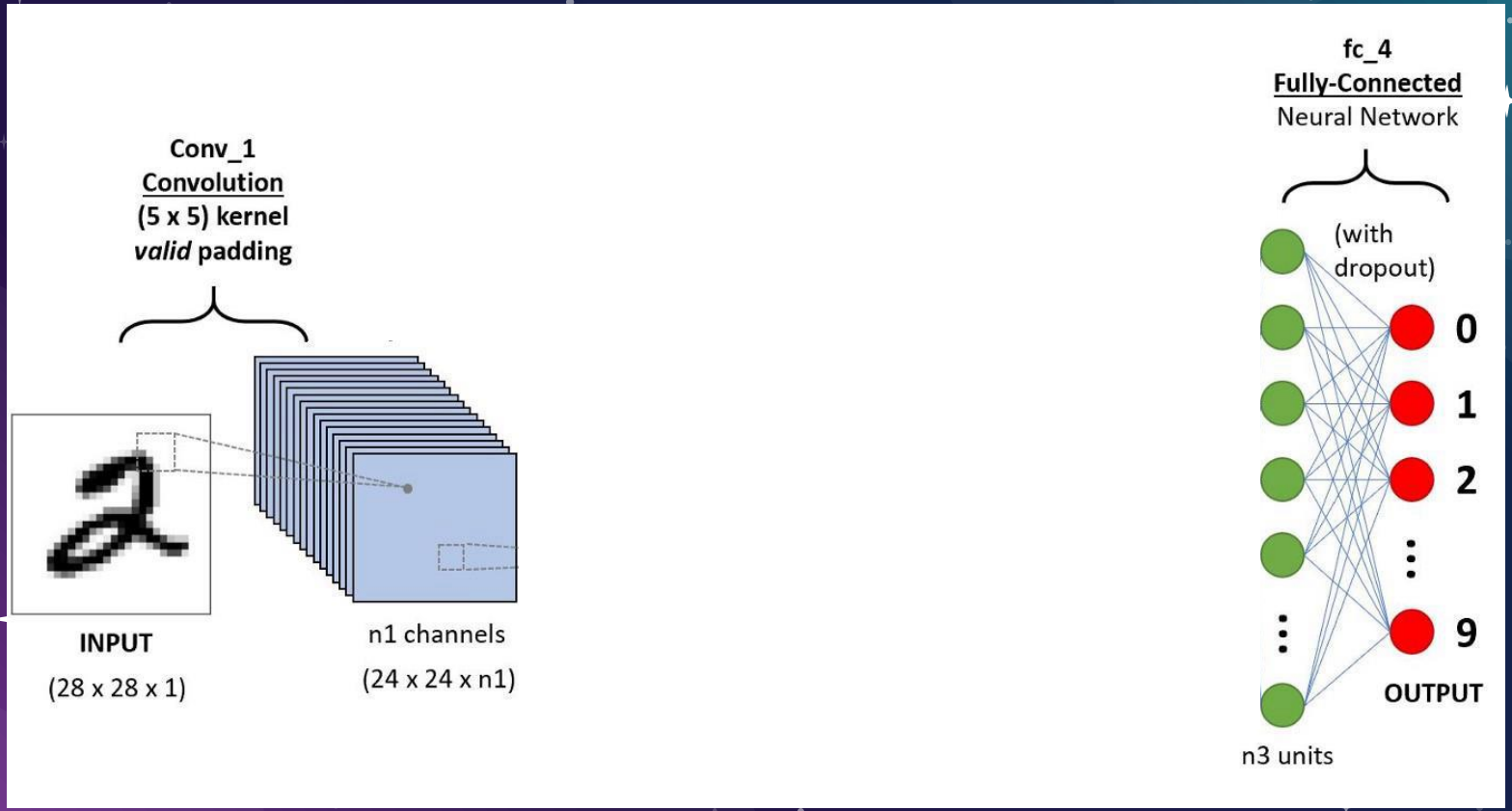
(28 x 28 x 1)

?



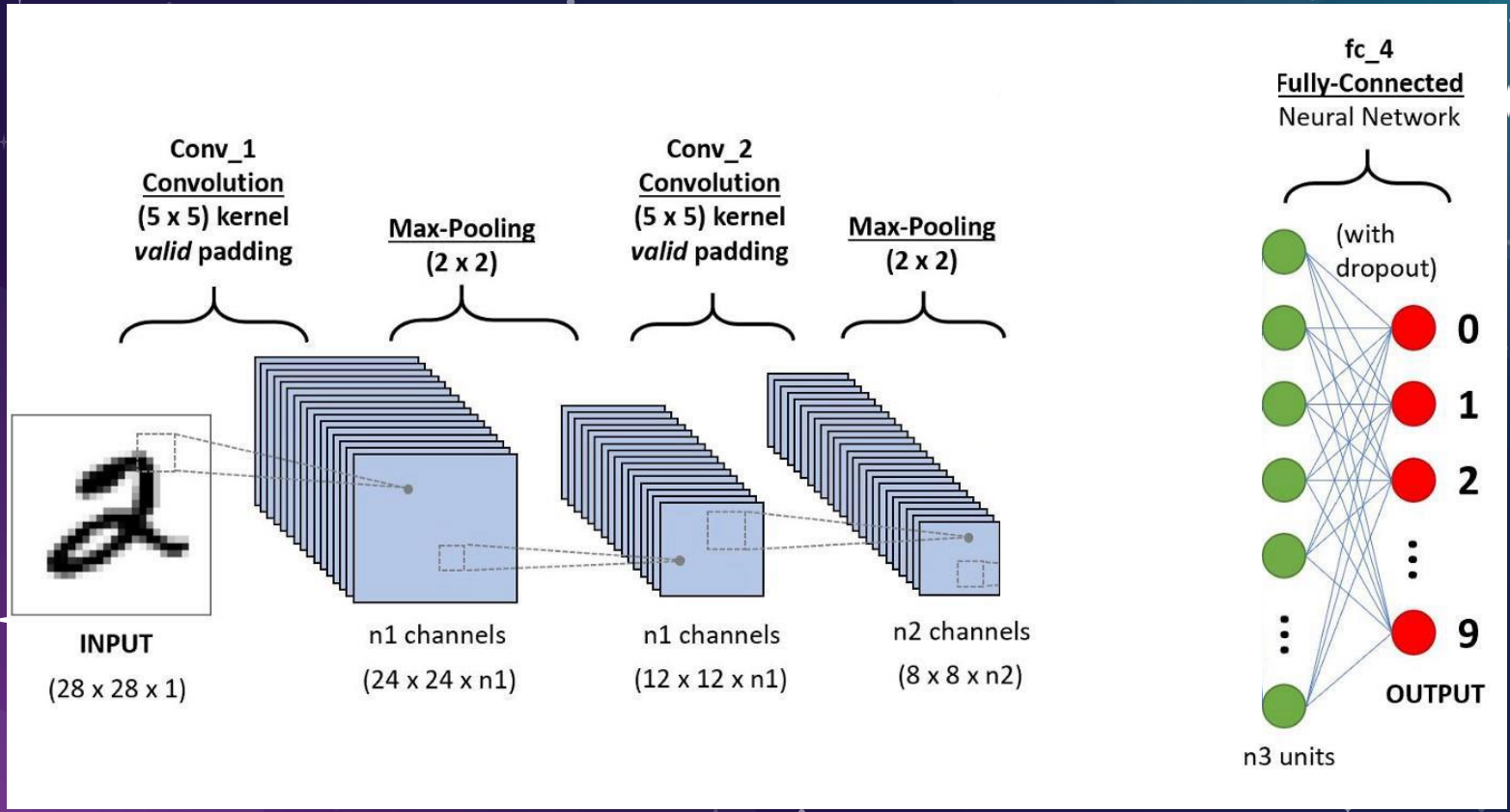
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

AlexNet architecture



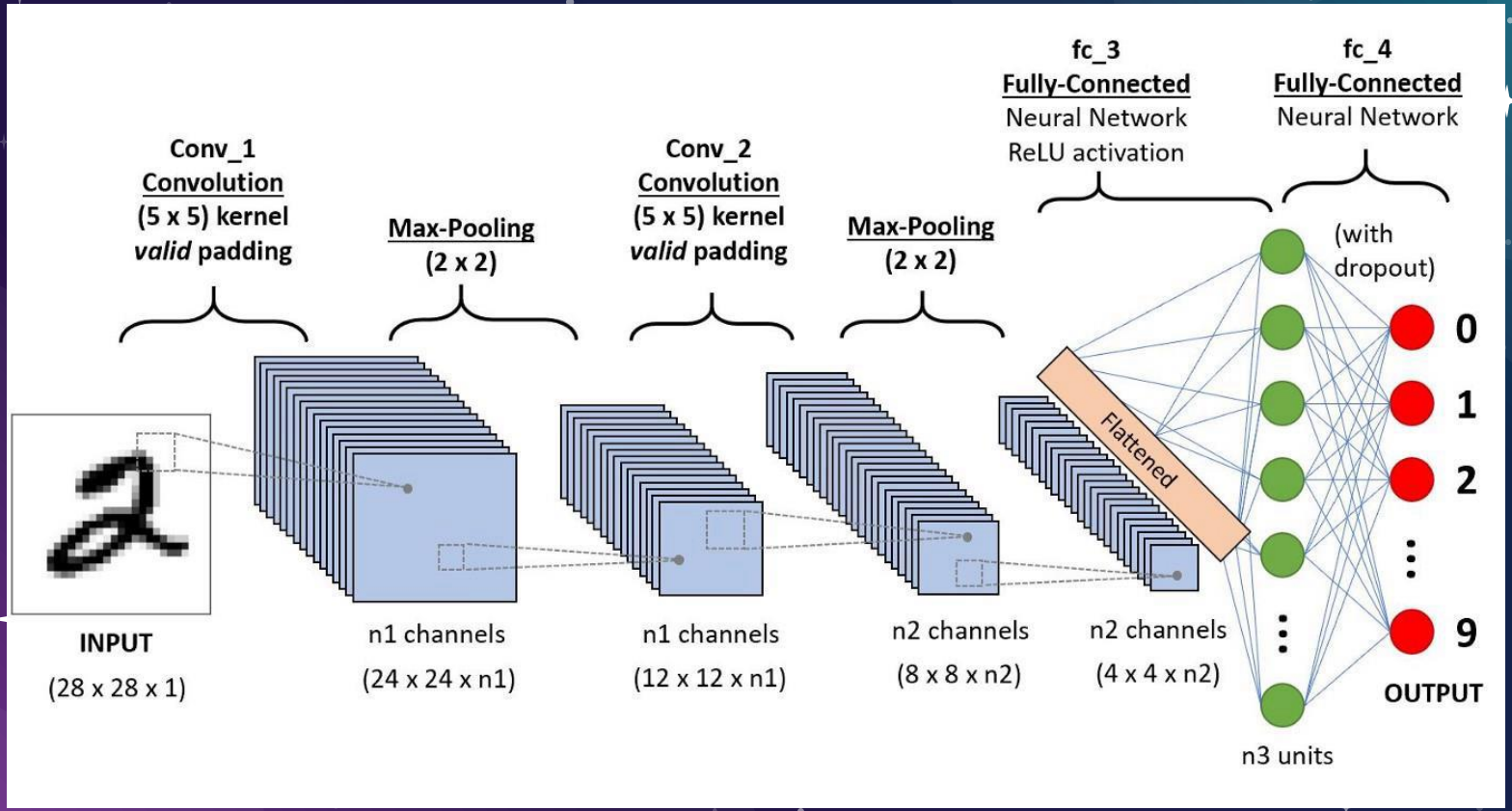
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

AlexNet architecture



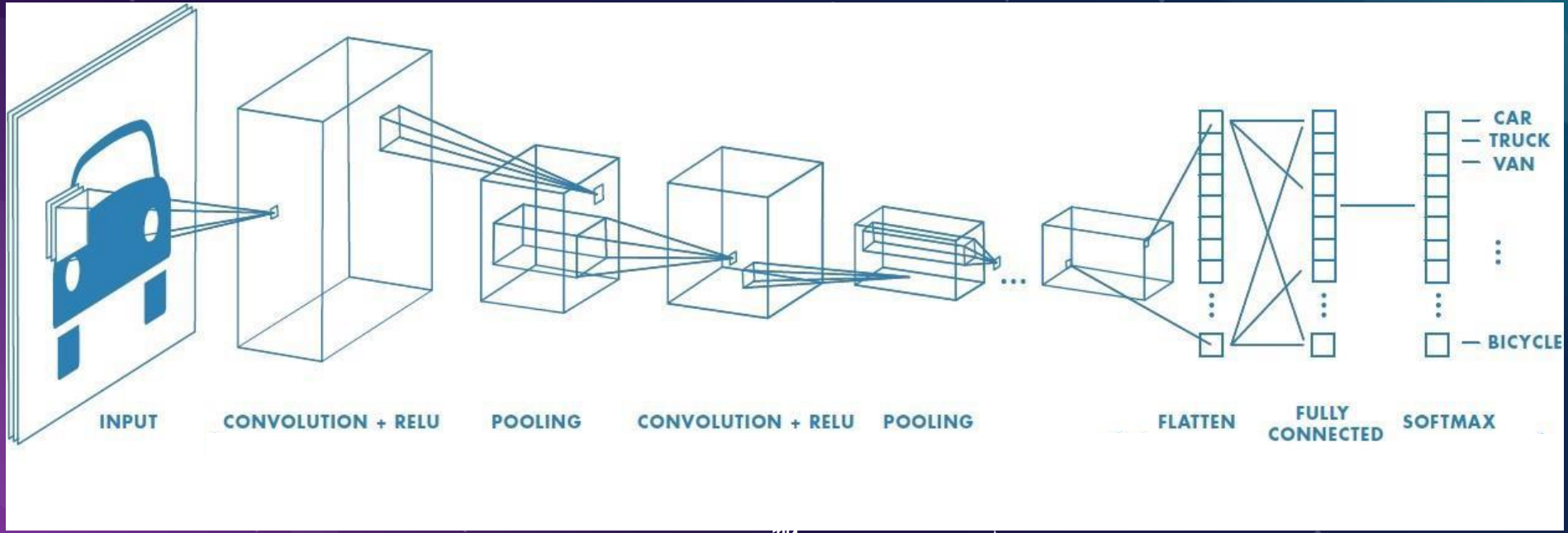
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

