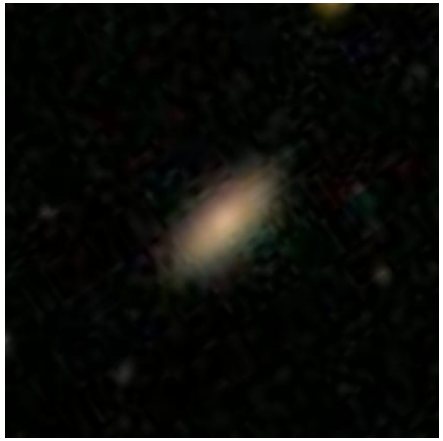
Symmetries and specialized architectures

Vera Maiboroda, vera.maiboroda@cern.ch

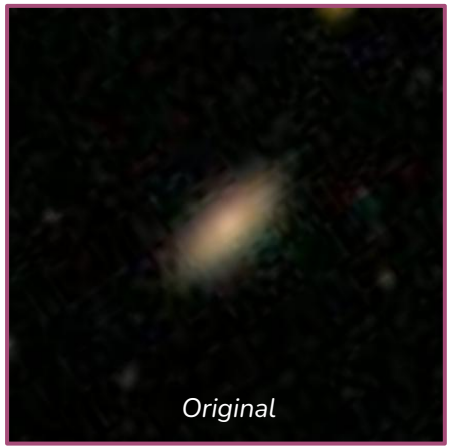
Let’s get back to the galaxies classification task

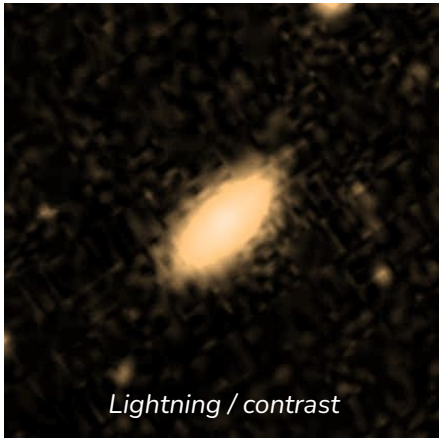


Elliptical

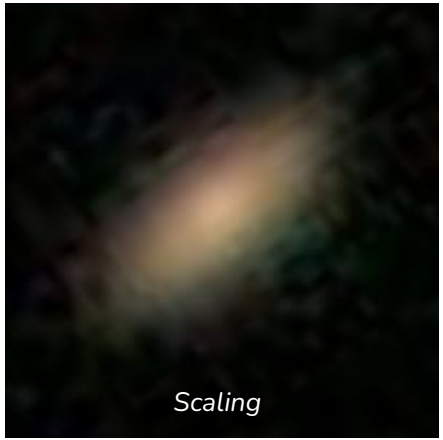
Spiral

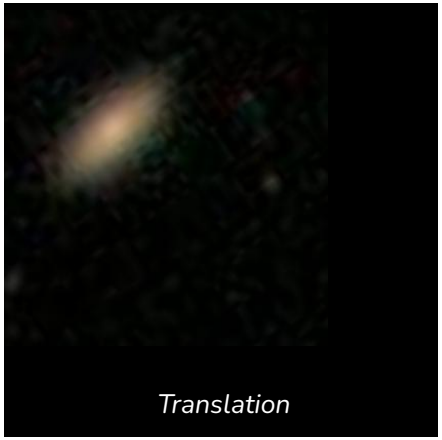
Why image classification is a difficult task?

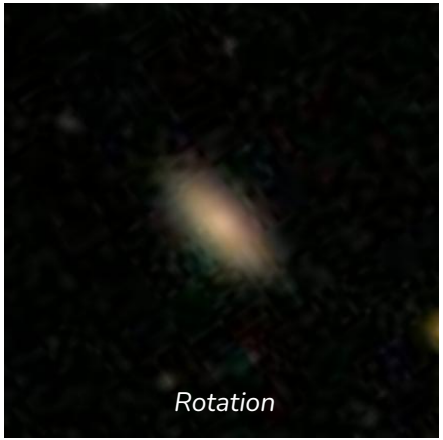




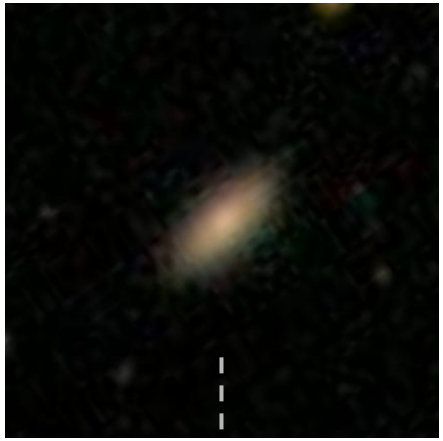
The same object might appear differently on image.







Traditional approaches for image processing: Hand-engineered features

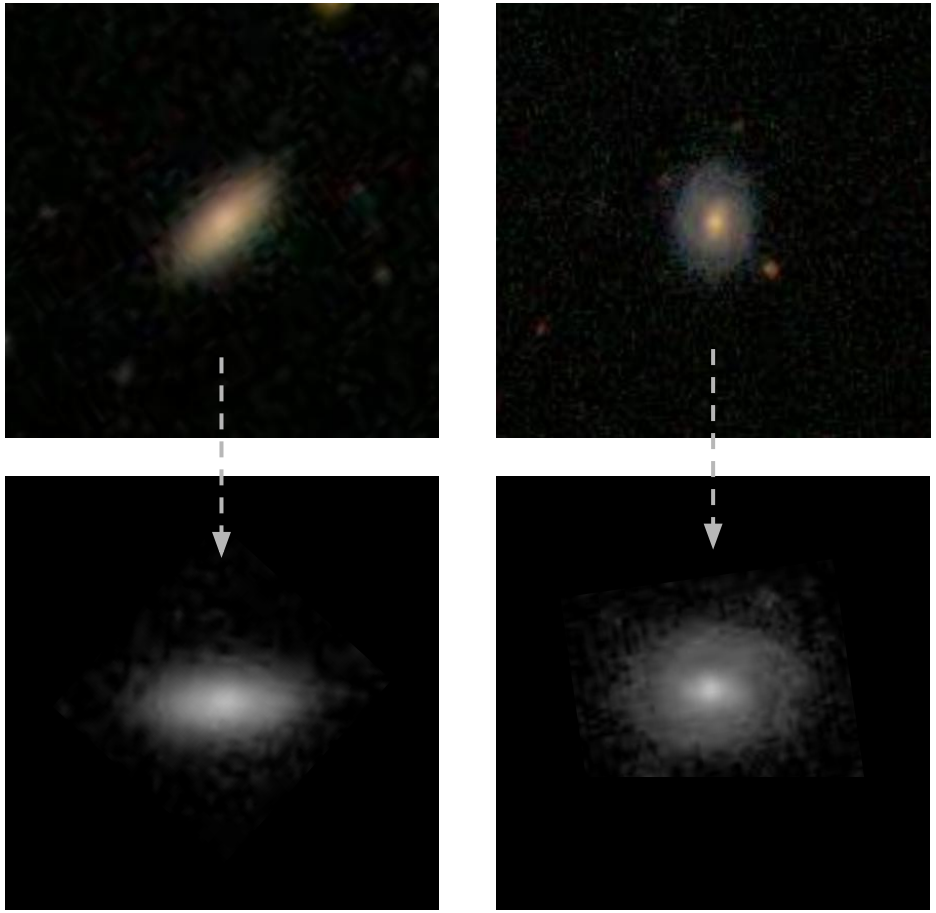


Let's describe the object with (extracted from image) hand-engineered features:

- Requires human domain-specific expertise
- Might lead to loss of information
- ...

Complicated :(

Traditional approaches for image processing: Preprocessing

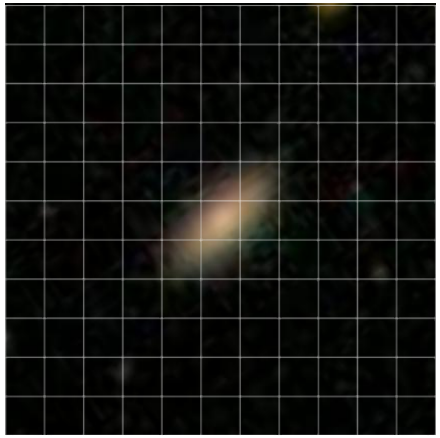


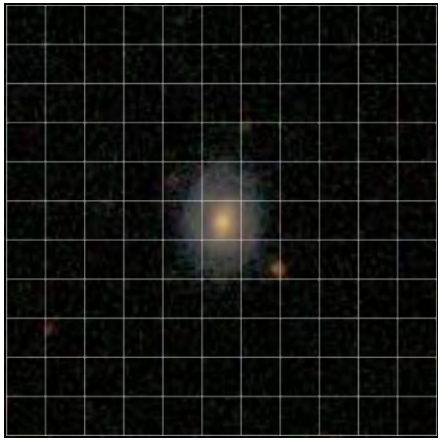
Let’s make an object appear in a same way:

- Colour normalisation
- Spatial transformations: rotation, scaling, etc
- Background removal
- ...

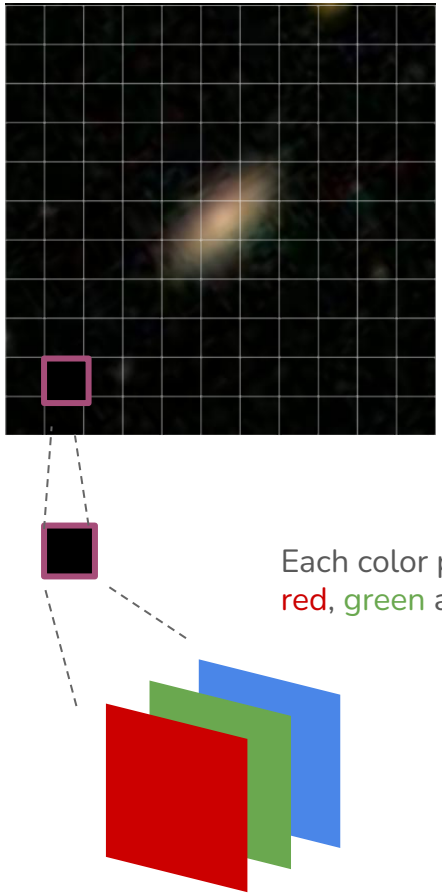
Complicated :(

How an image is represented?

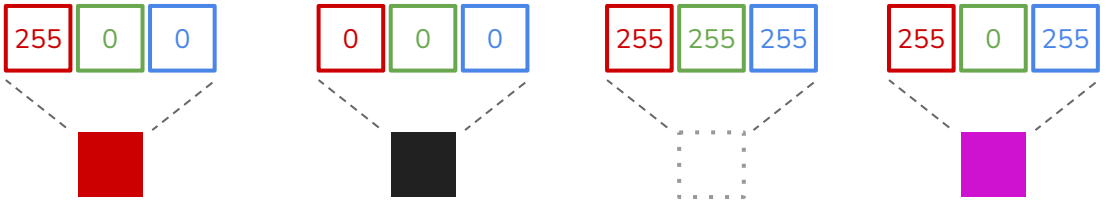




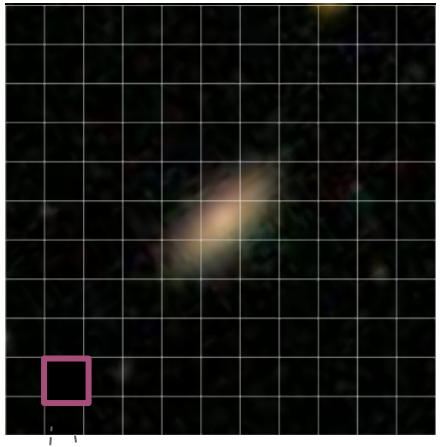
How an image is represented?

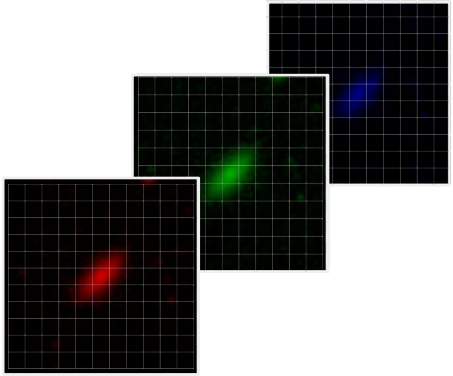


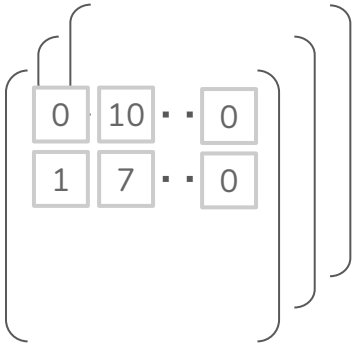
Each color pixel is represented via 3 channels:
red, green and blue with intensities [0, 255].



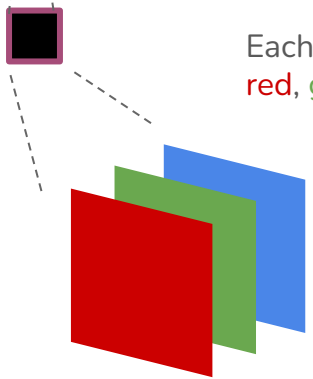
How an image is represented?