AlexNet architecture



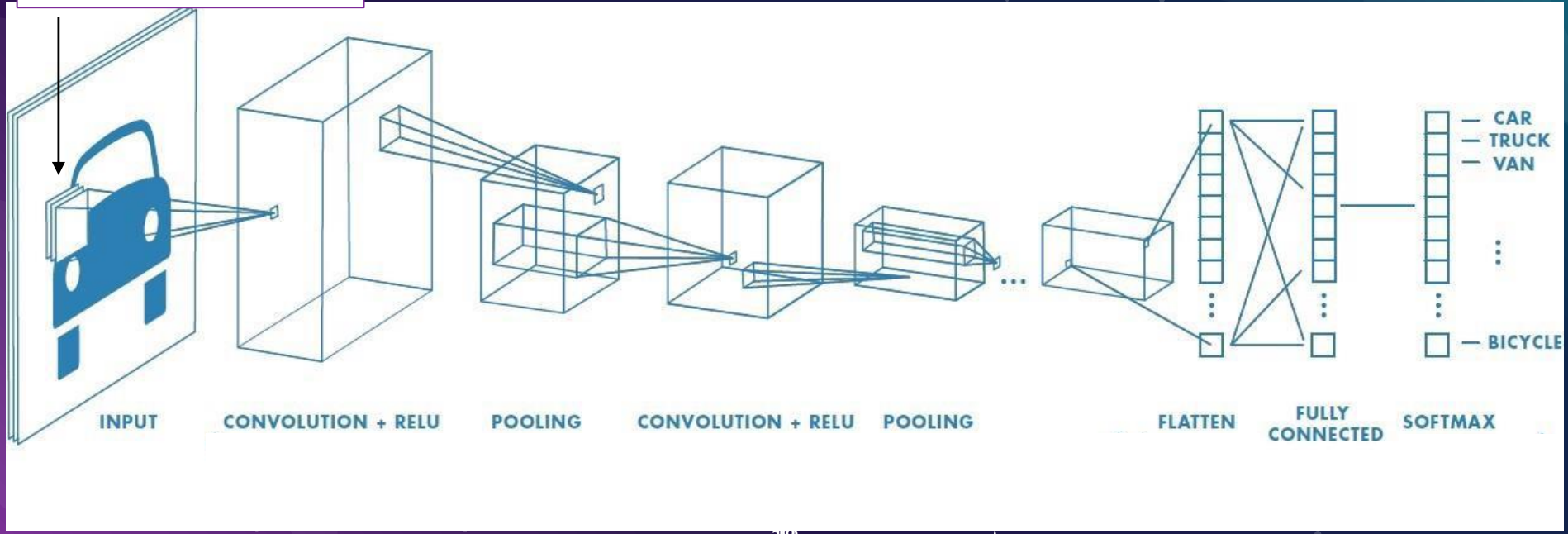
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNN Terminologies

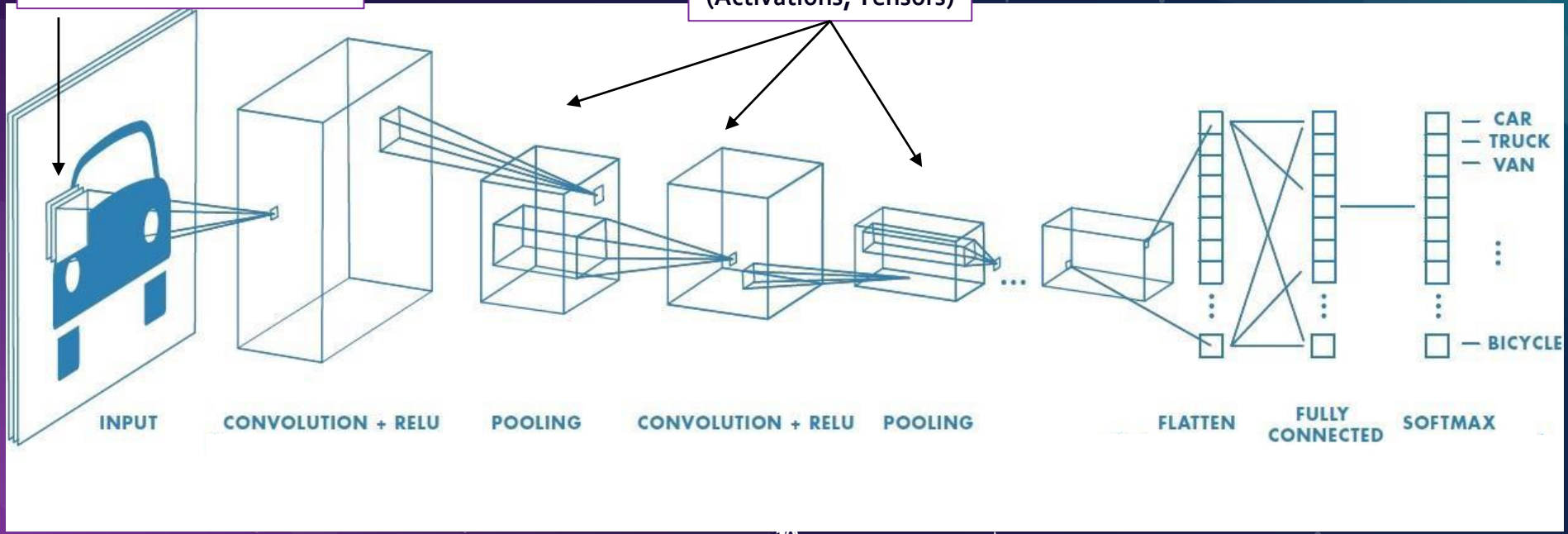


CNN Terminologies

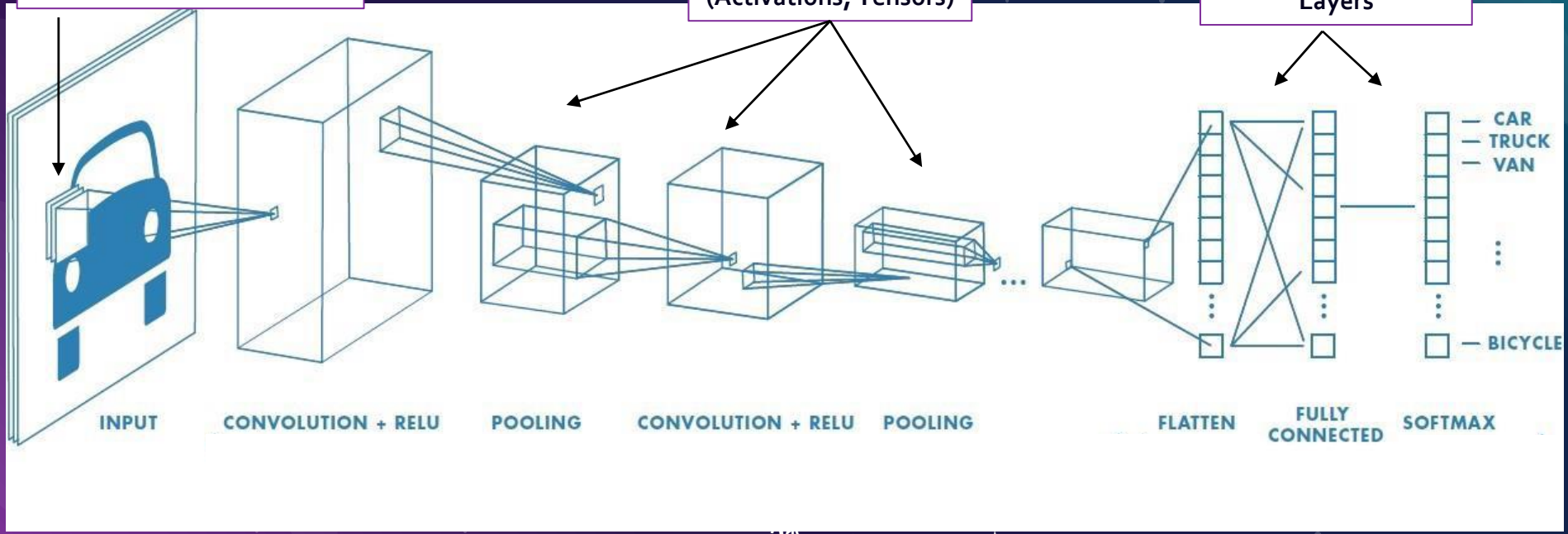
Filter (Kernel) 3x3, 5x5, etc.
Receptive Field



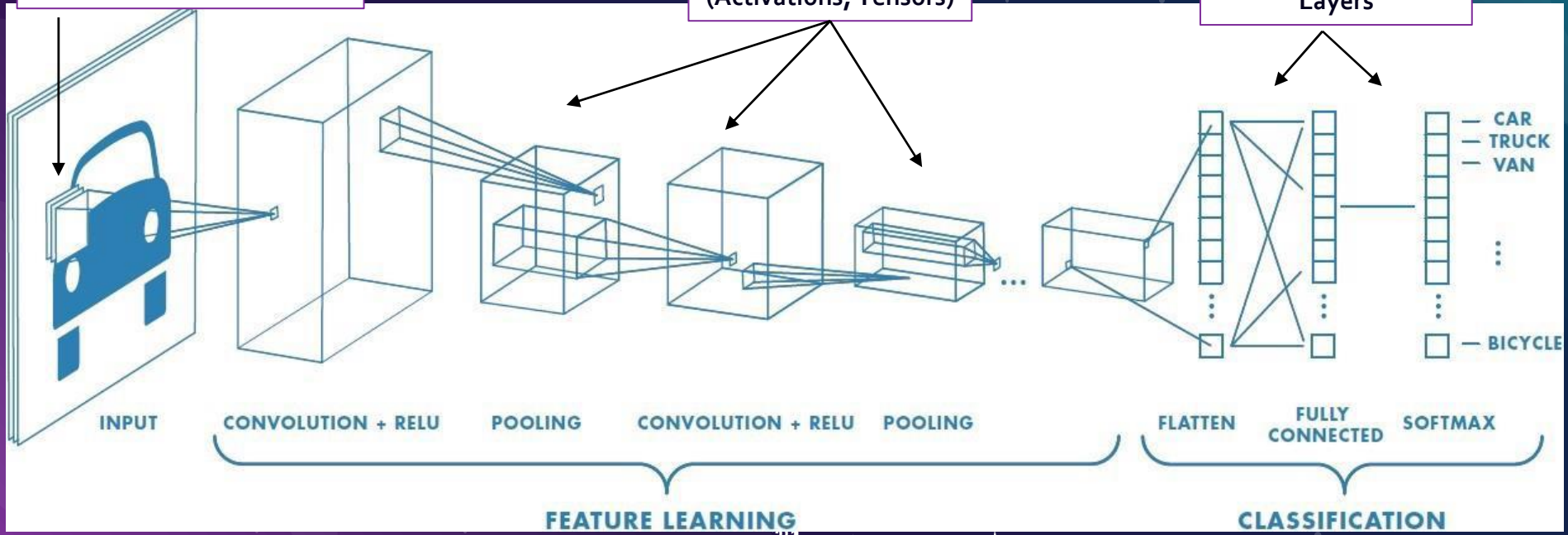
CNN Terminologies

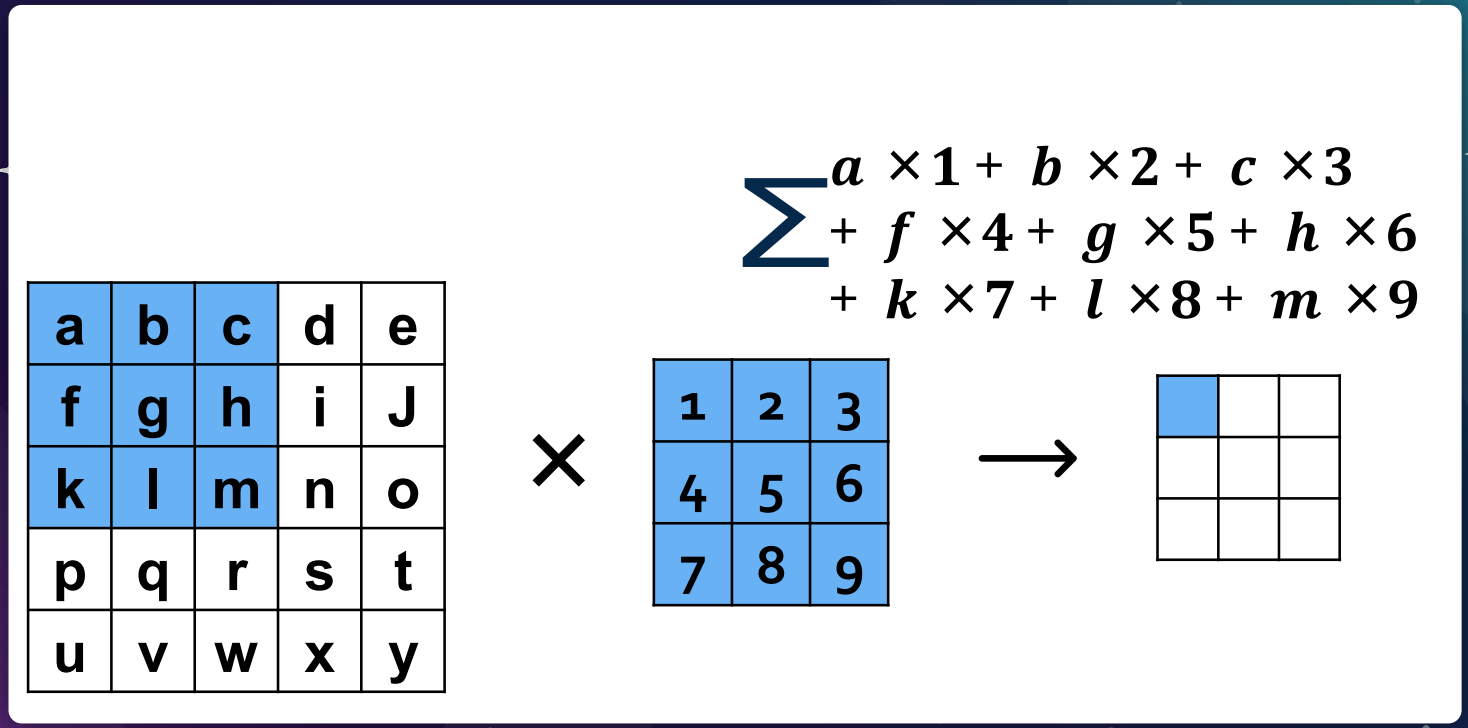


CNN Terminologies

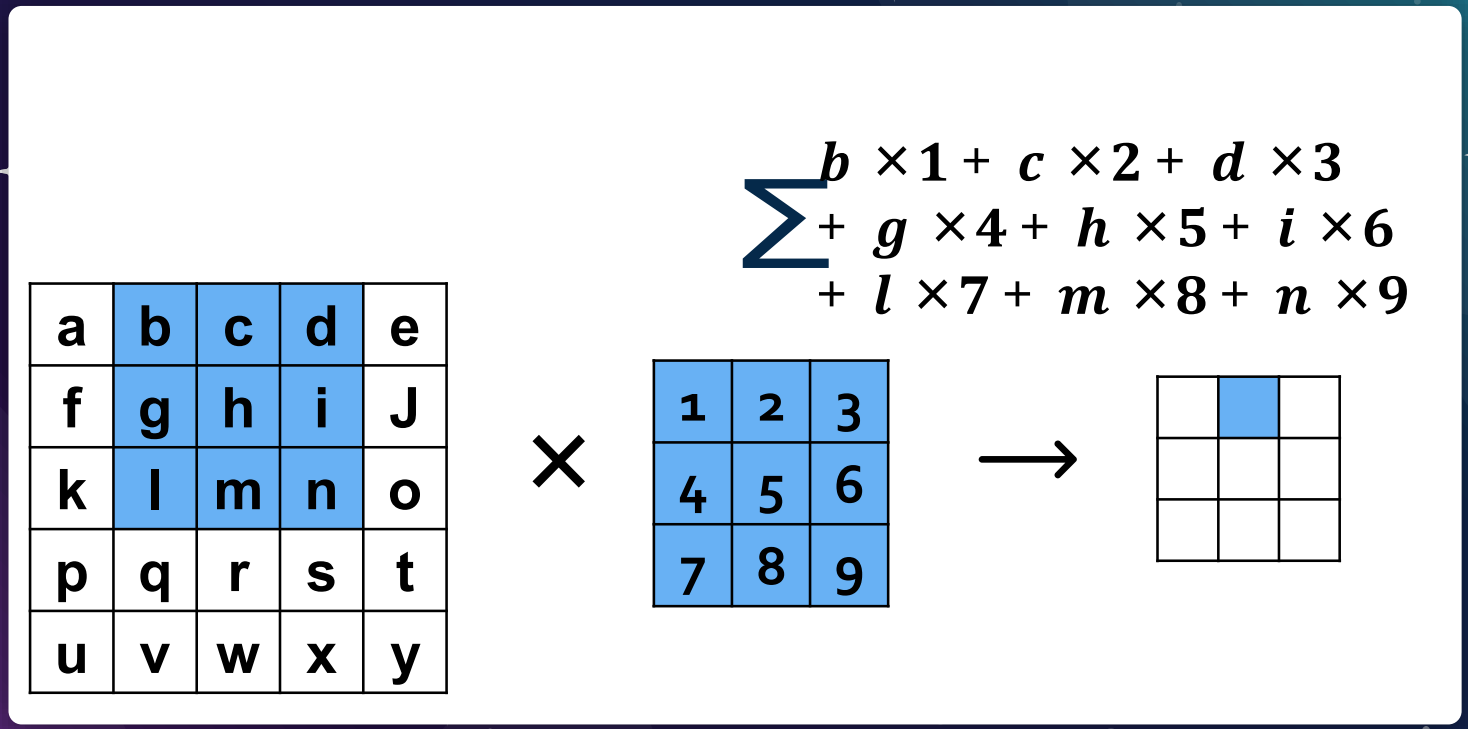


CNN Terminologies





A Closer Look at CNN Computation



A Closer Look at CNN Computation

One input channel

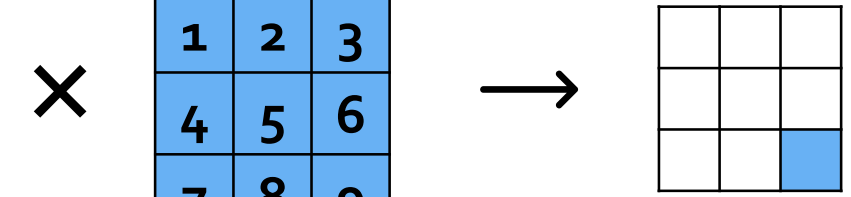
a	b	c	d	e
f	g	h	i	J
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

$$\sum m \times 1 + n \times 2 + o \times 3 + r \times 4 + s \times 5 + t \times 6 + w \times 7 + x \times 8 + y \times 9$$

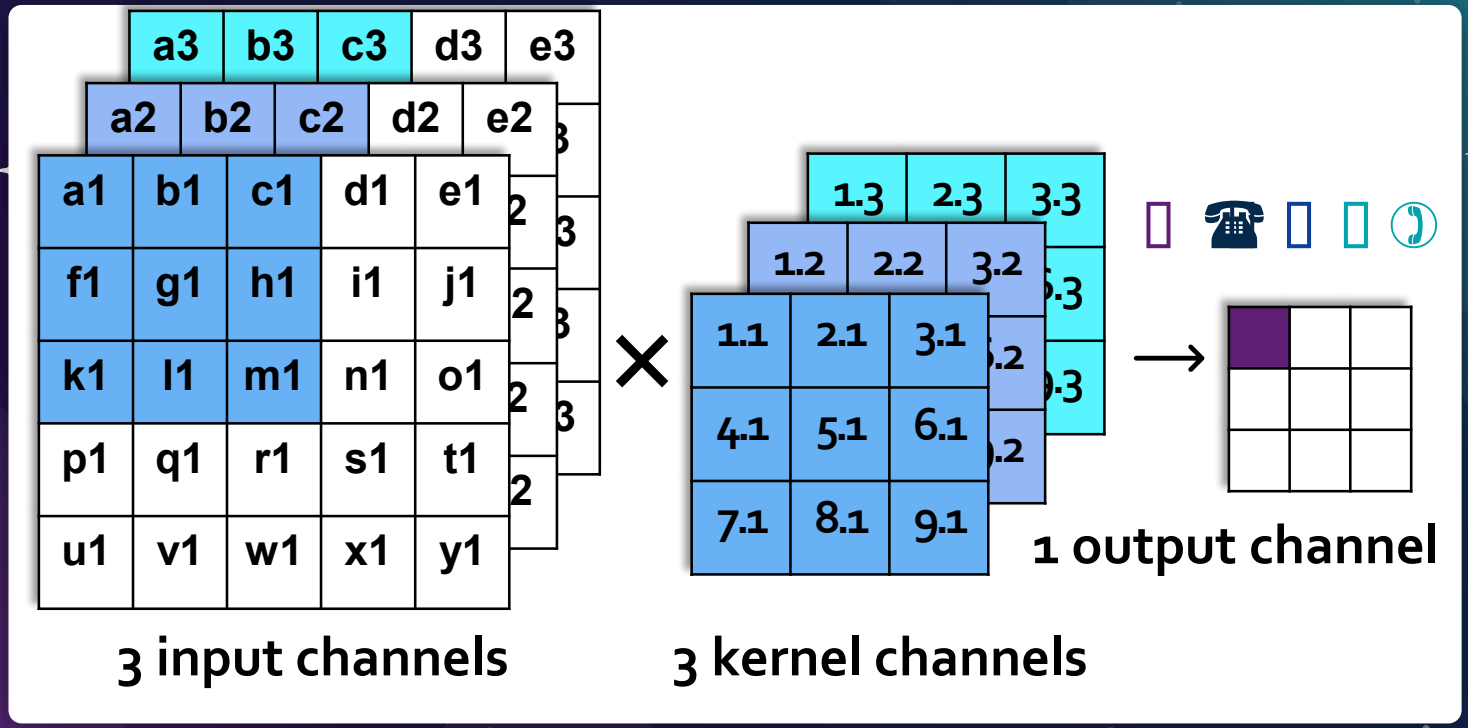
One kernel channel

1	2	3
4	5	6
7	8	9

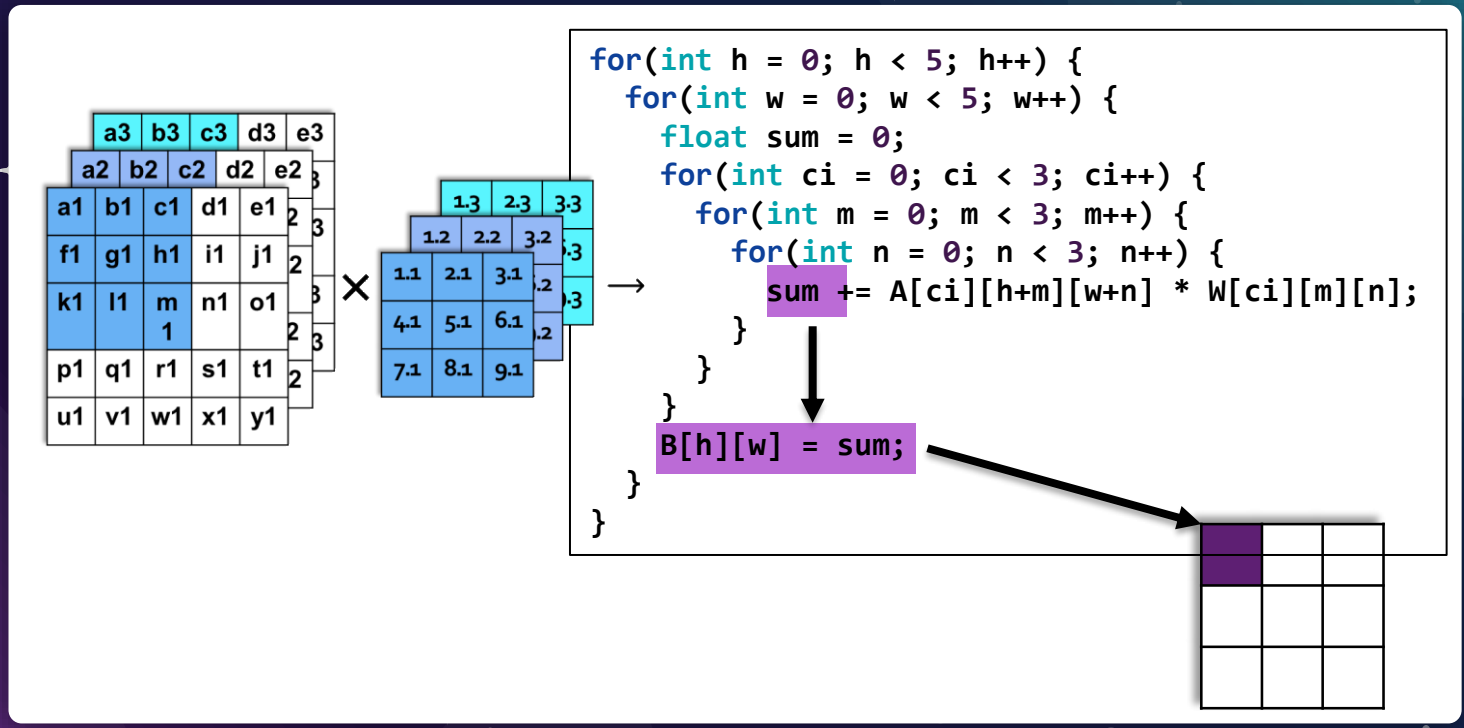
One output channel



A Closer Look at CNN Computation



A Closer Look at CNN Computation w/ multiple channels



A Closer Look at CNN Computation w/ multiple channels


1.5
Practical
https://t.ly/tLW-J

2.
Recurrent Neural Networks

Recurrent Neural Network

- So far, the neural network only considers the current input. What about time?