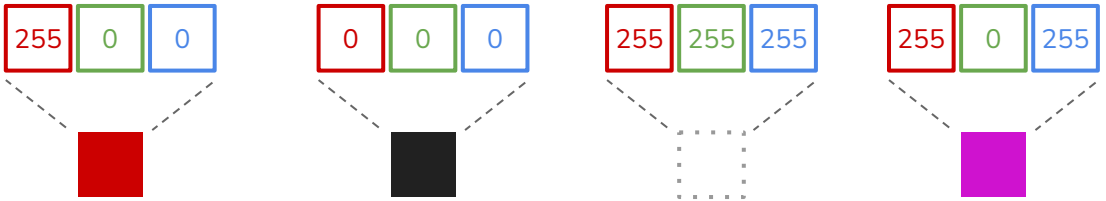




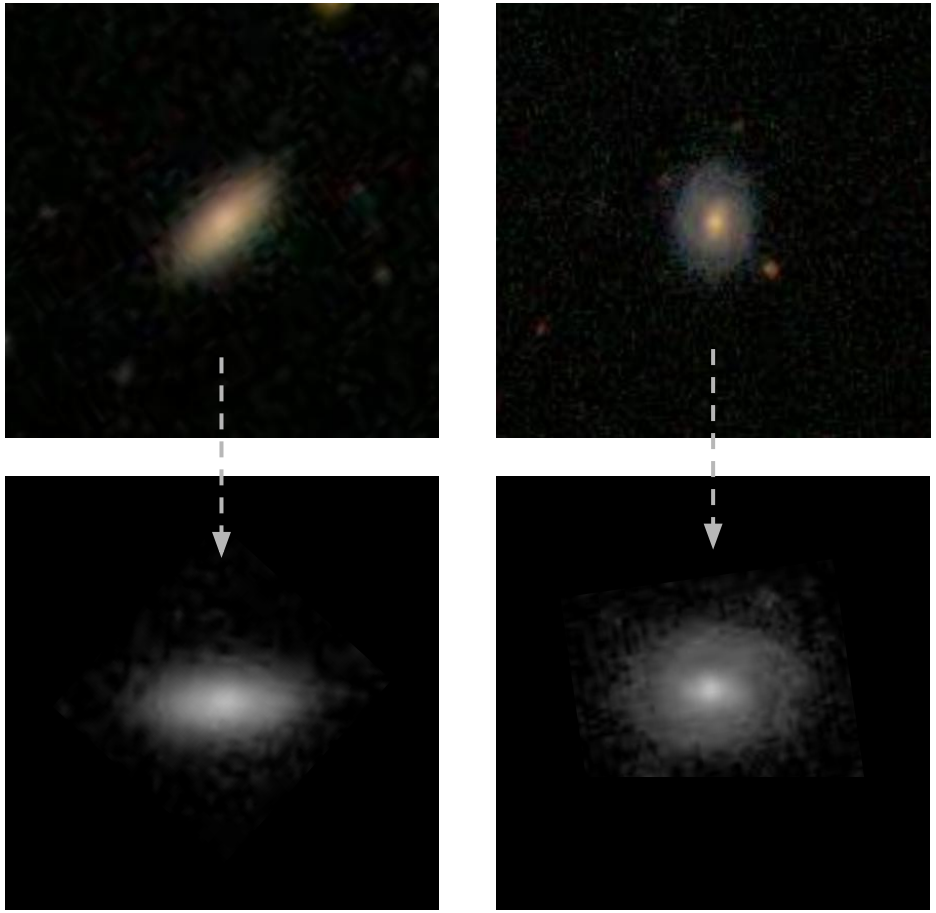
Galaxy image = tensor of size [3, 424, 424]



Each color pixel is represented via 3 channels:
red, green and blue with intensities [0, 255].



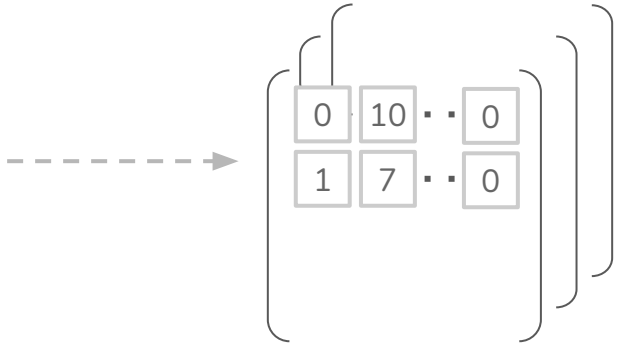
Even with preprocessing, an image contains too many features



Let's make an object appear in a same way:

- Colour normalisation
- Spatial transformations: rotation, scaling, etc
- Background removal
- ...

Complicated :(



tensor of size [3, 424, 424] → 540k features 🥲

Examples of structured data

- 1D:
  - Time series
  - Sequences
- 2D:
  - Black-white images
- 3D:
  - Color images
  - Energy deposition in calorimeter cells
- 4D:
  - Video (time series of images)
- ...

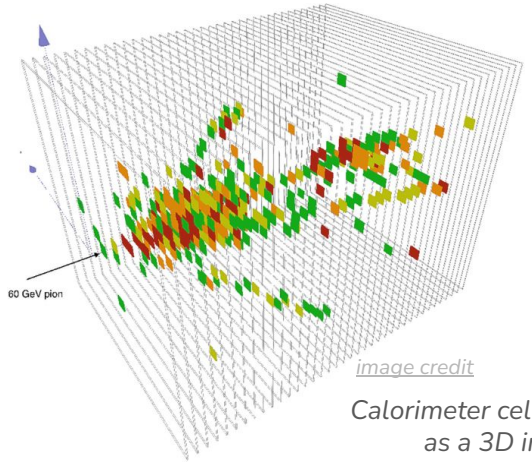
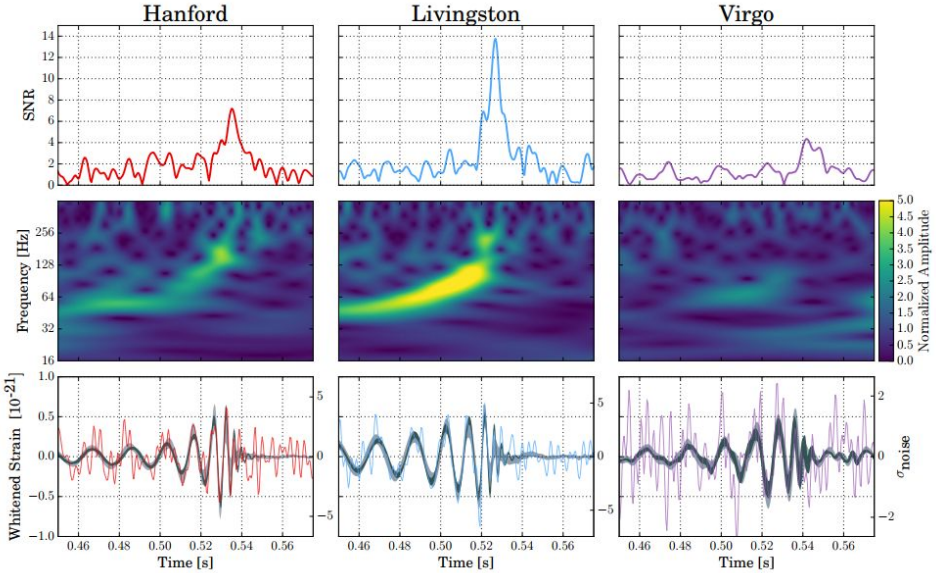


image credit

Calorimeter cells response as a 3D image

image credit



Gravitational waves data

Examples of structured data

- 1D:
  - Time series
  - Sequences
- 2D:
  - Black-white images
- 3D:
  - Color images
  - Energy deposition in calorimeter cells
- 4D:
  - Video (time series of images)
- ...

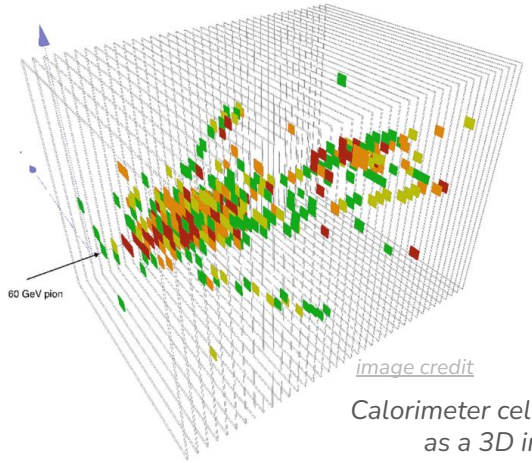
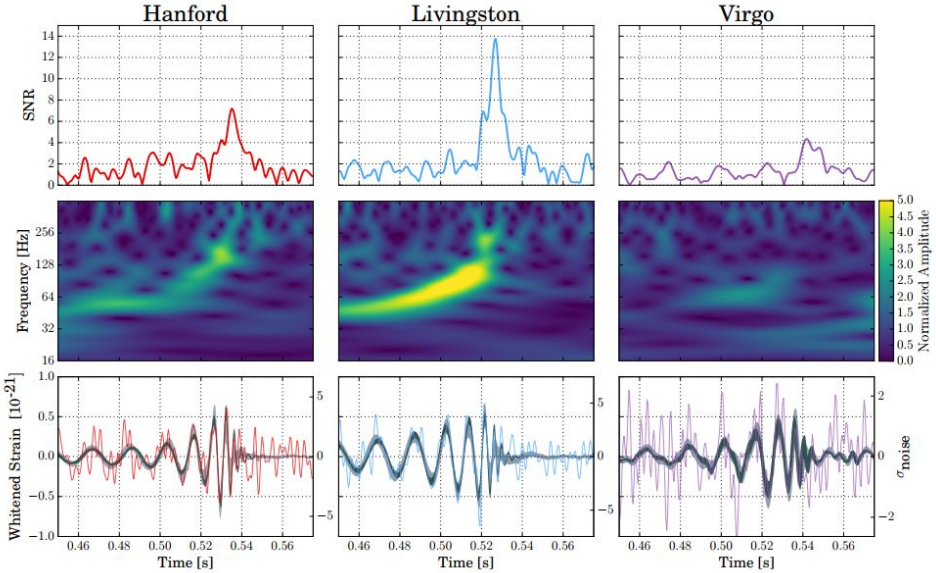


image credit

Calorimeter cells response as a 3D image

image credit



Gravitational waves data

- Order matters
- ↑ dimensionality → ↑ model parameters
  → ↑ training data needed

Another type of data: graphs

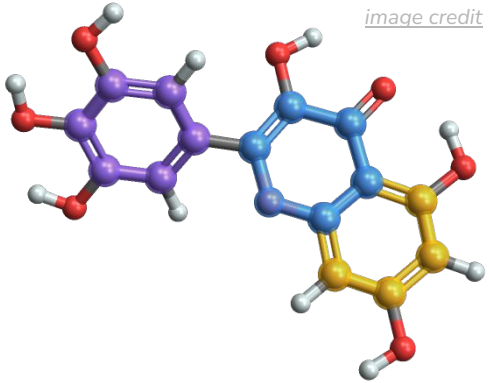
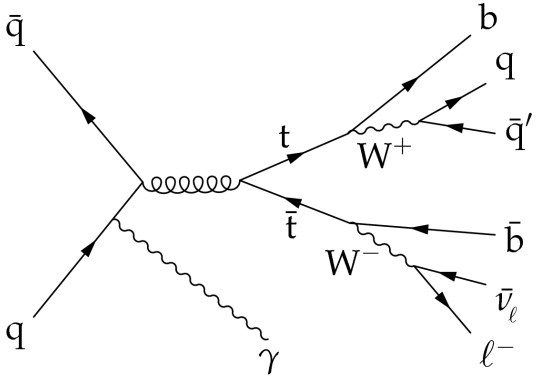
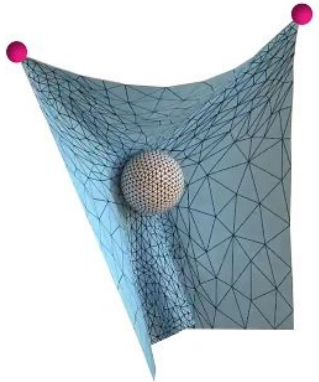


image credit

Molecular graph of myricetin.



Feynman diagram for the $t\bar{t}\gamma$ process in the semileptonic final state.



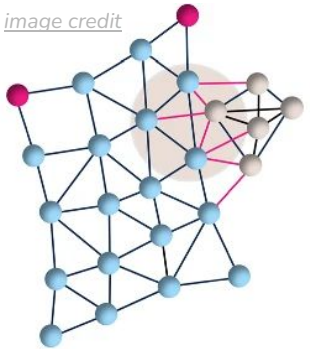


image credit

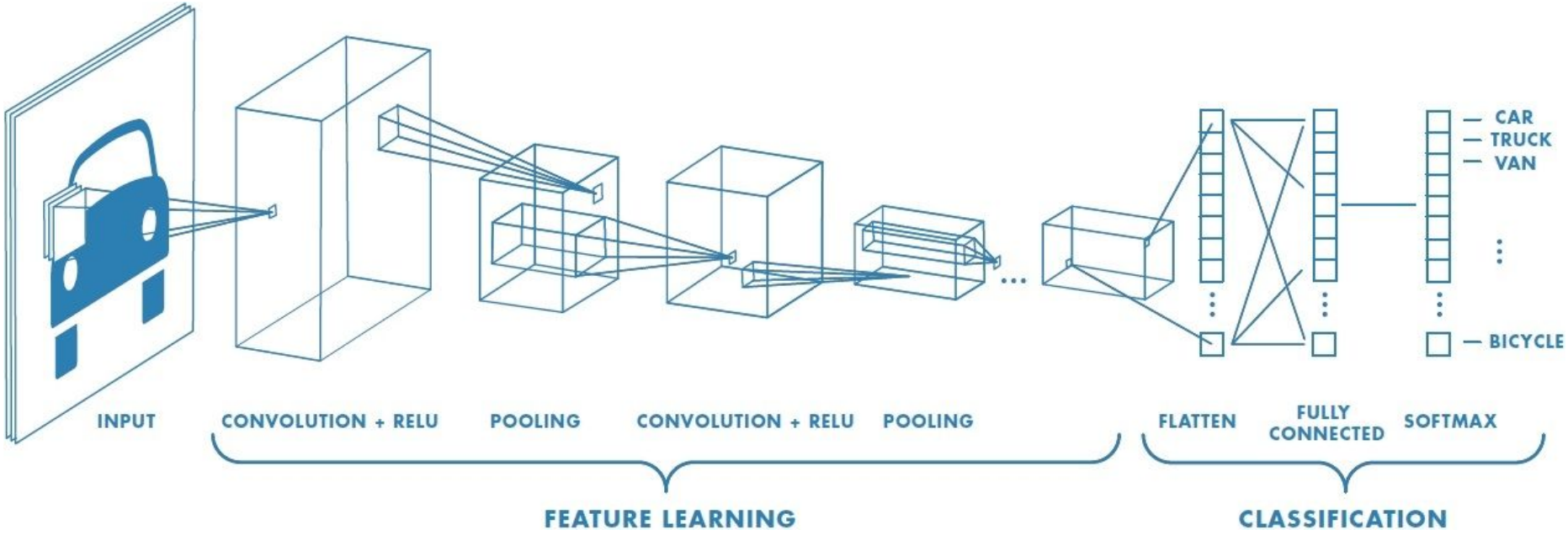
Graph representation method for modelling the contact between objects.

Class goals

- Special cases:
  - Convolutional neural networks (CNNs)
  - Graph neural networks (GNNs)
- Universal case:
  - Transformers → imposing desired symmetries?

CNNs

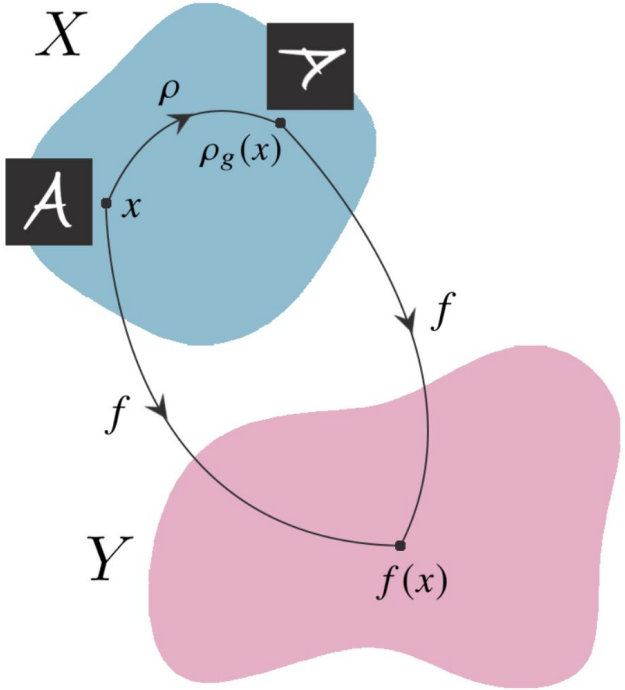
Automatic feature extraction from image



Invariance vs equivariance

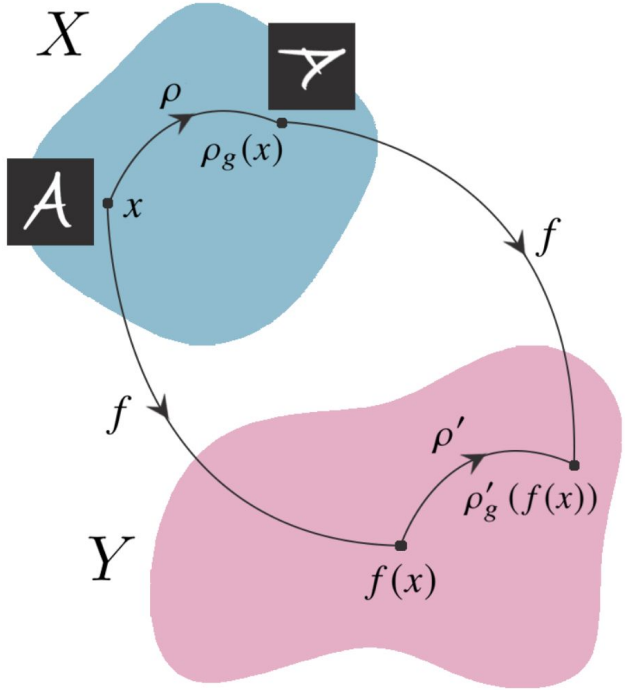
Invariance

$$f(\rho_g(x)) = f(x)$$



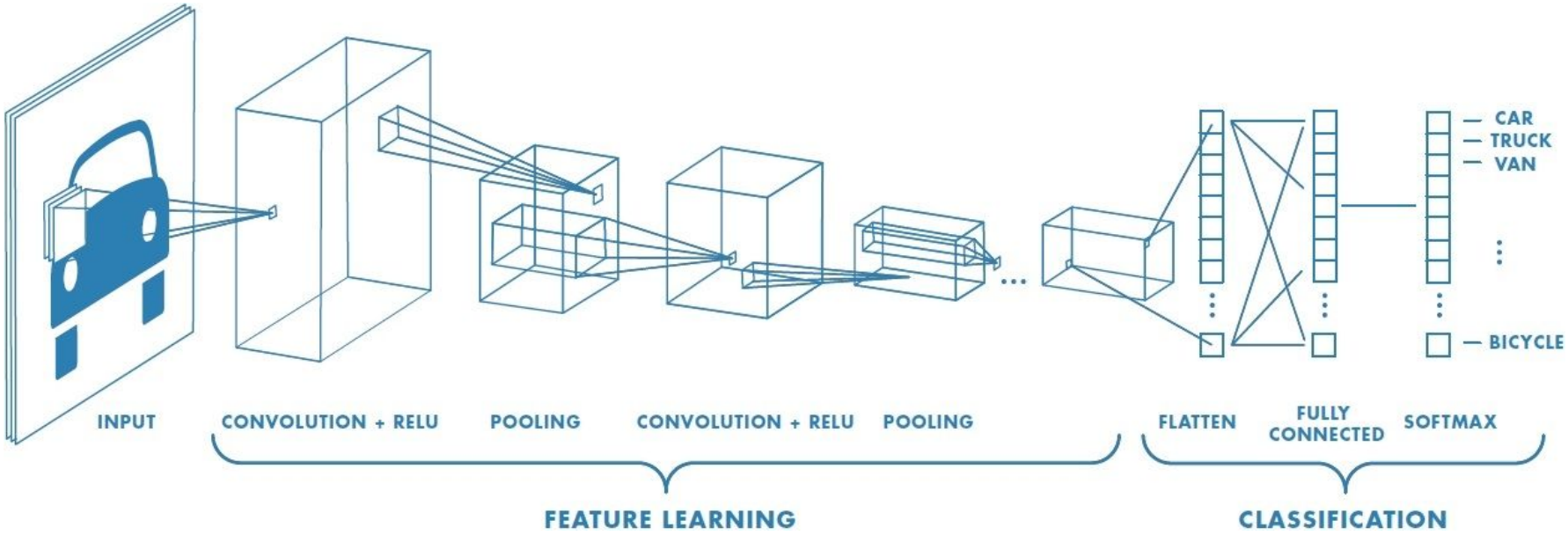
Equivariance

$$f(\rho_g(x)) = \rho'_g\,(f(x))$$

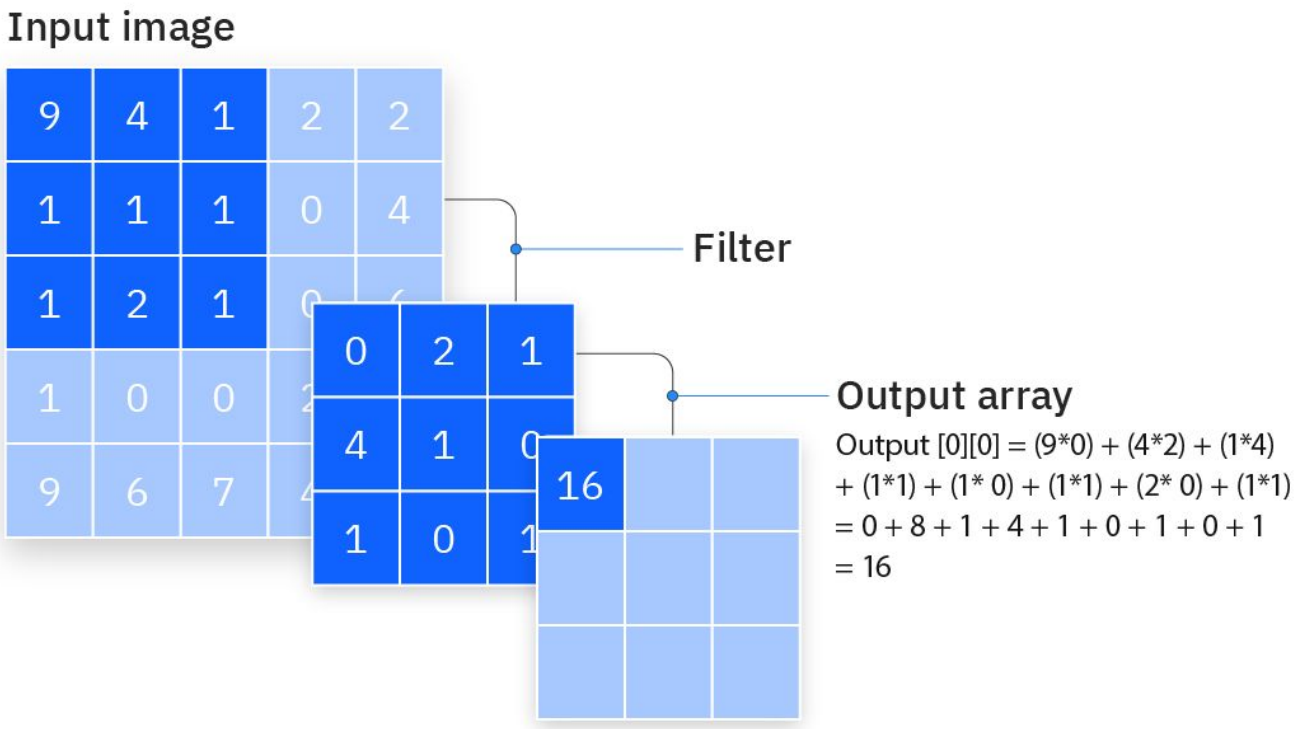


Hierarchical representation

- **Convolution**
  locally extracts features for translational equivariance
- **Pooling**
  provides approximate invariance to translations
- Sequence of convolutions and poolings
  models hierarchy of local representations