Recurrent Neural Network

- So far, the neural network only considers the current input. What about time?
  - Make the neural network considers the previous output together with the current input

Recurrent Neural Network

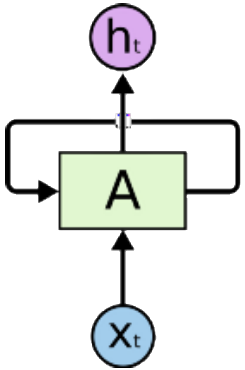
- A class of neural network with backward edges
  - Can learn sequential information
  - Indirectly factor in all previous inputs

Recurrent Neural Network

- A class of neural network with backward edges
  - Can learn sequential information
  - Indirectly factor in all previous inputs

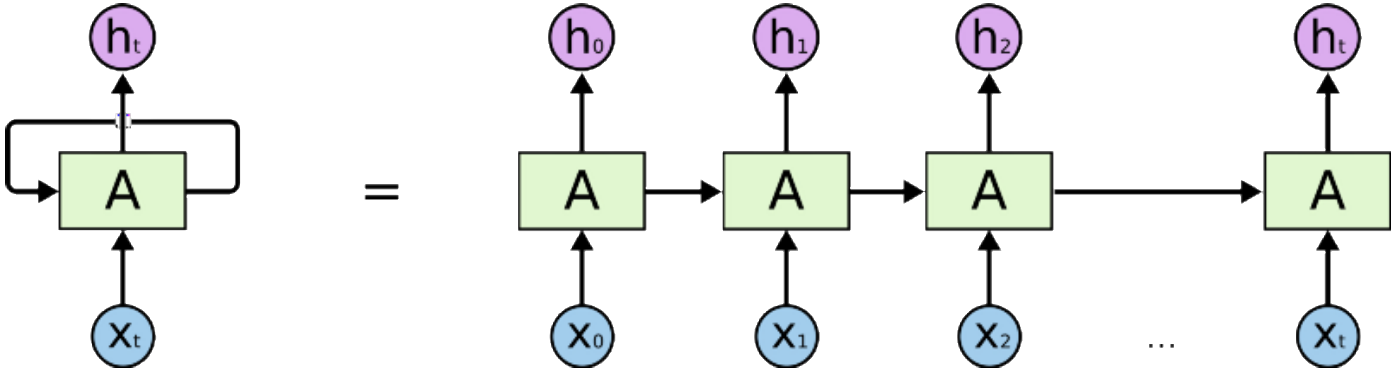
Vanishing effect

Short-term memory



Recurrent Neural Network

https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN



Unfolding of three time steps

Recurrent Neural Network

https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN

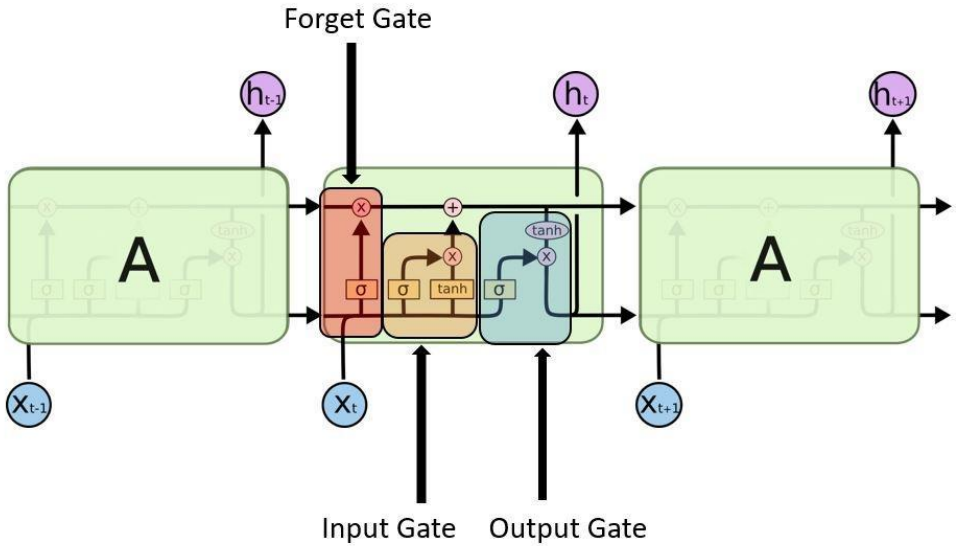
Long Short-Term Memory (LSTM)

- How about long-term memory?
- Can a neural network learn the pattern of both type of memories (Long + Short) ?

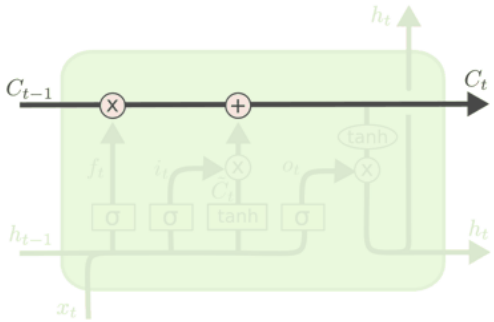
Long Short-Term Memory (LSTM)

IDEA:

Instead of only using the current input and previous output, also add a memory component



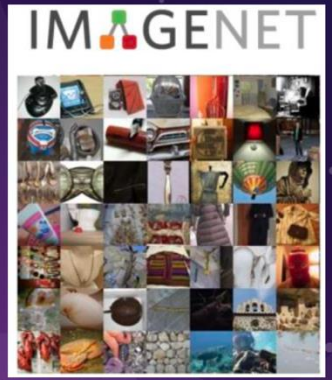
Long Short-Term Memory (LSTM)

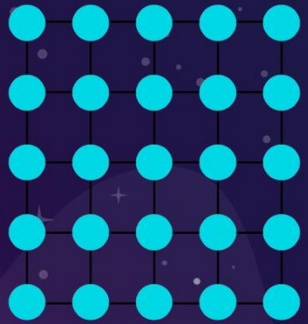

h_t
C_{t-1}
C_t
tanh
f_t
i_t
$\tilde{C}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
x_t

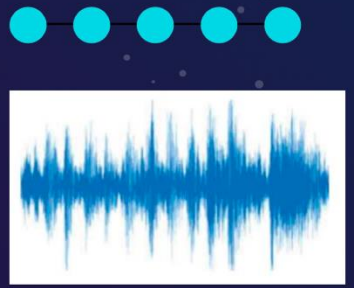
3.
Graph Neural Networks

Graph Neural Network (GNN)

Traditional neural networks are designed for simple sequences & grids







Graph Neural Network (GNN)

Reality: A lot of real-world data does not “live” on grids

- ★ Arbitrary size and complex topological structure
- ★ No fixed node ordering or reference point
- ★ Often dynamic and have multimodal features

Graph Neural Network (GNN)

Image credit: Medium

Social Networks

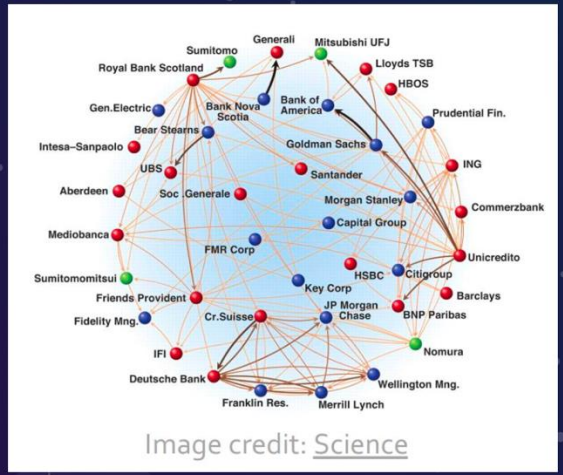
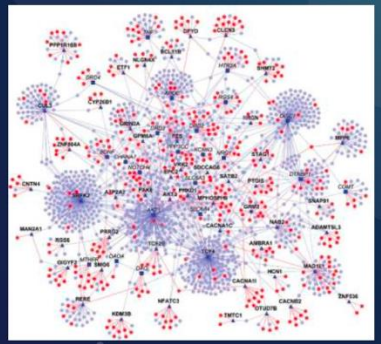


Image credit: Science

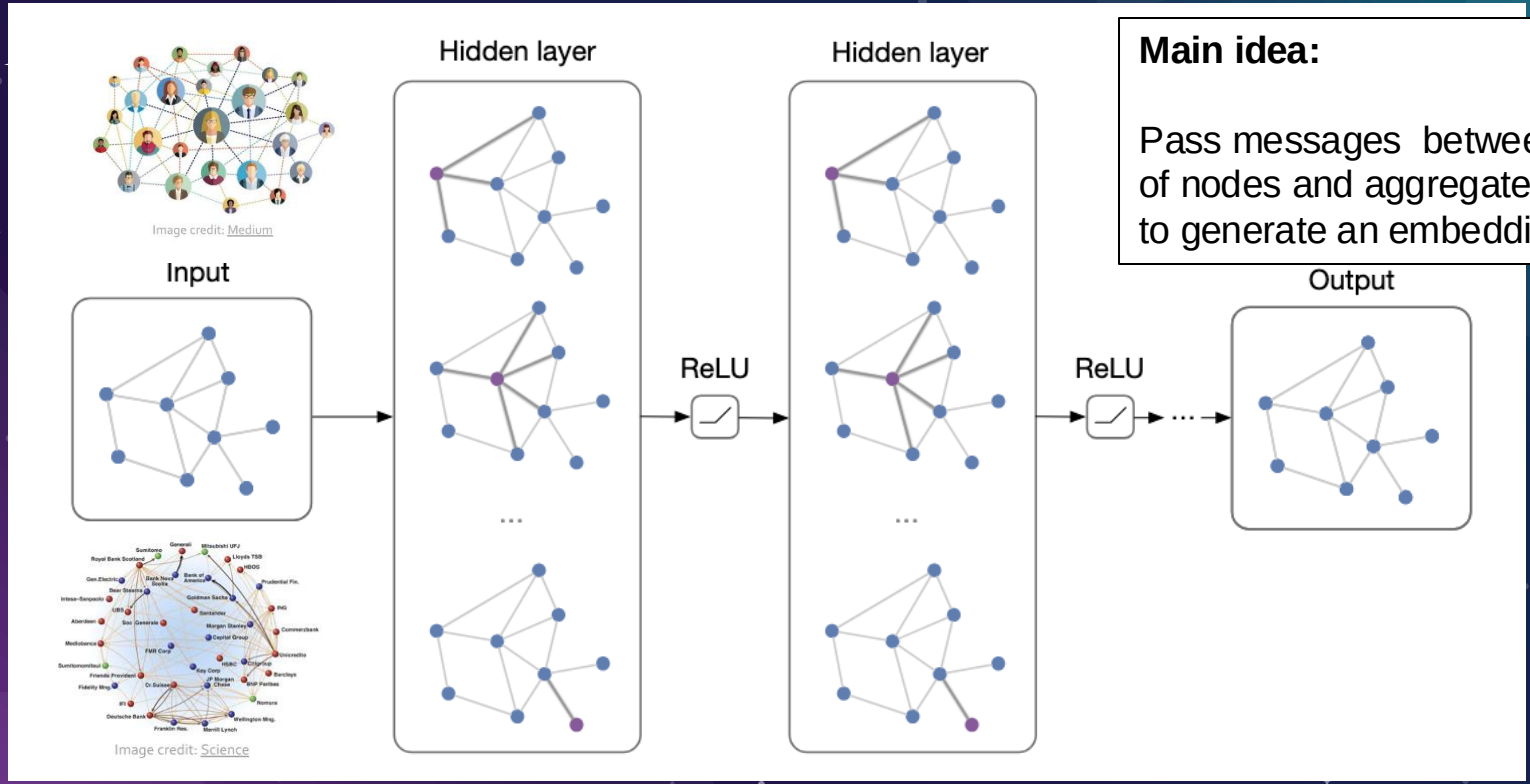
Economic Networks



[Madhavicmu]

Protein Interaction Networks

Graph Neural Network (GNN)



Main idea:

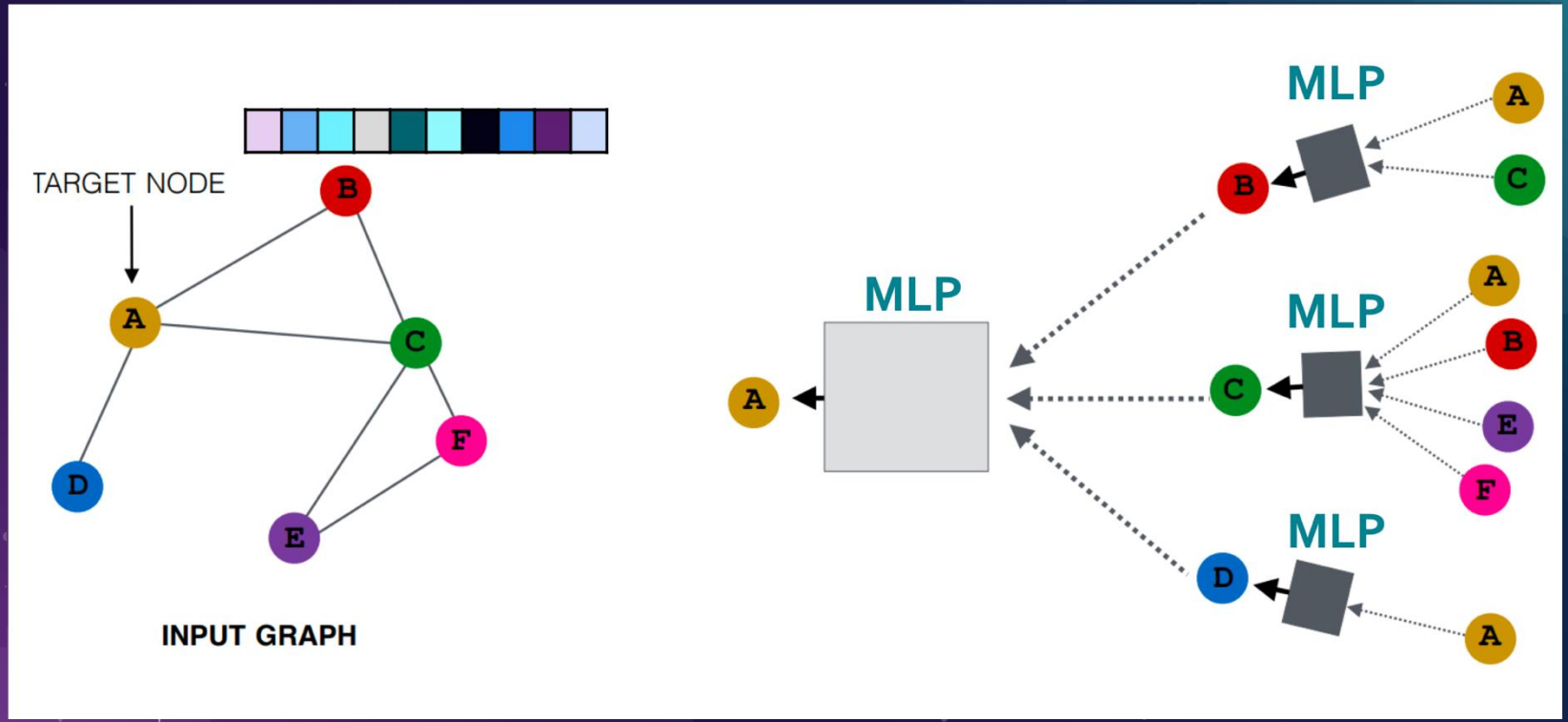
Pass messages between pairs of nodes and aggregate them to generate an embedding

How to compute GNNs?

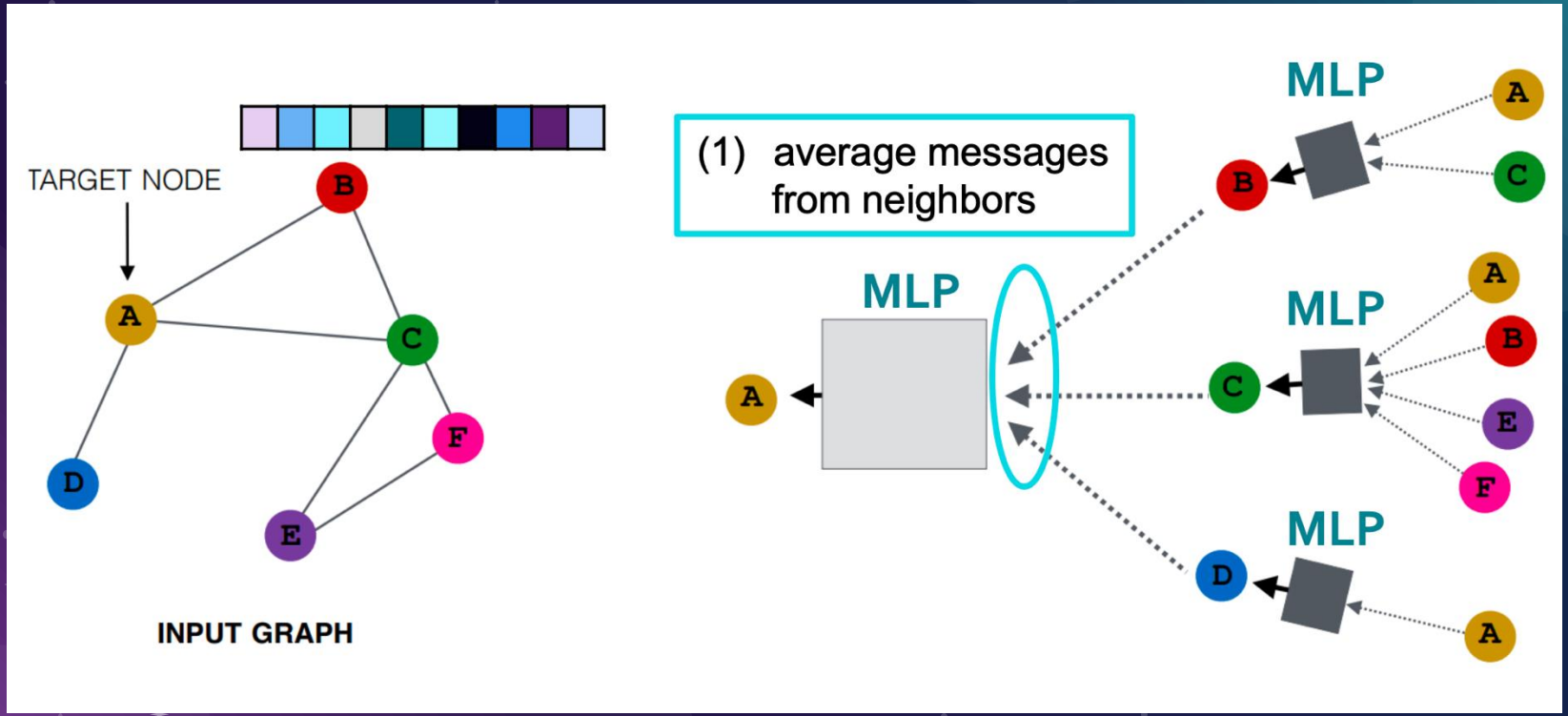
Key idea: Generate node embeddings from local network neighborhoods

Node embedding: a vector representing the node features

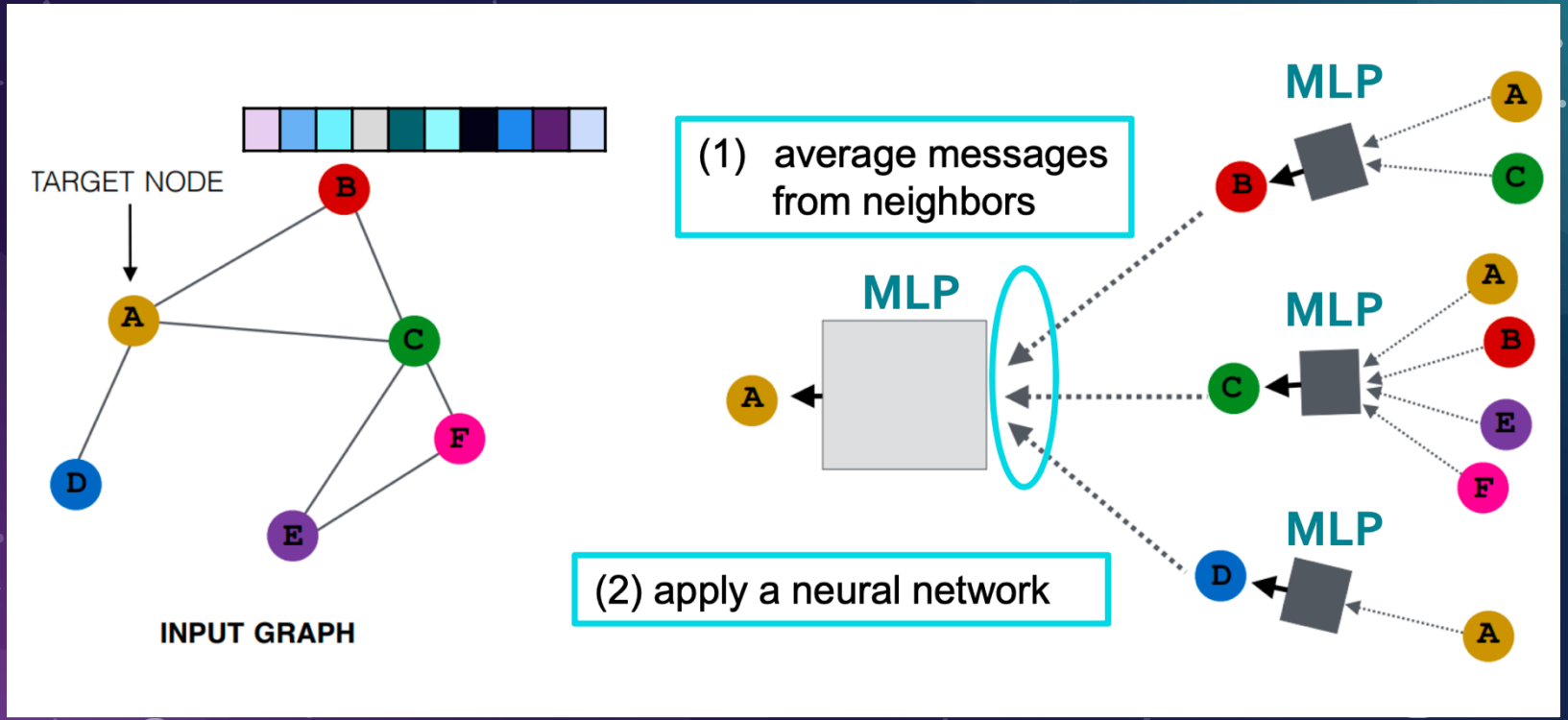
How to compute GNNs?



How to compute GNNs?



How to compute GNNs?



How to compute GNNs?

Initial 0-th layer embeddings are equal to node features

$$h_v^0 = x_v$$

How to compute GNNs?

Initial 0-th layer embeddings are equal to node features

$$\mathrm{h}_v^0 = \mathrm{x}_v$$

$$\mathrm{h}_v^{(k+1)} = \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(k)}}{|\mathrm{N}(v)|}$$

Average of neighbor's previous layer embeddings

h_v^k: the hidden representation of node v at layer k

- W_k: weight matrix for neighborhood aggregation
- B_k: weight matrix for transforming hidden vector of self

Notice summation is a permutation invariant pooling/aggregation.

How to compute GNNs?