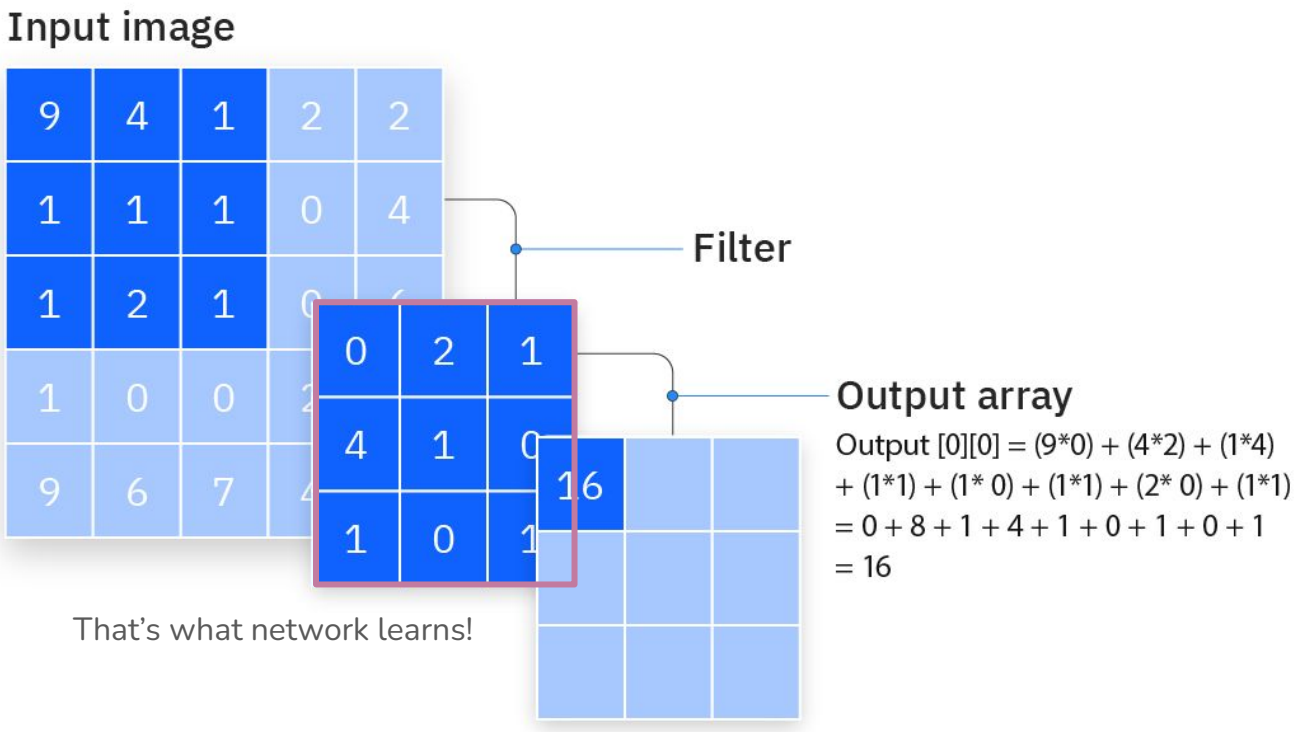


What is a convolution layer



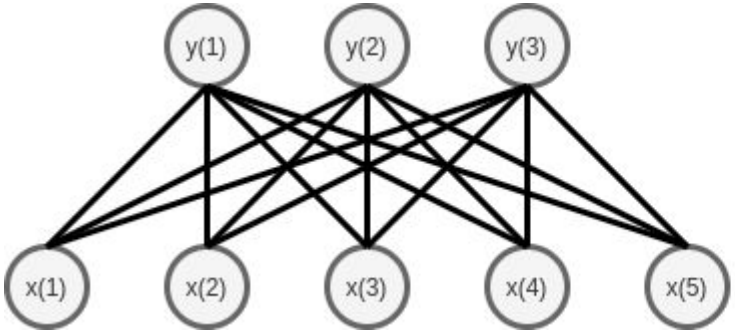
Visualisations here and here

What is a convolution layer

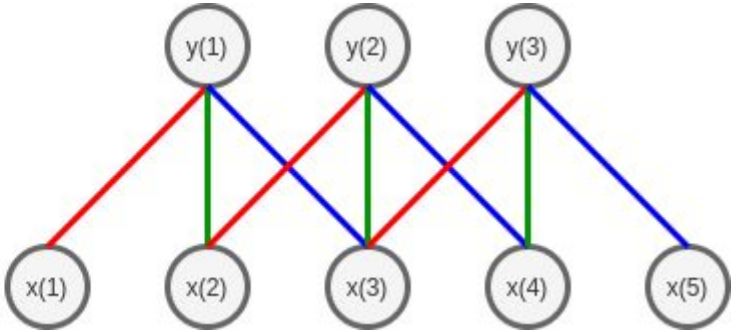


Visualisations here and here

Weight sharing

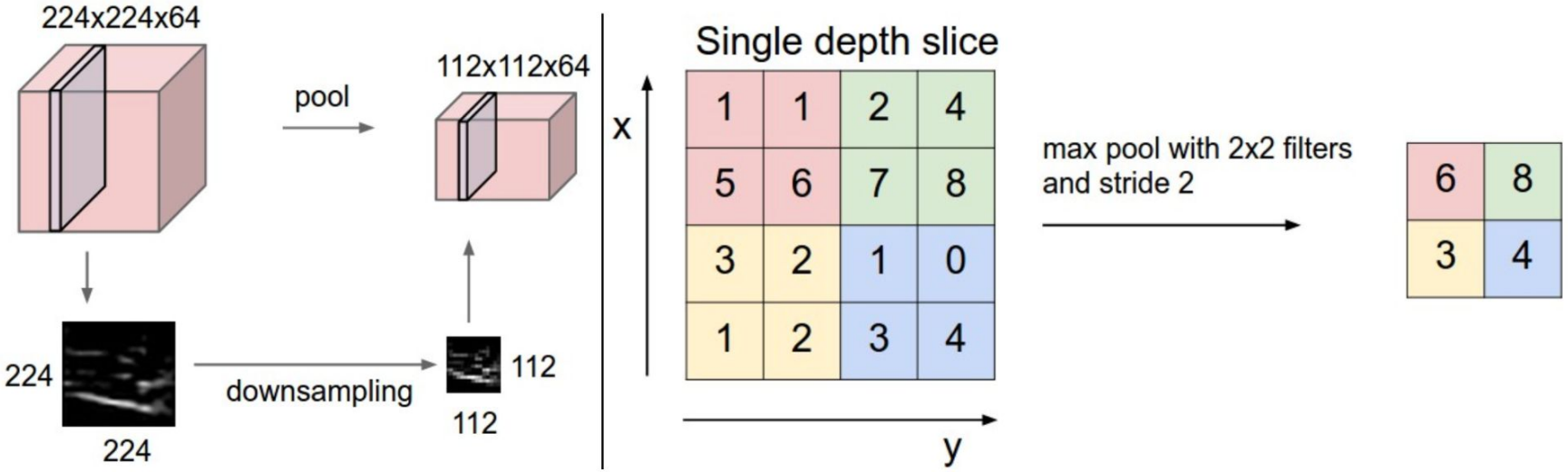


MLP



CNN: same weights - same color, much less to learn

What is pooling

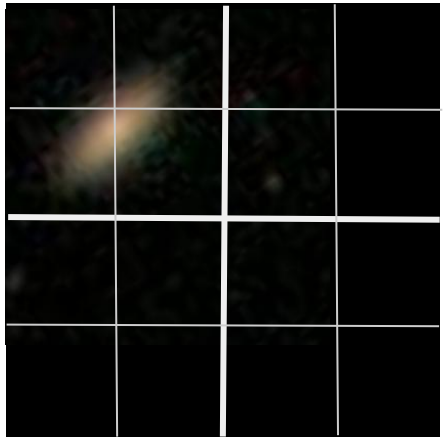


Pooling - downsampling operation, typically applied after a convolution layer

- Max: each pooling operation selects the maximum value of the current view
- Average: each pooling operation averages the values of the current view

Visualisations here and here

How does translational “invariance” work

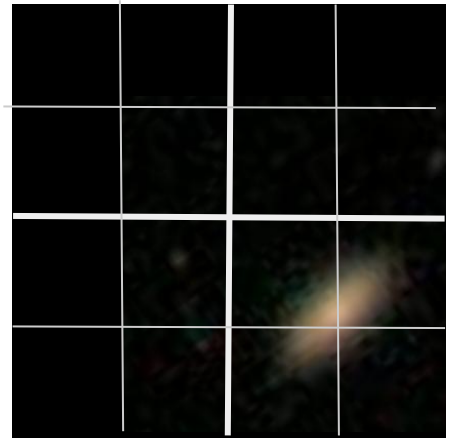


X

conv

=

3	1
0	0



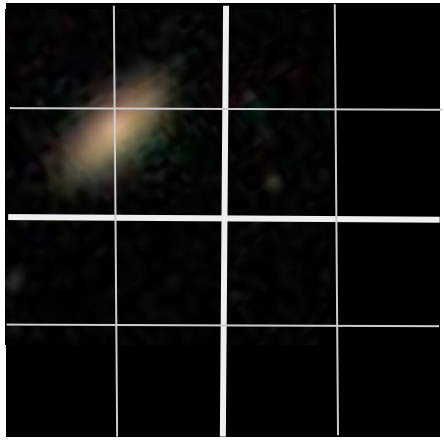
X

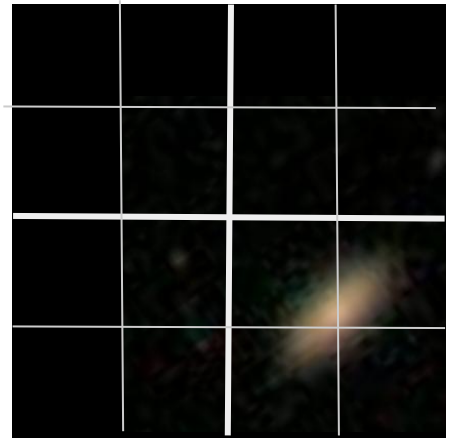
conv

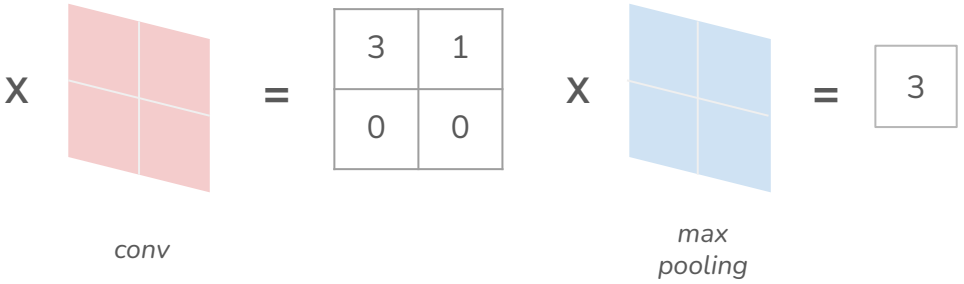
=

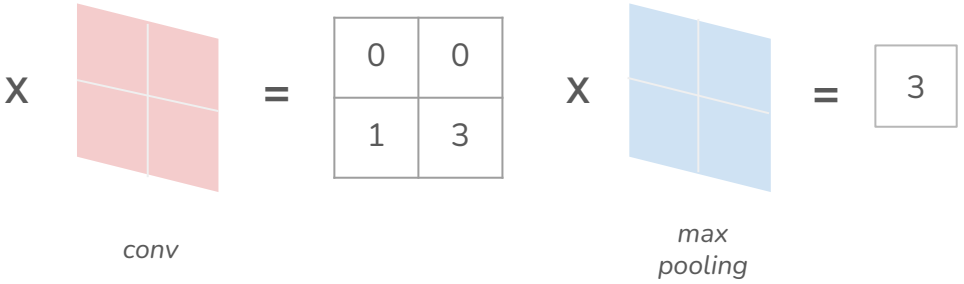
0	0
1	3

How does translational “invariance” work



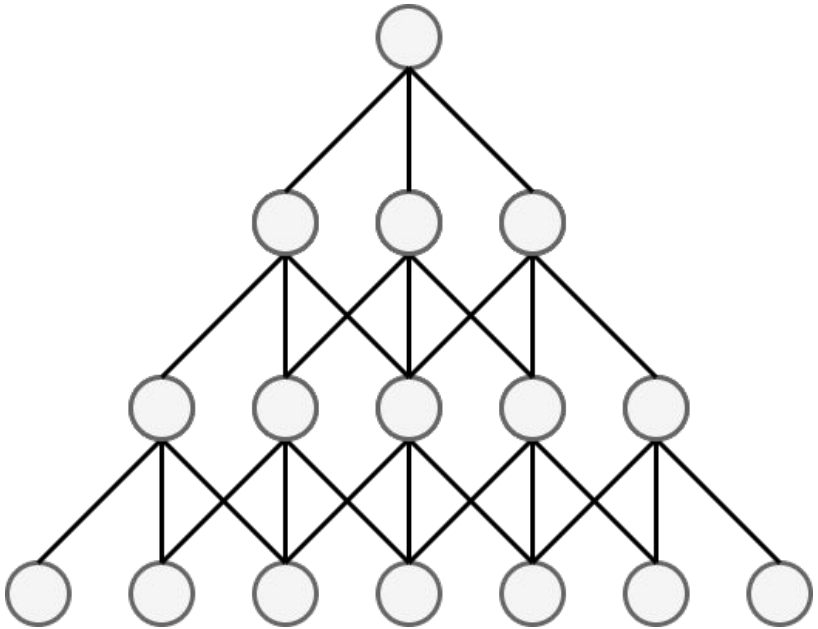




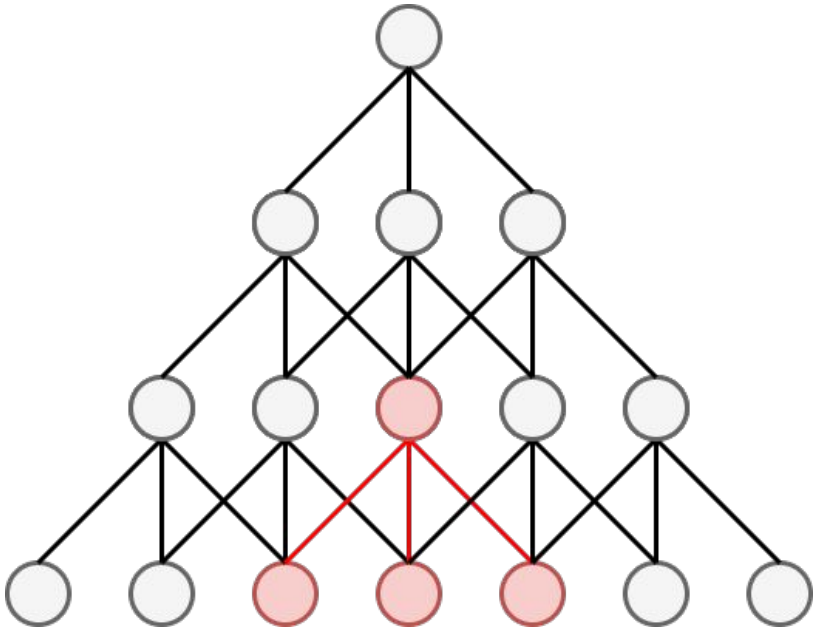


Receptive field

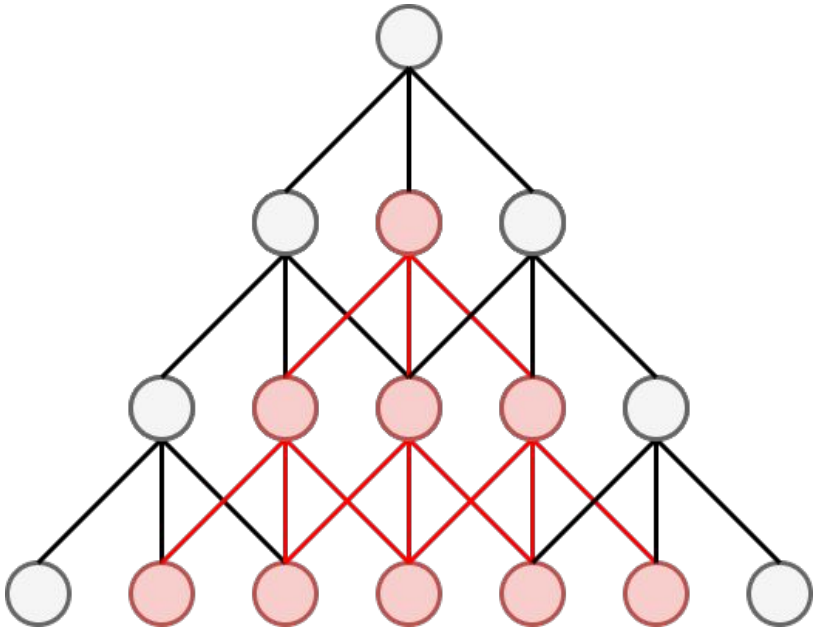
Sequence of convolutional layers allows to detect more complex features by increasing receptive field