Initial 0-th layer embeddings are equal to node features

$$\mathrm{h}_v^0 = \mathrm{x}_v$$

$$\mathrm{h}_v^{(k+1)} = \sigma\left(\mathrm{W}_k \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(k)}}{|\mathrm{N}(v)|}\right)$$

Average of neighbor's previous layer embeddings

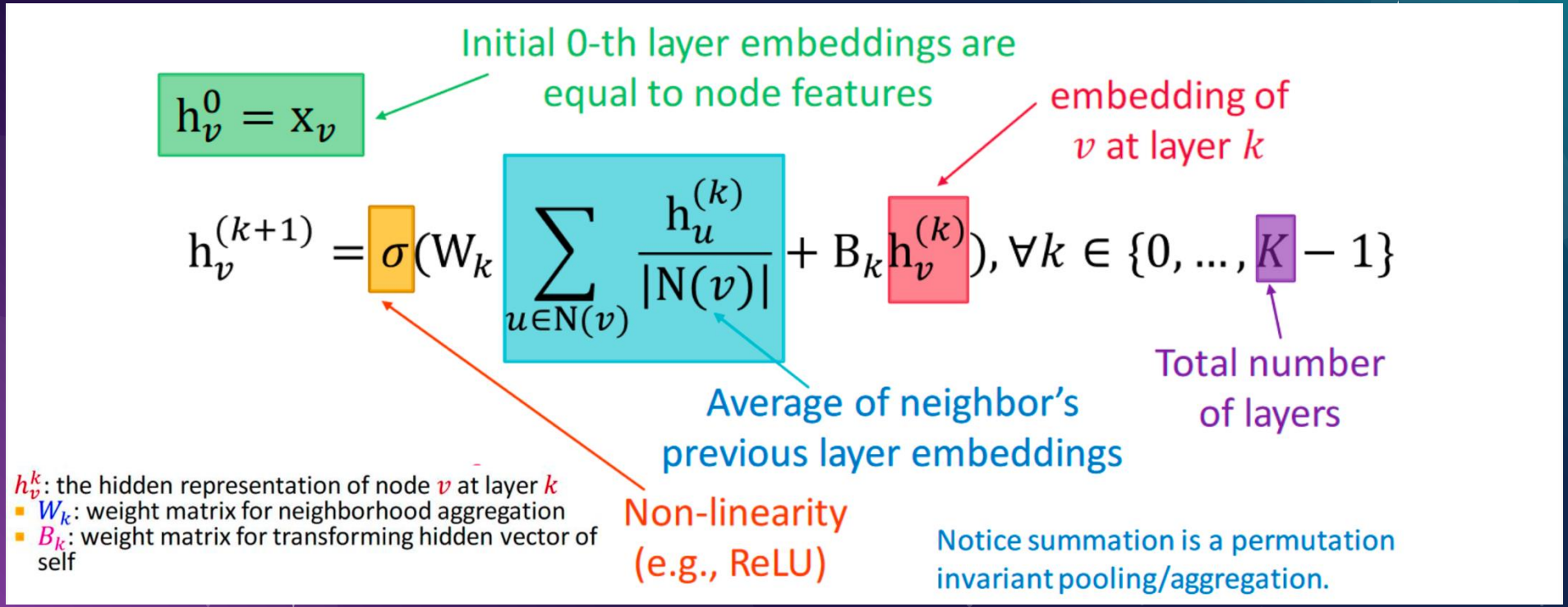
Non-linearity (e.g., ReLU)

h_v^k: the hidden representation of node v at layer k

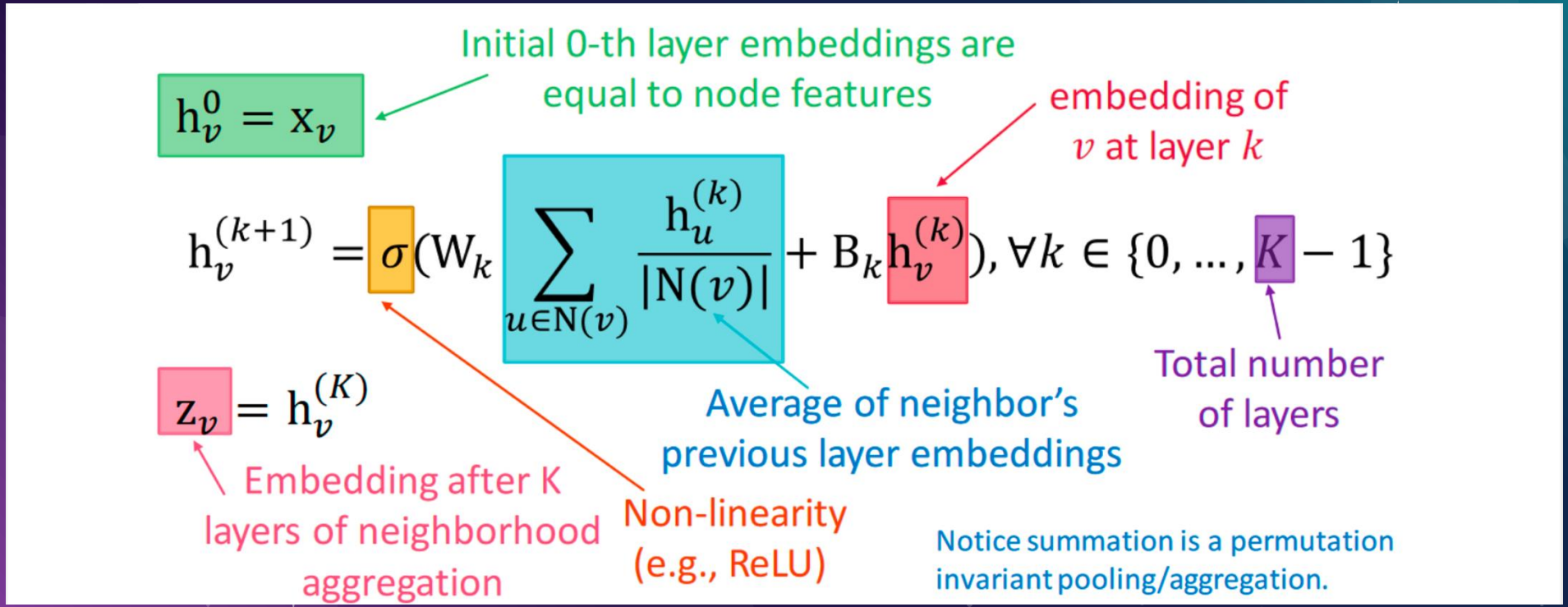
- W_k: weight matrix for neighborhood aggregation
- B_k: weight matrix for transforming hidden vector of self

Notice summation is a permutation invariant pooling/aggregation.

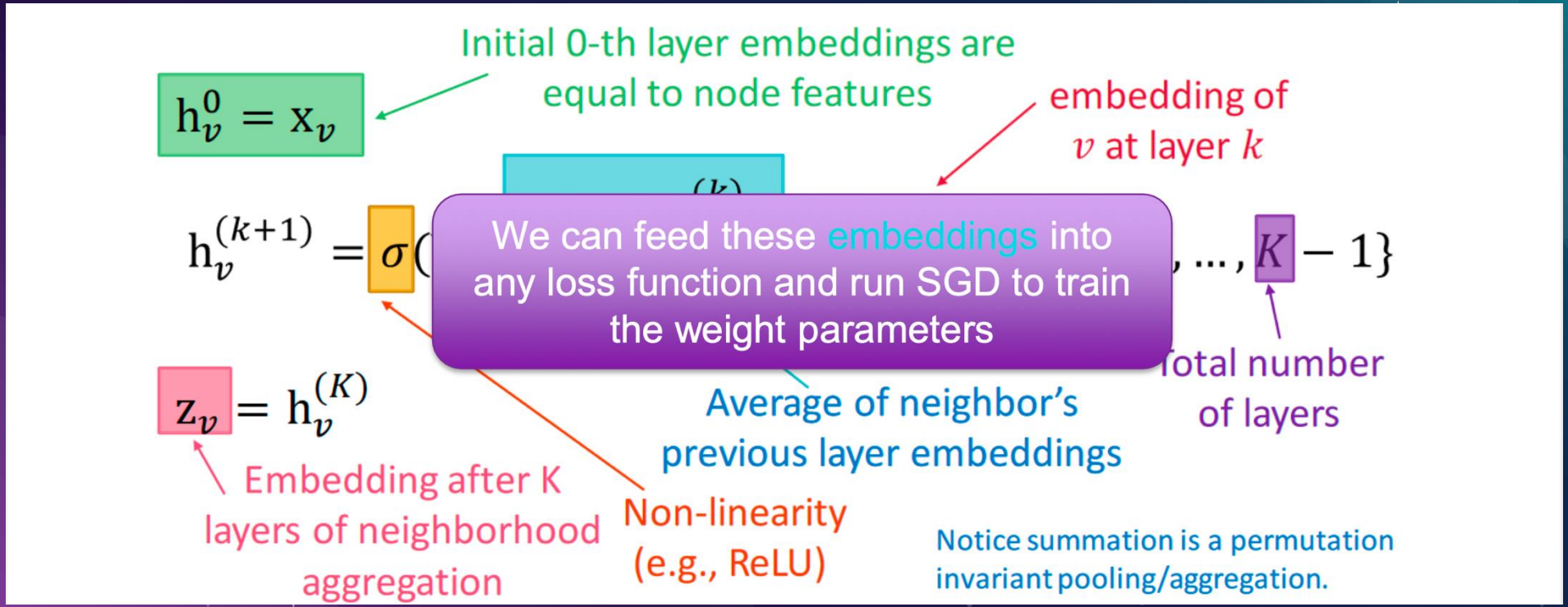
How to compute GNNs?

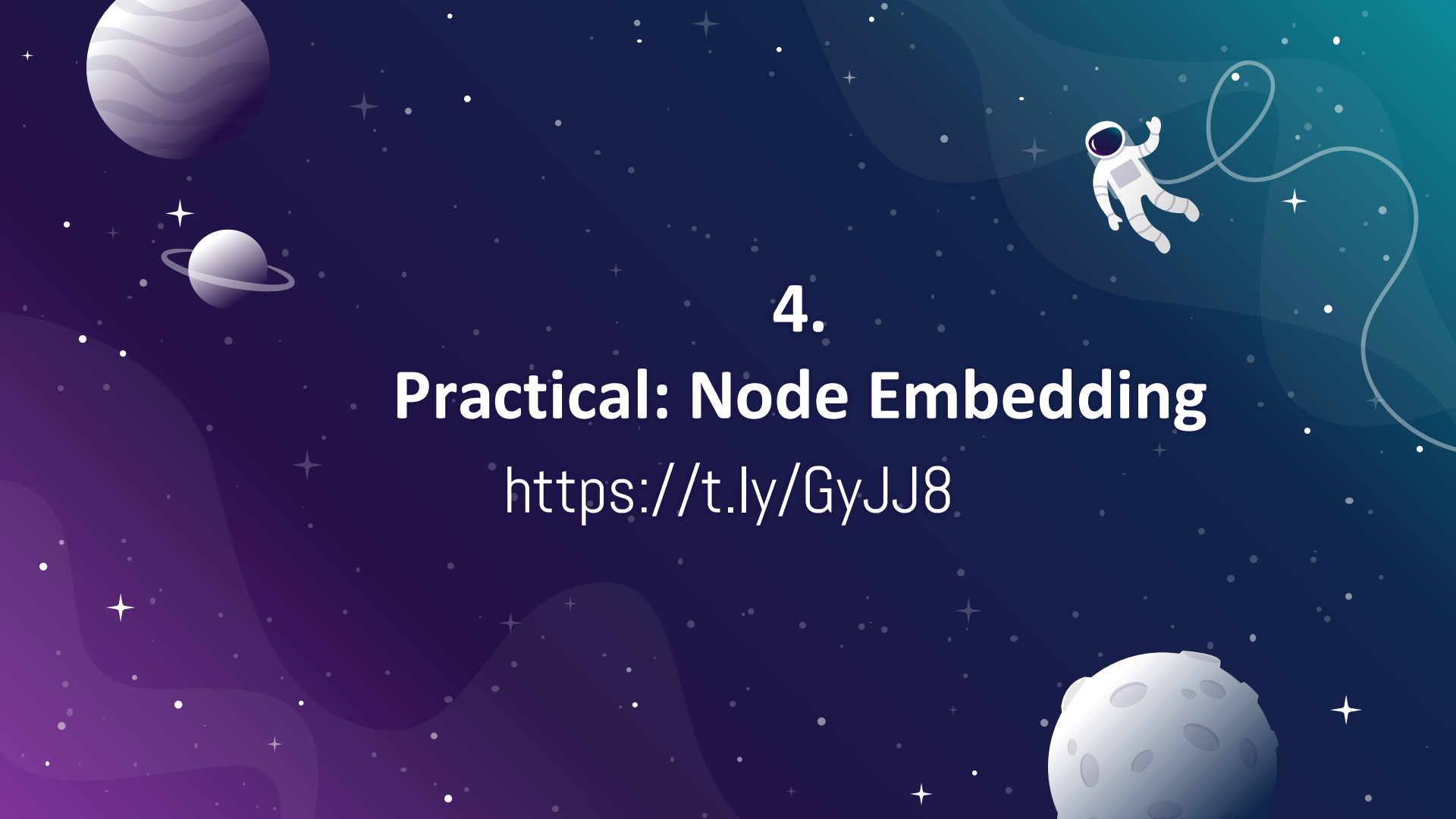


How to compute GNNs?



How to compute GNNs?



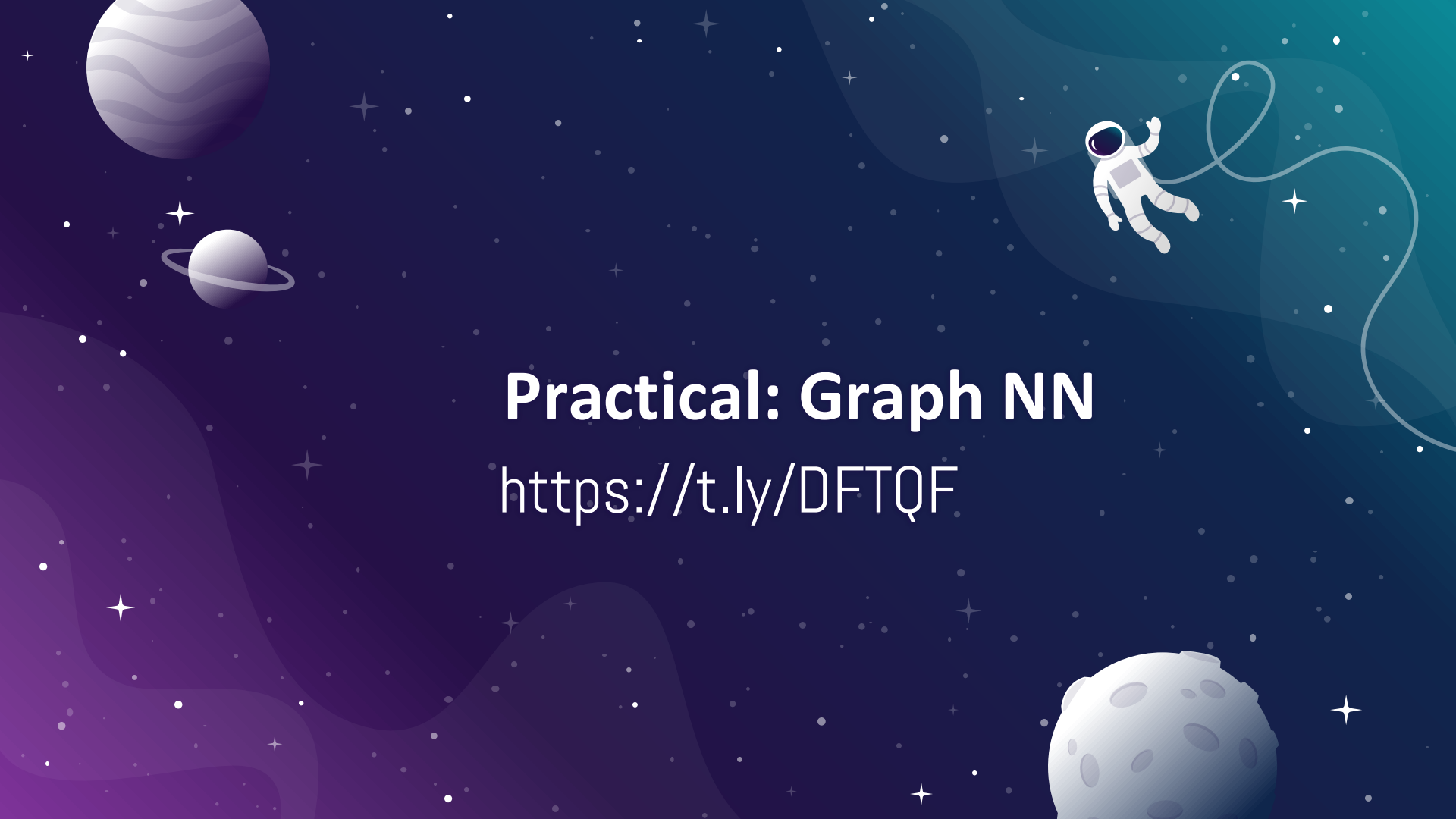


4.
Practical: Node Embedding

https://t.ly/GyJJ8


Practical: Node2Vec
https://t.ly/vC44k


Practical: Recommenders
https://t.ly/BUbL7

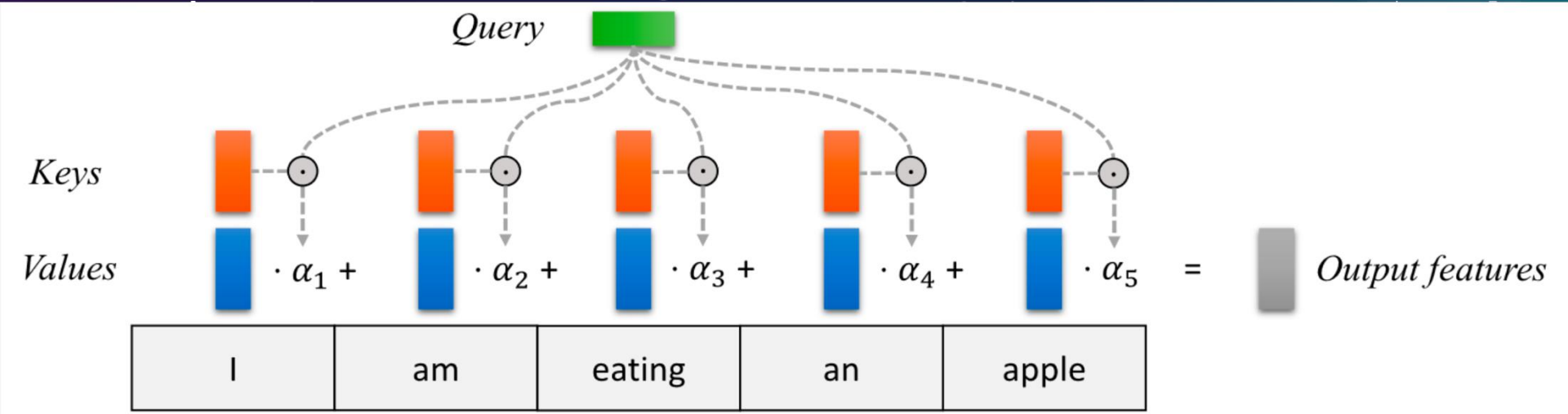

Practical: Graph NN
https://t.ly/DFTQF

5.
ATTENTION Neural Networks

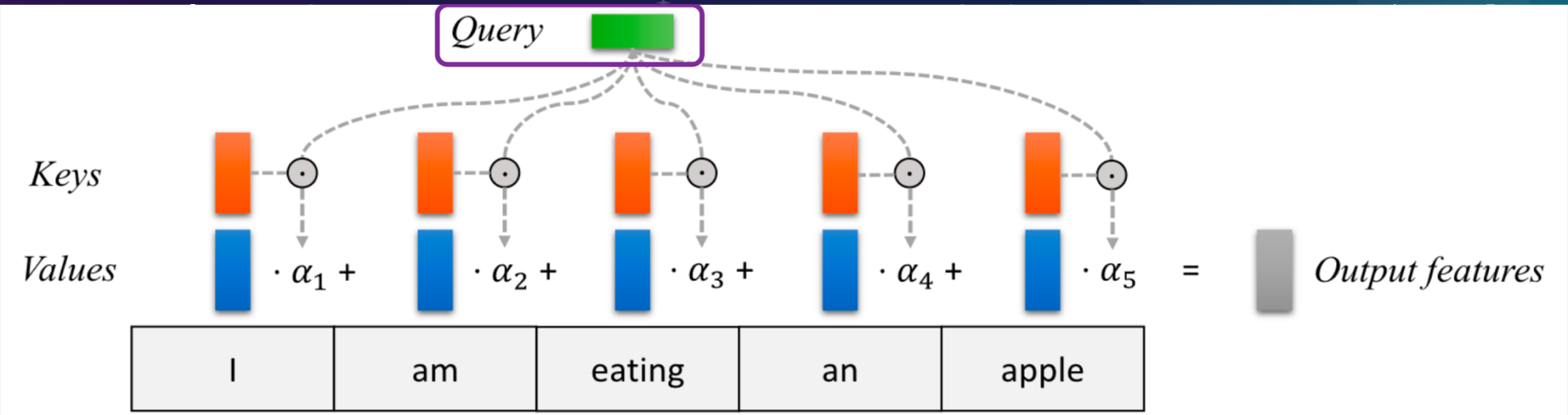
"

The attention mechanism describes a weighted average of (sequence) elements with the weights dynamically computed based on an input query and elements' keys