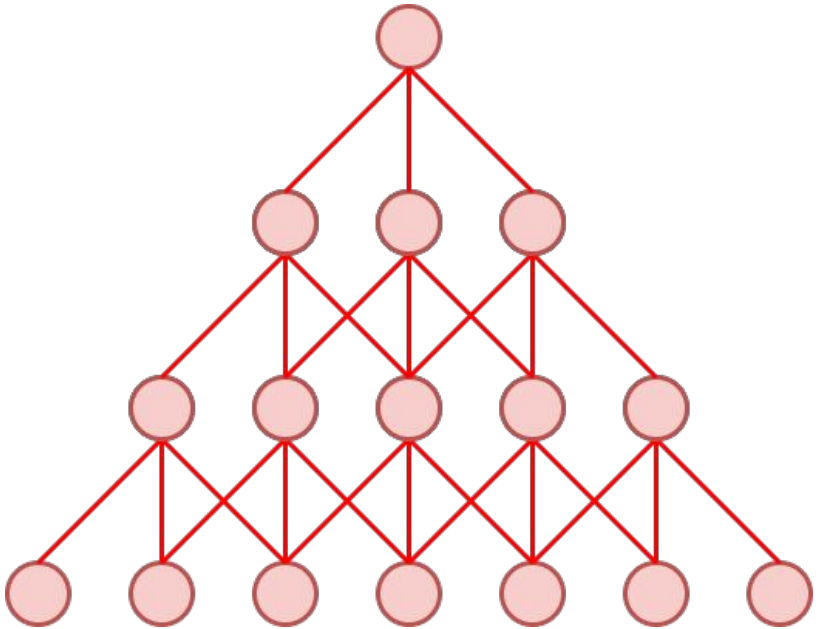
Receptive field



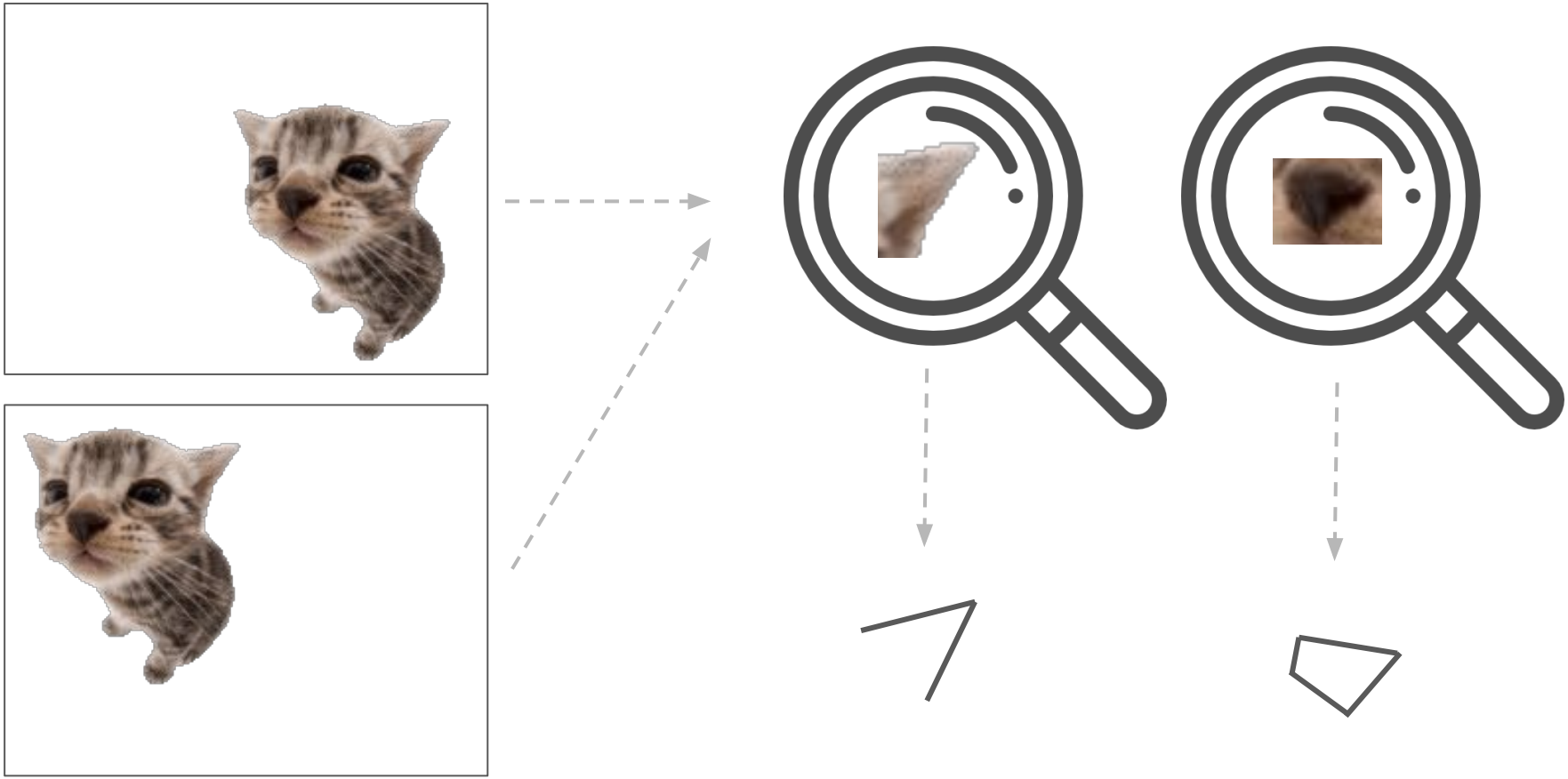
Receptive field



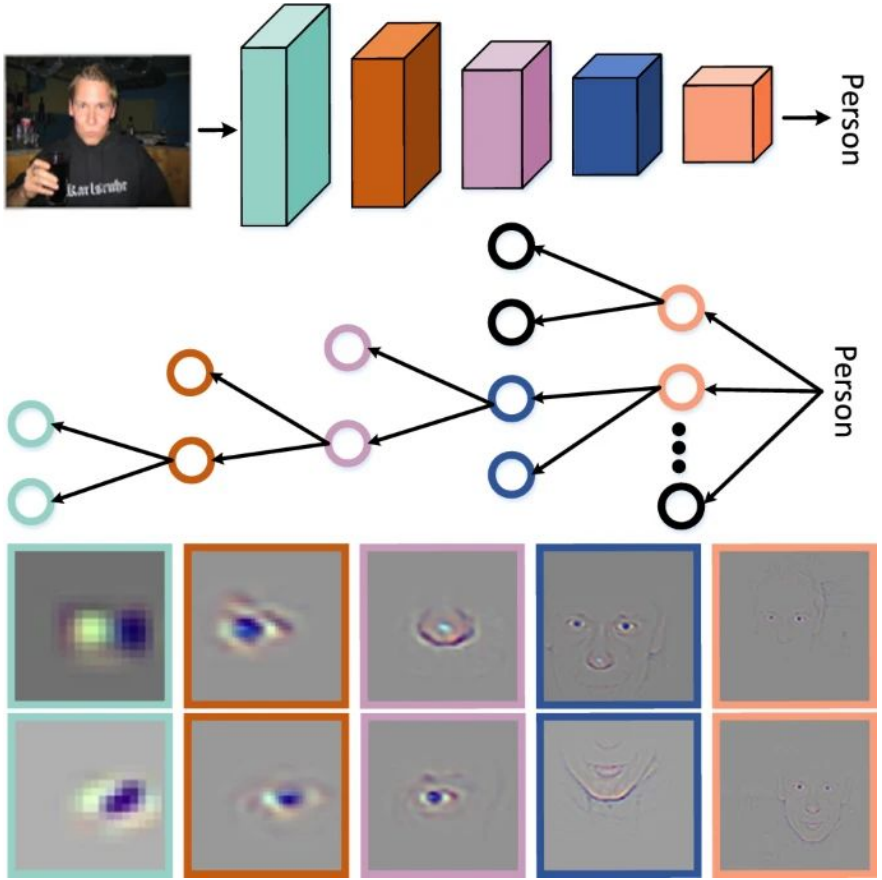
Receptive field



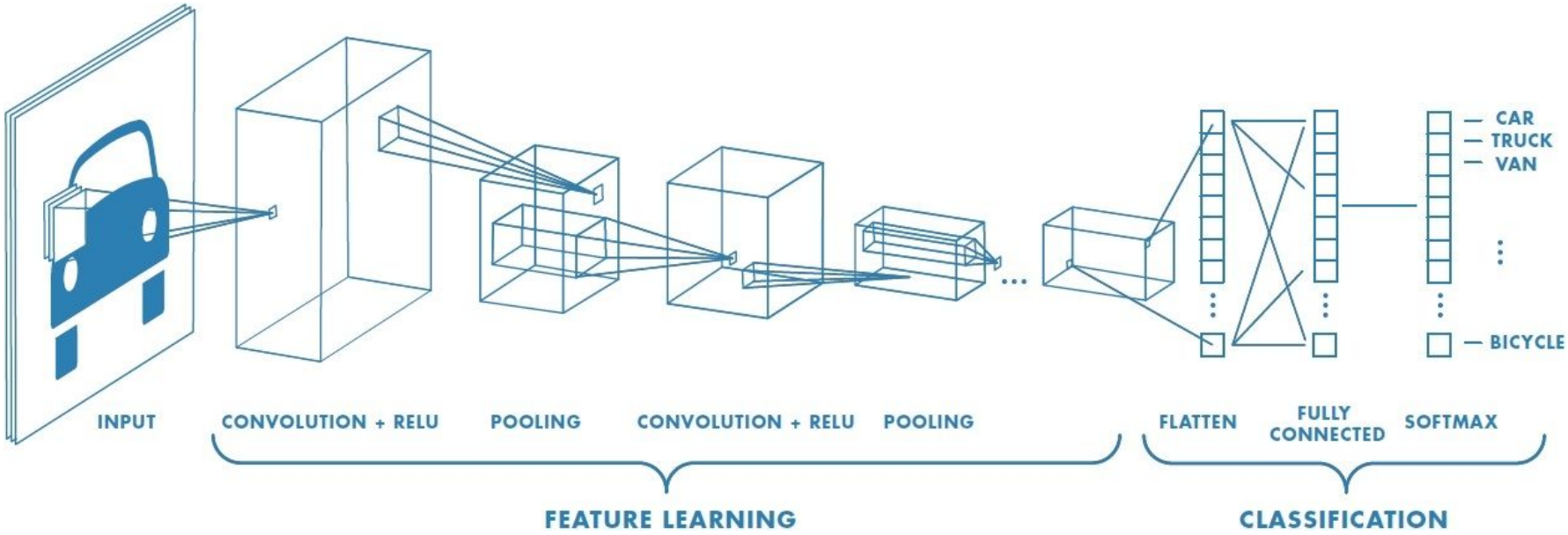
Hierarchical representation



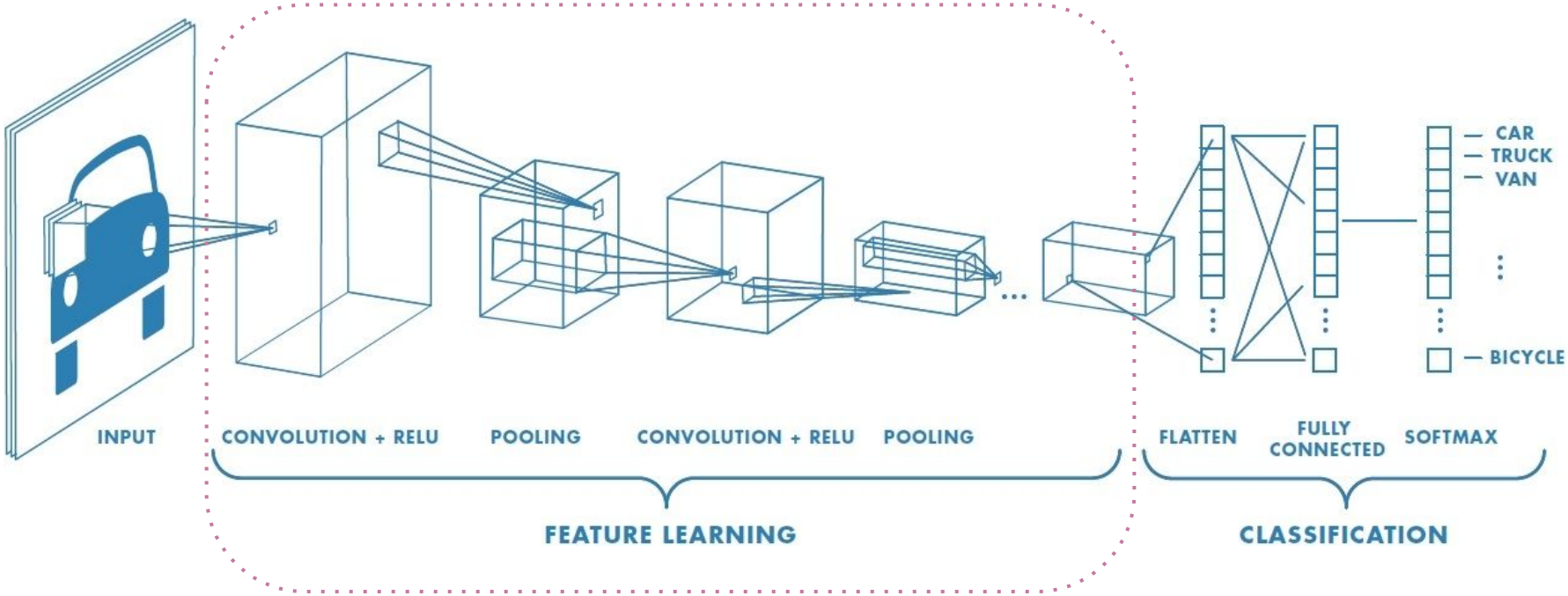
What does a CNN see



We can do transfer learning



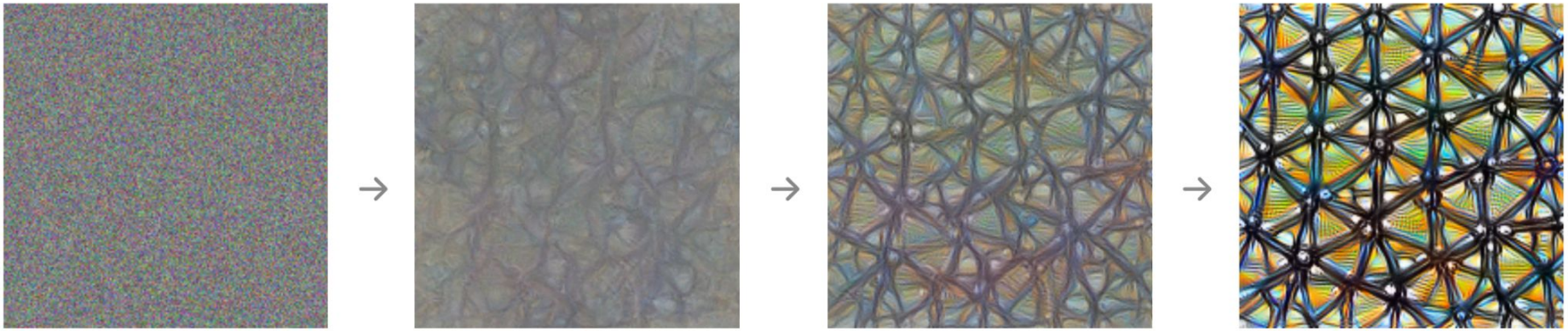
We can do transfer learning



Reuse already trained feature extractor part

We can do transfer learning

image credit



We can do transfer learning

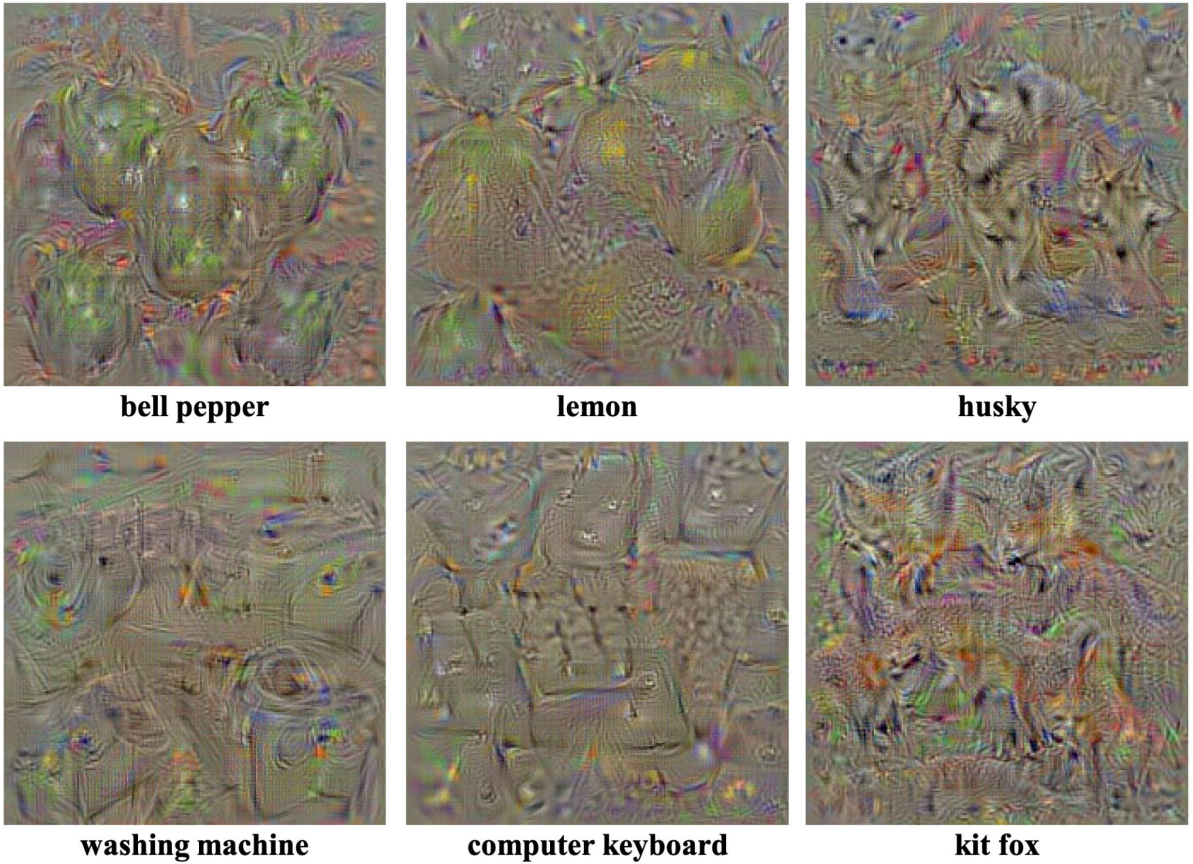
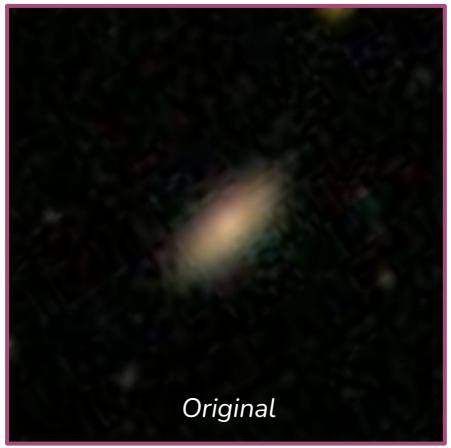
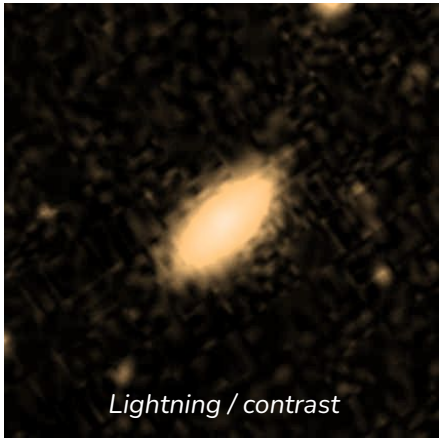
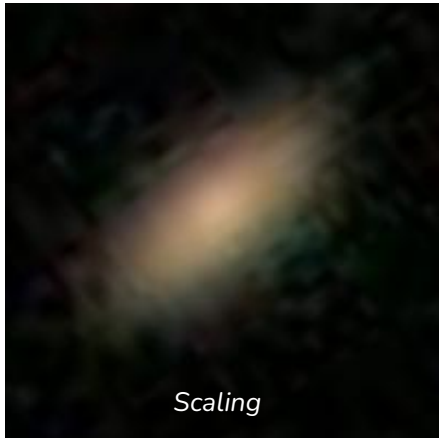


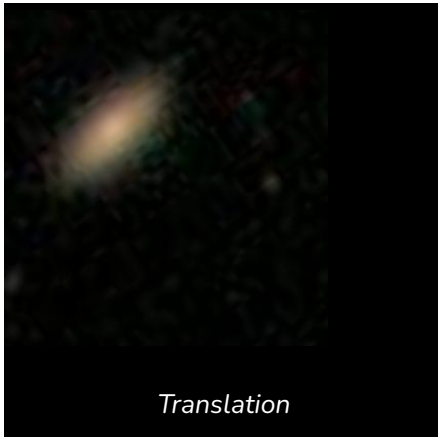
image credit

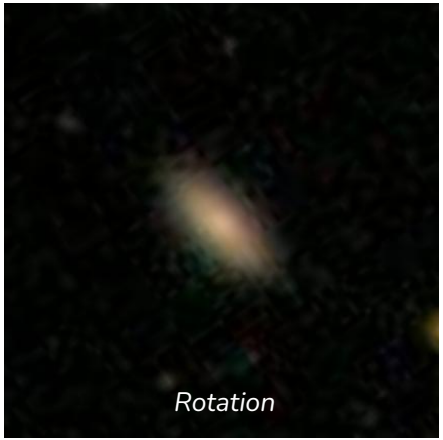
Made it for translation, but what about other invariants?



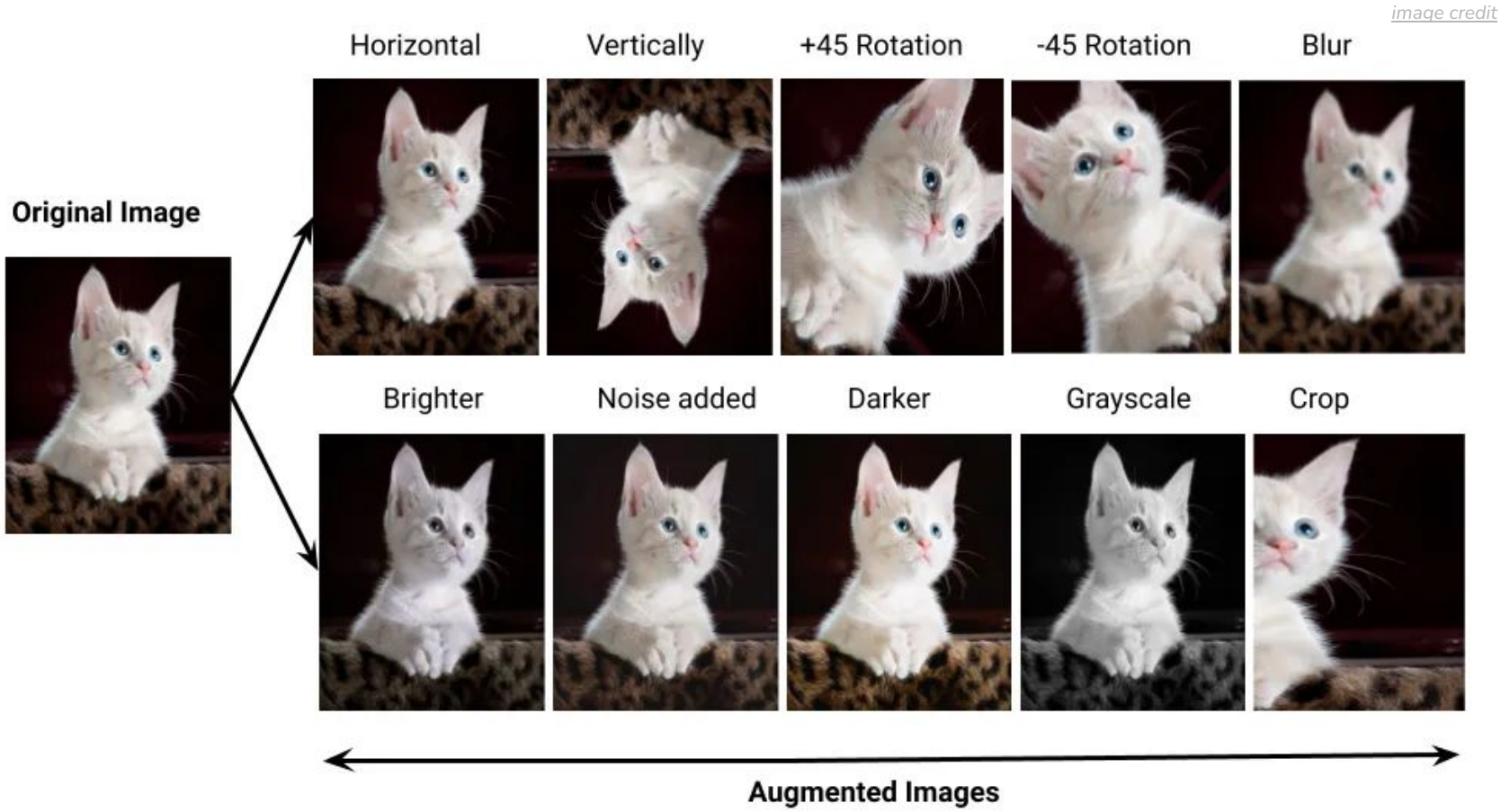








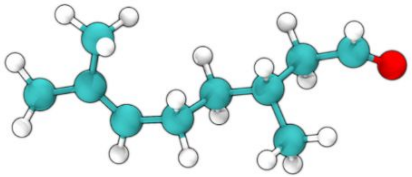
Made it for translation, but what about other invariants?

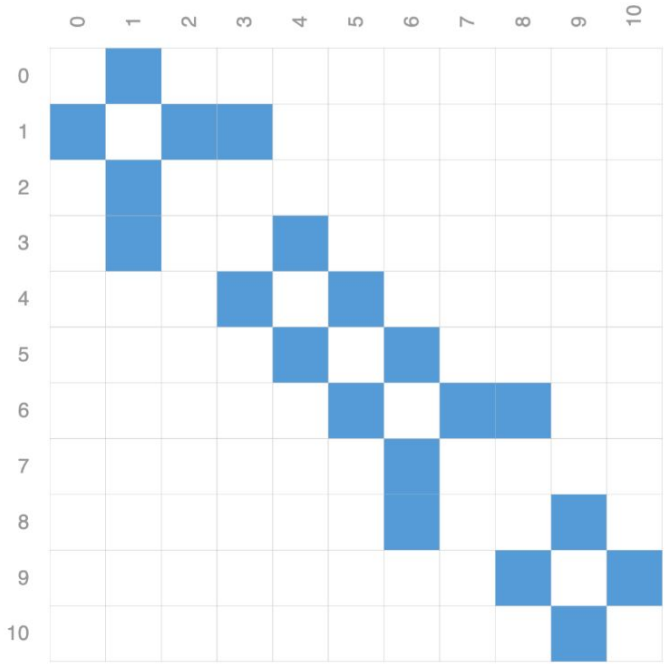


Visualisations here and here

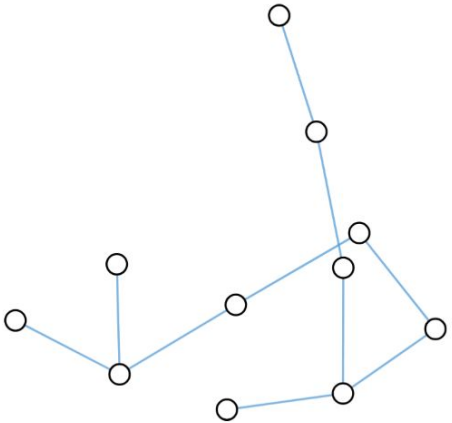
GNNs

How to describe a graph?