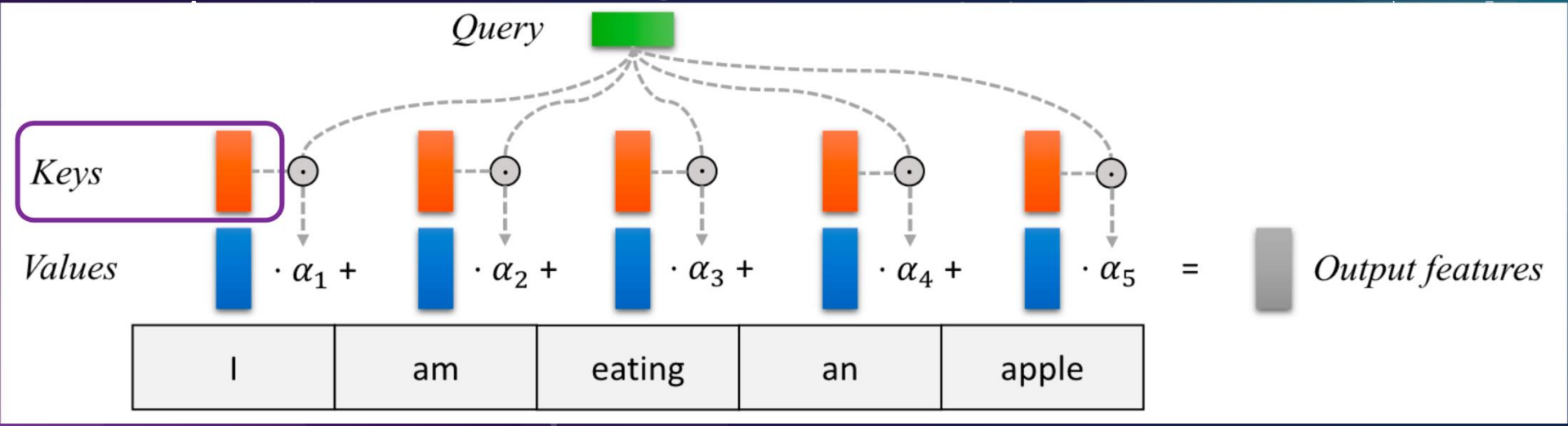
Attention Mechanism



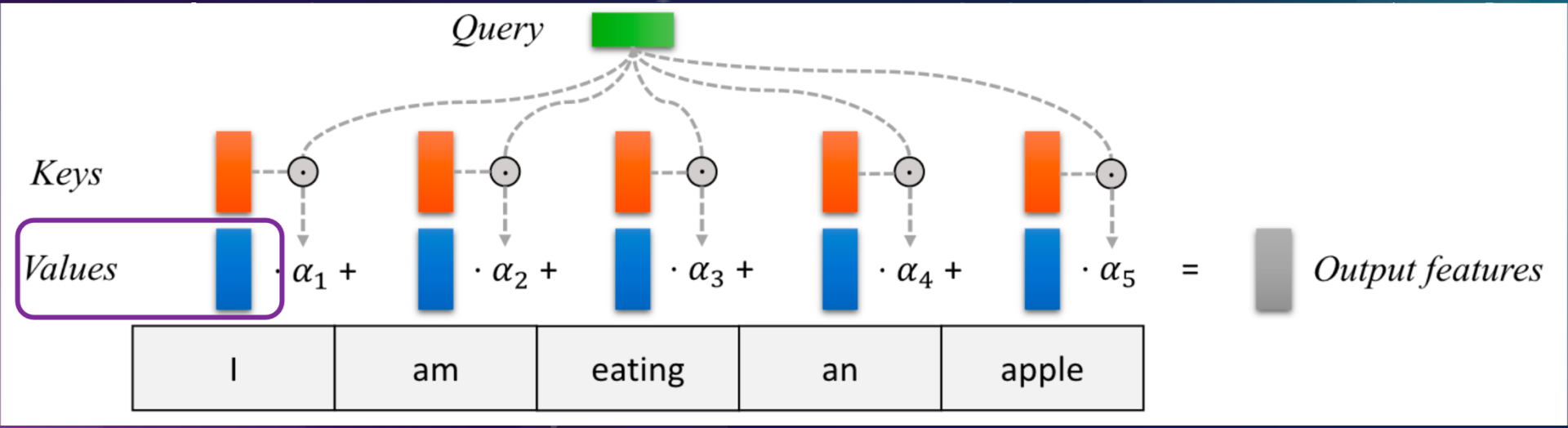
Attention Mechanism



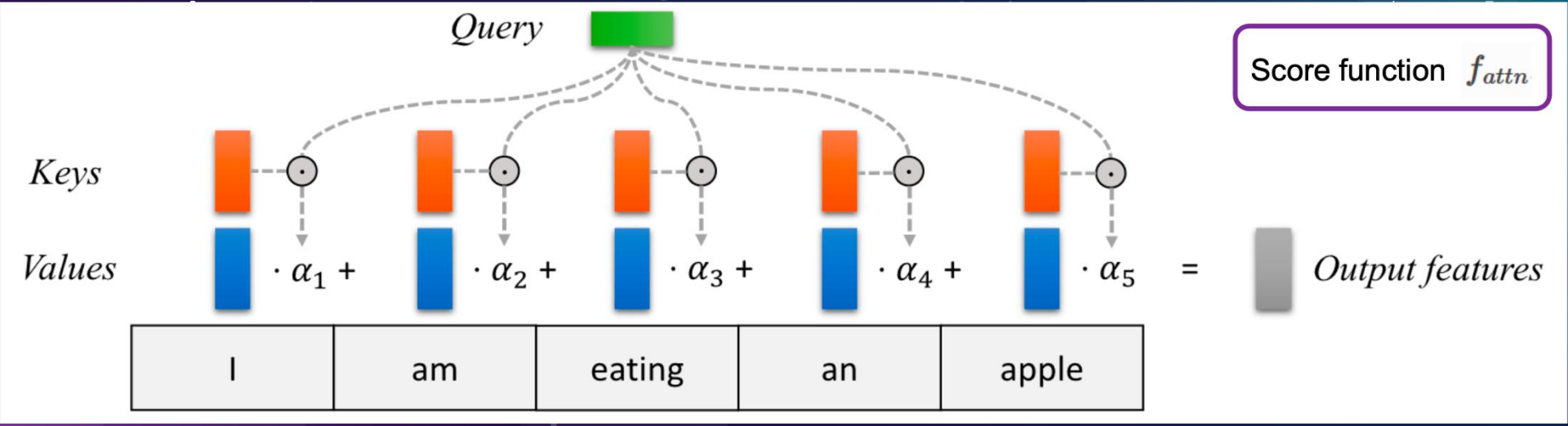
Attention Mechanism



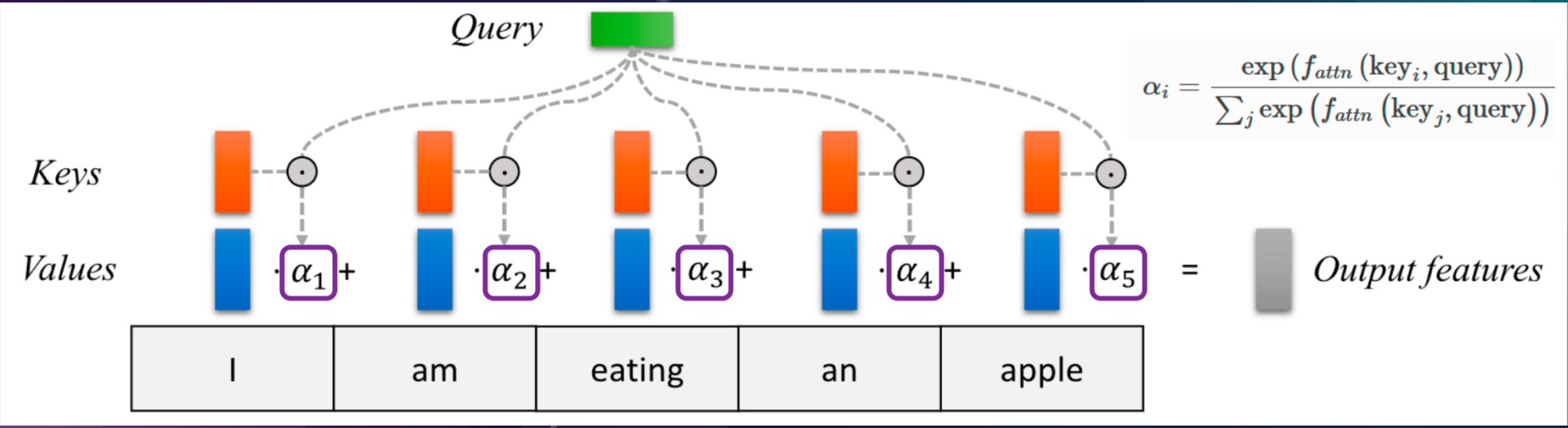
Attention Mechanism



Attention Mechanism

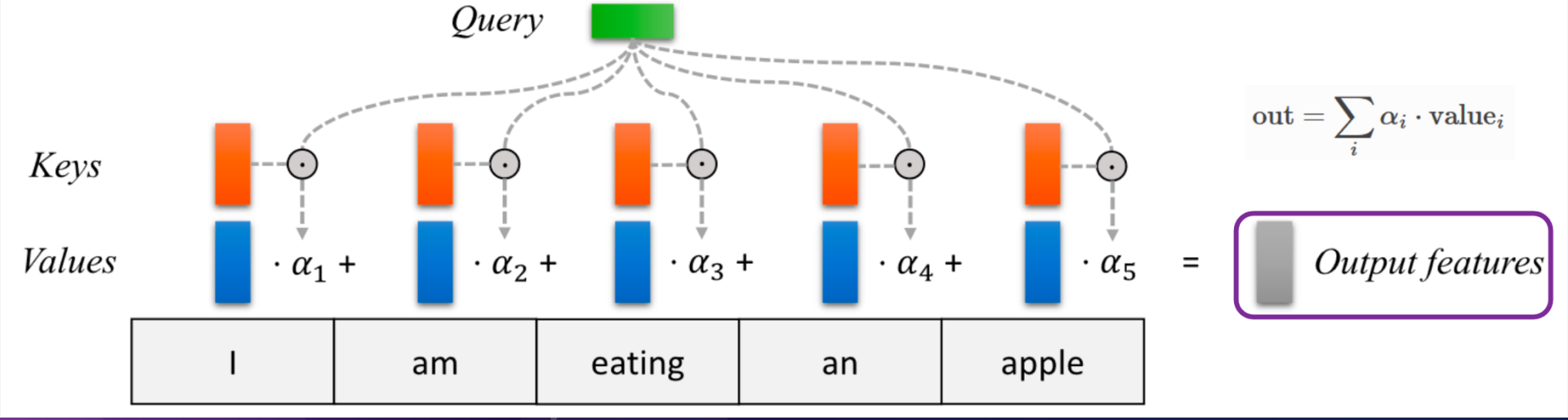


Attention Mechanism



$$\alpha_i = \frac{\exp\left(f_{attn}\left(\text{key}_i, \text{query}\right)\right)}{\sum_j \exp\left(f_{attn}\left(\text{key}_j, \text{query}\right)\right)}$$

Attention Mechanism



$$\text{out} = \sum_i \alpha_i \cdot \text{value}_i$$

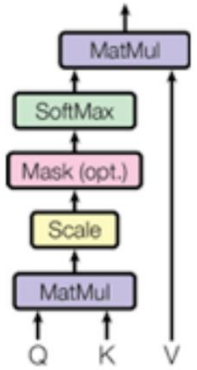
Attention Mechanism

Given a set of queries $Q \in \mathbb{R}^{T \times d_k}$, keys $K \in \mathbb{R}^{T \times d_k}$ and values $V \in \mathbb{R}^{T \times d_v}$ where T is the sequence length, and d_k and d_v are the hidden dimensionality for queries/keys and values respectively.

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

```python
def scaled_dot_product(q, k, v, mask=None):
    d_k = q.size()[-1]
    attn_logits = torch.matmul(q, k.transpose(-2, -1))
    attn_logits = attn_logits / math.sqrt(d_k)
    if mask is not None:
        attn_logits = attn_logits.masked_fill(mask == 0, -9e15)
    attention = F.softmax(attn_logits, dim=-1)
    values = torch.matmul(attention, v)
    return values, attention
```

Scaled Dot-Product Attention



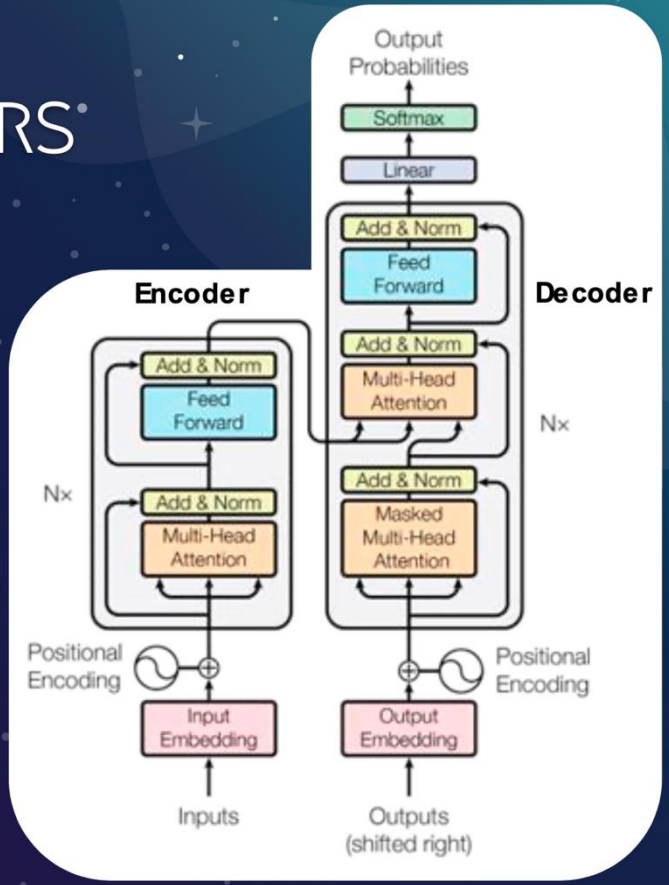
Vaswani et al., 2017

Transformers

Originally for machine translation.

An encoder-decoder structure

Modern Transformers only focus on encoder part (ViT, Bert, ...)



Vaswani et al., 2017

GCN vs GAT

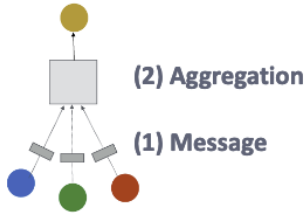
Graph Convolutional Networks (GCN)

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

Message

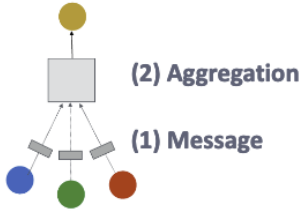
$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

Aggregation



GCN vs GAT

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$



Each Neighbor: $\mathbf{m}_u^{(l)} = \frac{1}{|N(v)|} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}$

Normalized by node degree
(In the GCN paper they use a slightly different normalization)

GCN vs GAT

Graph Attention Networks

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

Attention weights

In GCN

$\alpha_{vu} = \frac{1}{|N(v)|}$ is the **weighting factor (importance)** of node u's message to node v

Graph Attention Mechanism

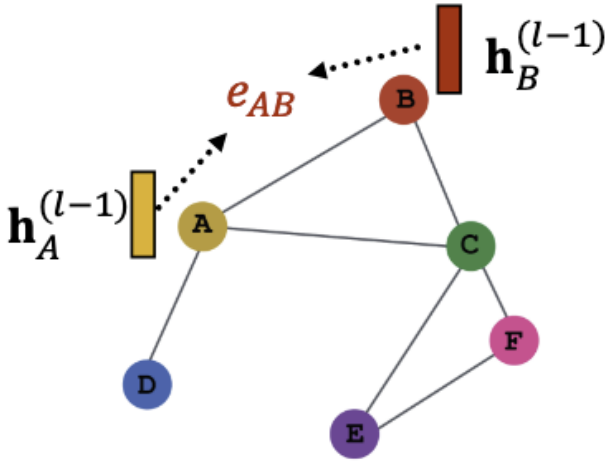
$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

Attention weights

Let α_{vu} be computed as a byproduct of an **attention mechanism a:**

$$e_{vu} = a(\mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}, \mathbf{W}^{(l)} \boldsymbol{h}_v^{(l-1)})$$

e_{vu} indicates the importance of u's message to node v



$$e_{AB} = a(\mathbf{W}^{(l)} \mathbf{h}_A^{(l-1)}, \mathbf{W}^{(l)} \mathbf{h}_B^{(l-1)})$$

Graph Attention Mechanism

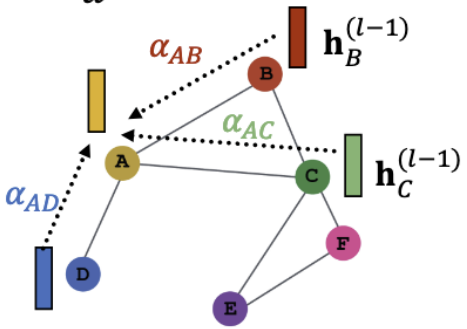
Use the **softmax** function, so that $\sum_{u \in N(v)} \alpha_{vu} = 1$:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

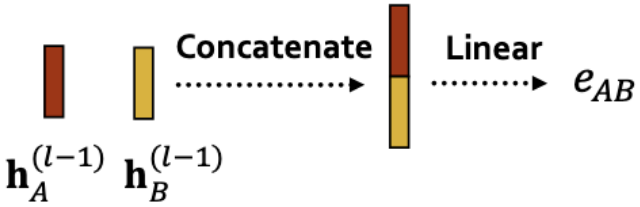
$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

Weighted sum using α_{AB}, α_{AC}, α_{AD}:

$$\mathbf{h}_A^{(l)} = \sigma(\alpha_{AB}\mathbf{W}^{(l)}\mathbf{h}_B^{(l-1)} + \alpha_{AC}\mathbf{W}^{(l)}\mathbf{h}_C^{(l-1)} + \alpha_{AD}\mathbf{W}^{(l)}\mathbf{h}_D^{(l-1)})$$



Graph Attention Mechanism



$$e_{AB} = a\left(\mathbf{W}^{(l)}\mathbf{h}_A^{(l-1)}, \mathbf{W}^{(l)}\mathbf{h}_B^{(l-1)}\right)$$
$$= \text{Linear}\left(\text{Concat}\left(\mathbf{W}^{(l)}\mathbf{h}_A^{(l-1)}, \mathbf{W}^{(l)}\mathbf{h}_B^{(l-1)}\right)\right)$$

Parameters of a are trained jointly

Multi-head Attention

$$\mathbf{h}_v^{(l)}[1] = \sigma\left(\sum_{u \in N(v)} \alpha_{vu}^{1} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

$$\mathbf{h}_v^{(l)}[2] = \sigma\left(\sum_{u \in N(v)} \alpha_{vu}^{2} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

$$\mathbf{h}_v^{(l)}[3] = \sigma\left(\sum_{u \in N(v)} \alpha_{vu}^{3} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}\left(\mathbf{h}_v^{(l)}[1], \mathbf{h}_v^{(l)}[2], \mathbf{h}_v^{(l)}[3]\right)$$


Practical: GCN
https://t.ly/GhdIs


Practical: GAT
https://t.ly/C90q0

End of lecture!

ANY QUESTIONS?

You can find me at raoni.lourenco@uni.lu