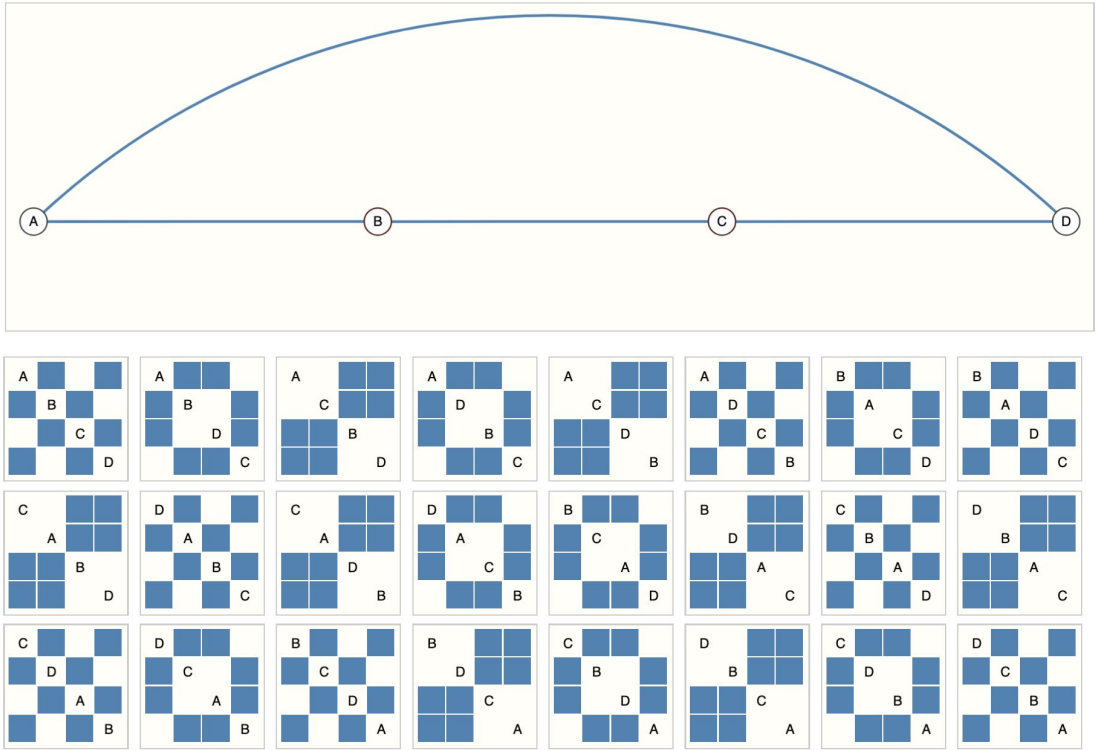


Adjacency matrix



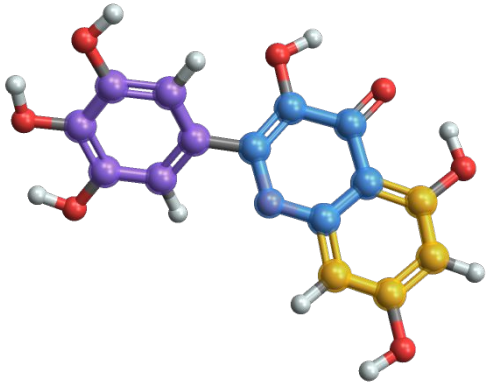
Graphs are **permutation invariant**, no difference on how we number nodes

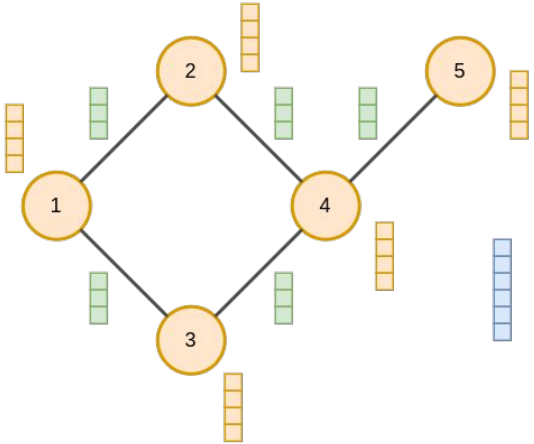
How to describe a graph?



Graphs are **permutation invariant**, no difference on how we number nodes

How to describe a graph?





$\mathbf{h}_i$ *Nodes description*

$\mathbf{e}_j$ *Edges description*

$\mathbf{g}$ *Graph description*

How to describe a graph?

Nodes

```
[0 , 1 , 1 , 0 , 0 , 1 , 1 , 1 ]
```

Edges

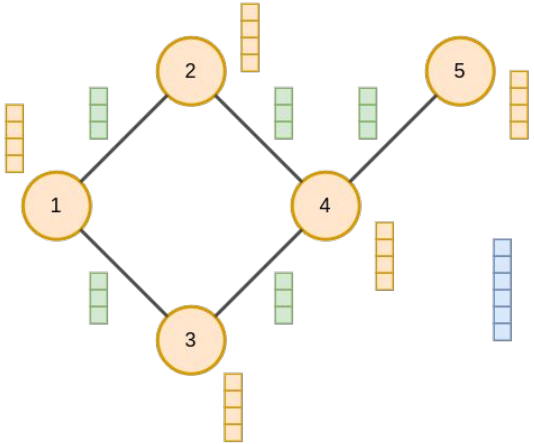
```
[2 , 1 , 1 , 1 , 2 , 1 , 1 ]
```

Adjacency List

```
[[1, 0] , [2, 0] , [4, 3] , [6, 2] ,
 [7, 3] , [7, 4] , [7, 5] ]
```

Global

```
0
```



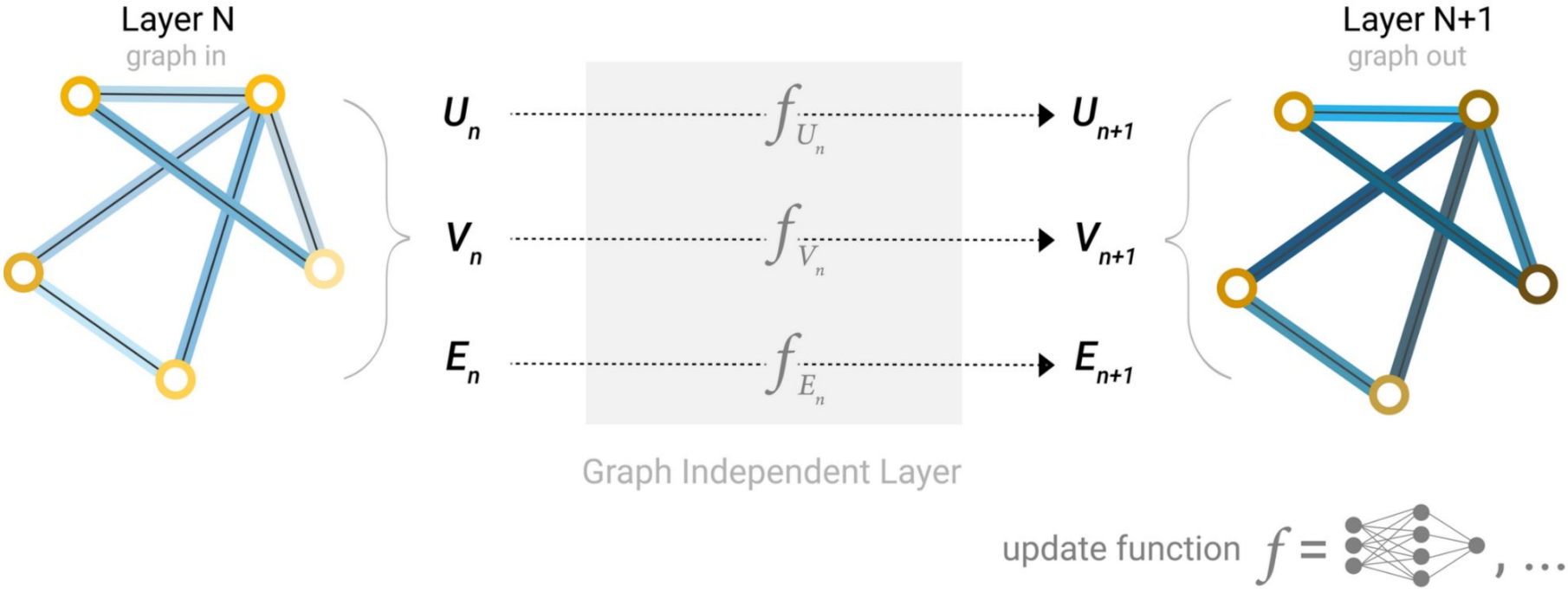
$\mathbf{h}_i$ *Nodes description*

$\mathbf{e}_j$ *Edges description*

$\mathbf{g}$ *Graph description*

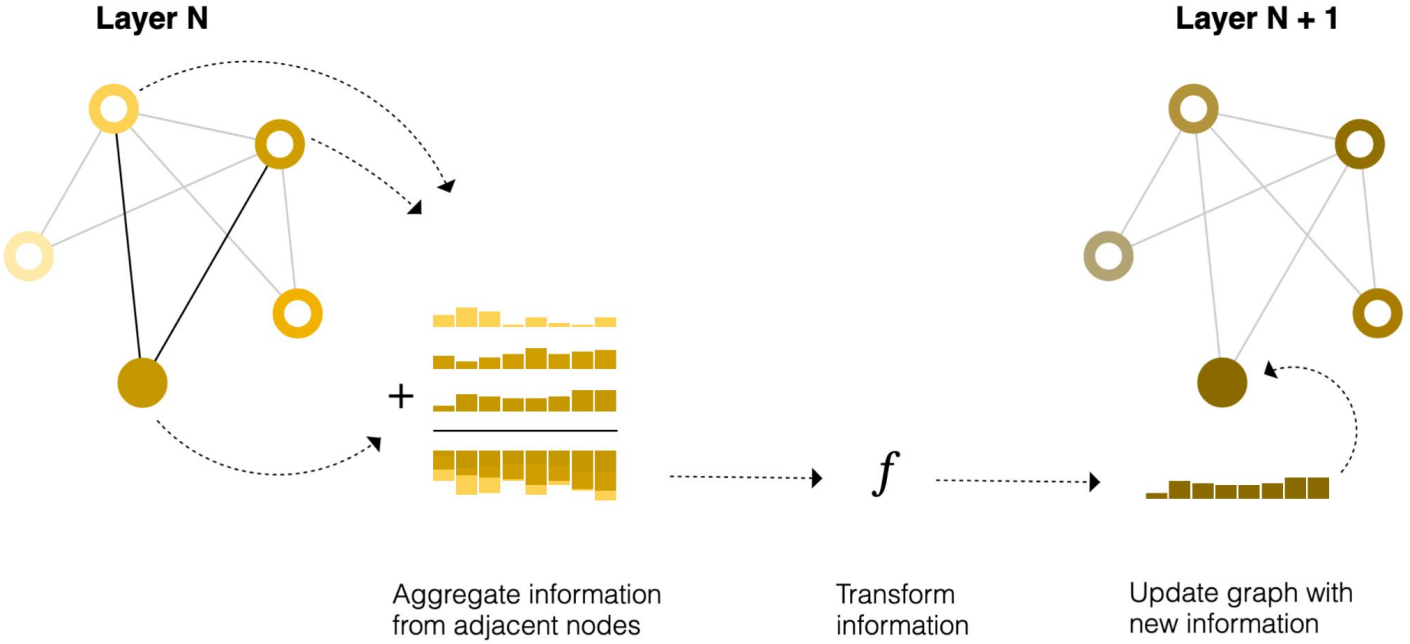
Two main operations: learn embeddings

image credit for GNN

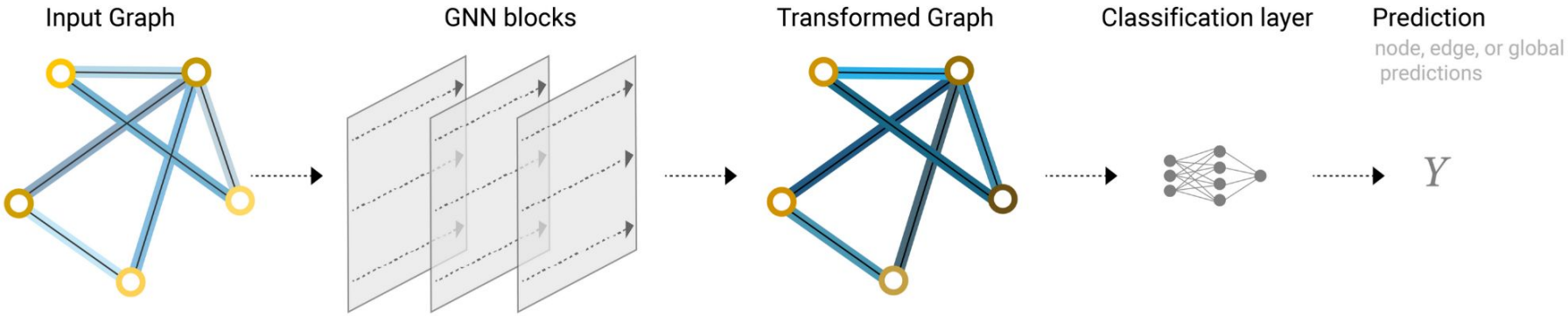


Two main operations: message passing

- For each node in the graph,
  gather all the neighboring node embeddings (or messages)
- Aggregate all messages via an aggregate function (like sum).
- All pooled messages are passed through an update function,
  usually a learned neural network.



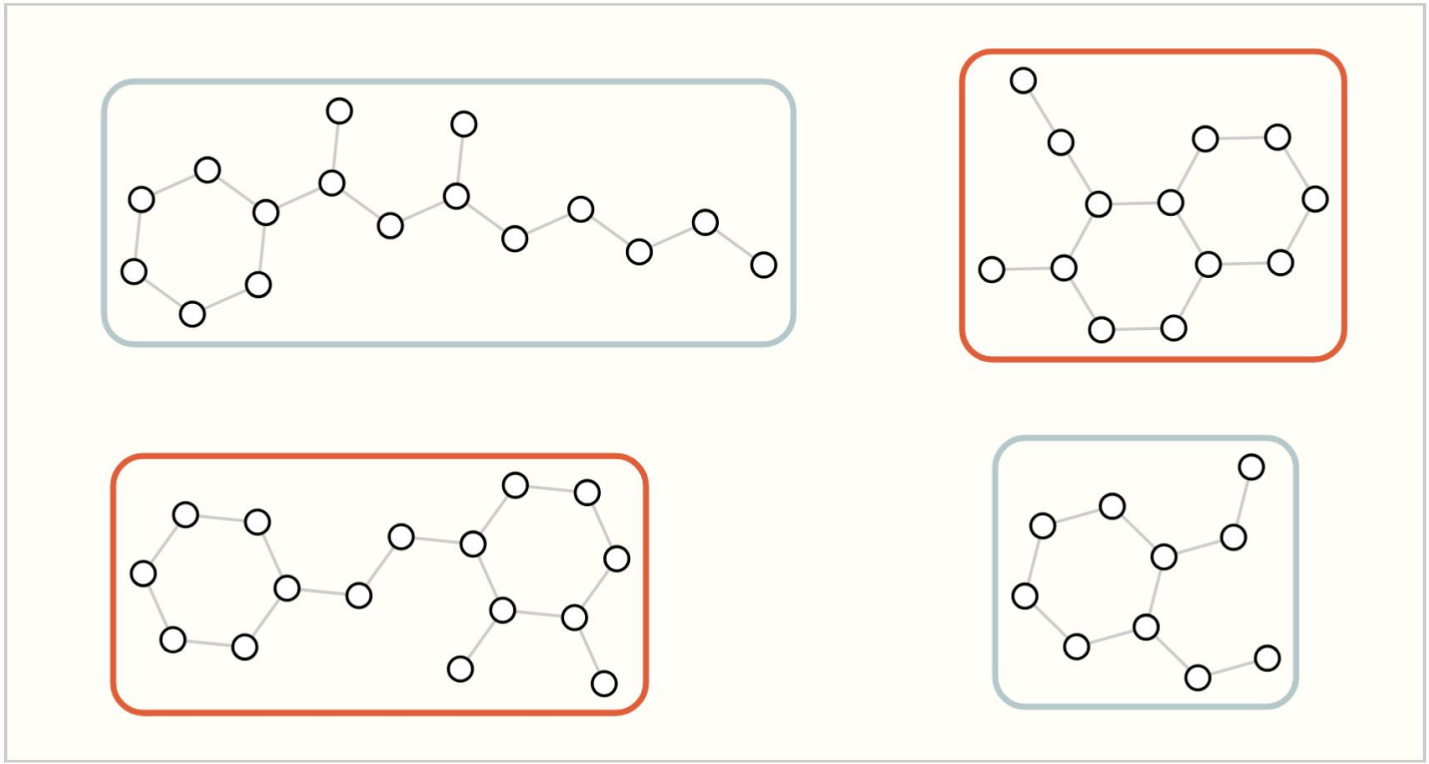
Pipeline



An end-to-end prediction task with a GNN model.

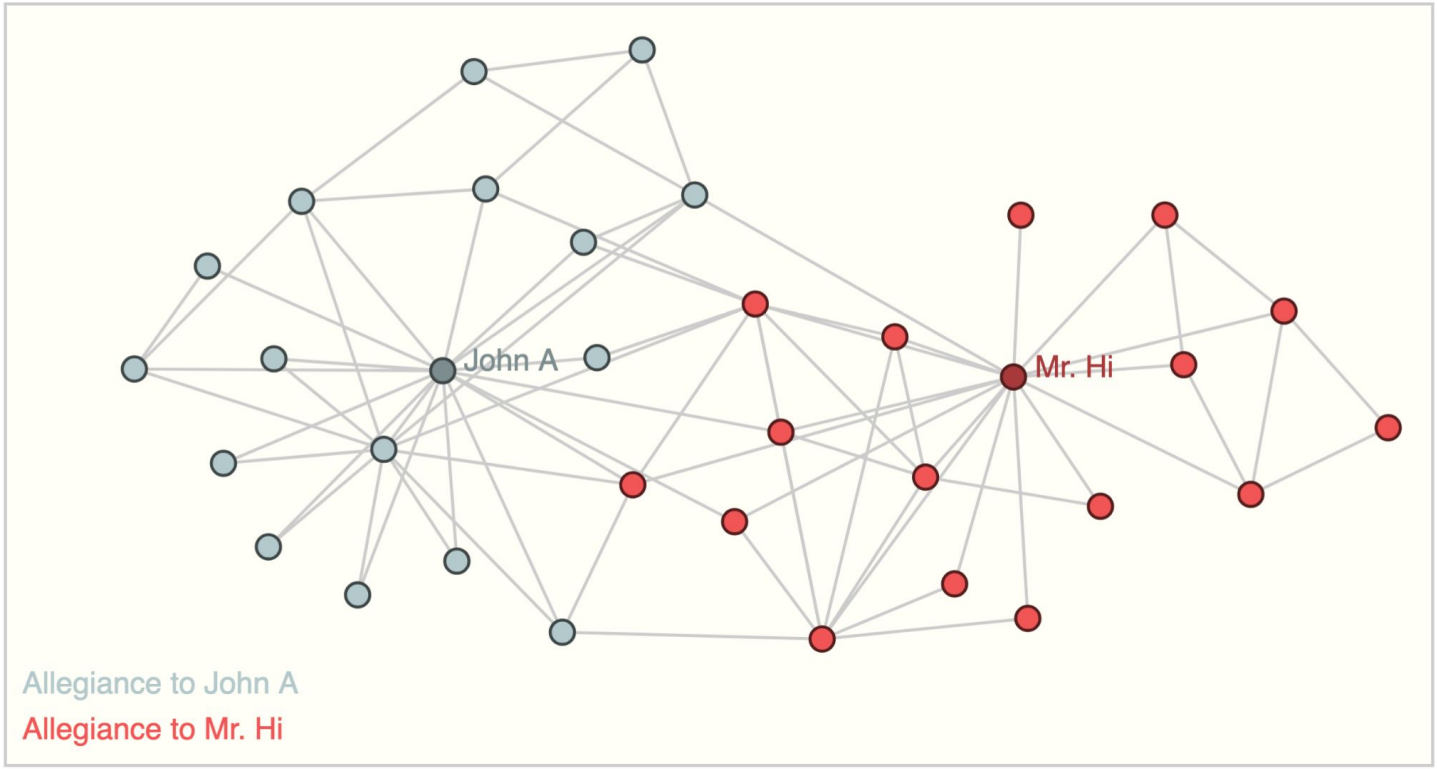
What can we do with graphs? Graph level

image credit for GNN



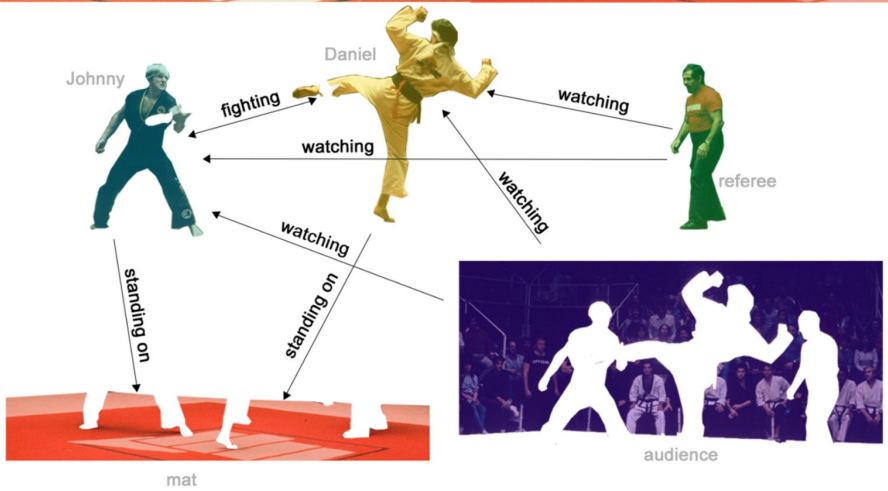
Output: labels for each graph, (e.g., "does the graph contain two rings?")

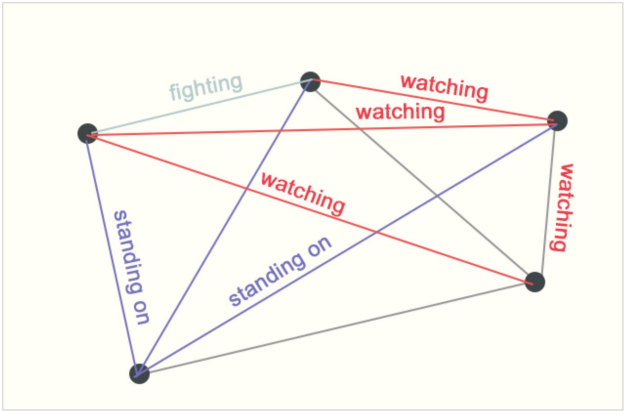
What can we do with graphs? Node level



Output: graph node labels

What can we do with graphs? Edge level

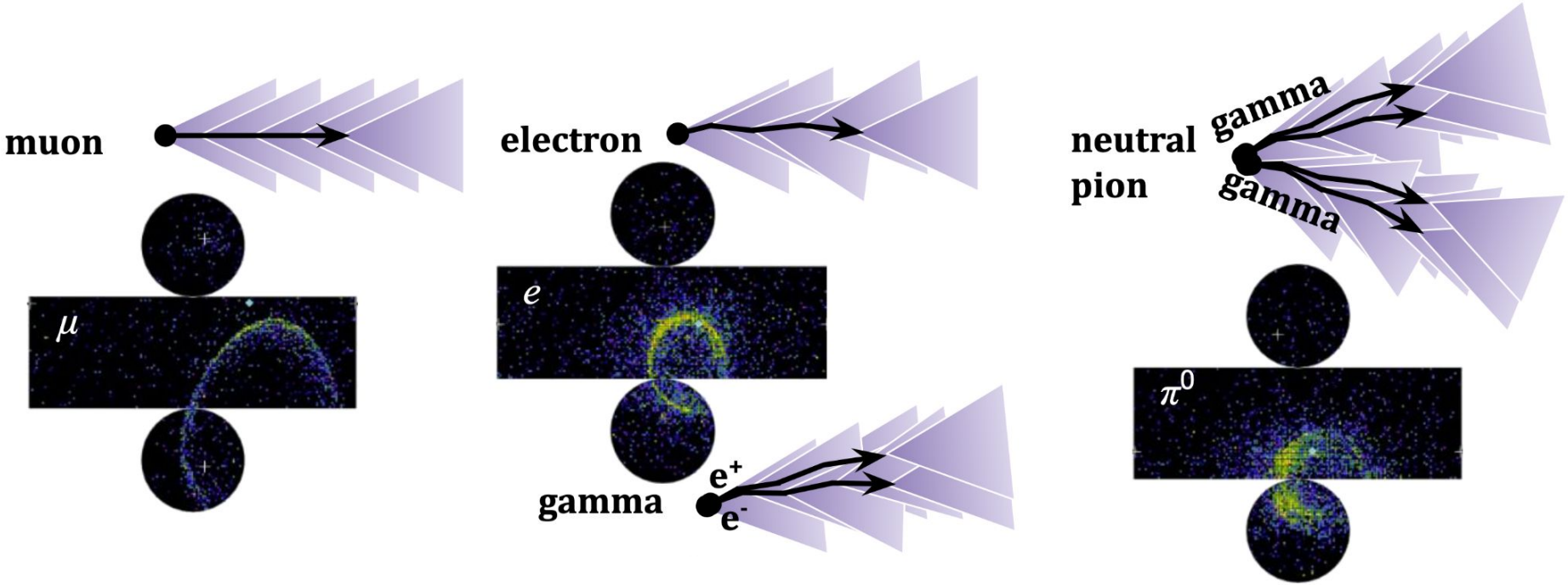




Output: labels for edges

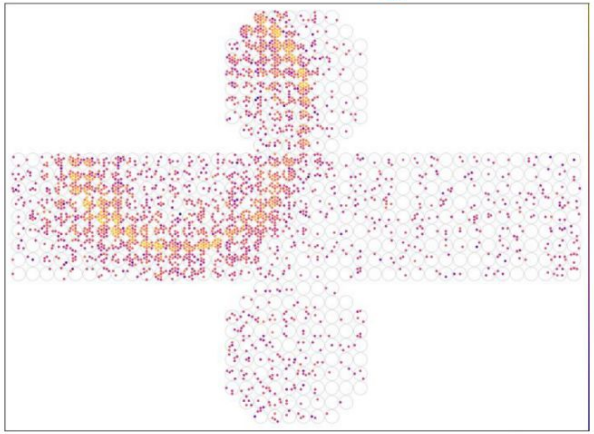
Example: GNN for particle reconstruction

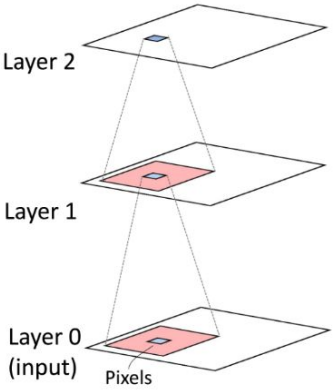
example from HK



Example: GNN for particle reconstruction

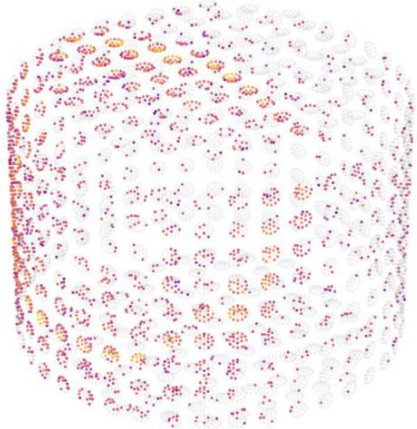
Networks on 2D image-like data

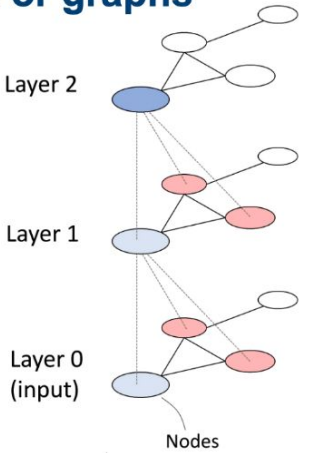




- Fast and well understood CNN-based networks
- Leverage extensive progress in computer vision

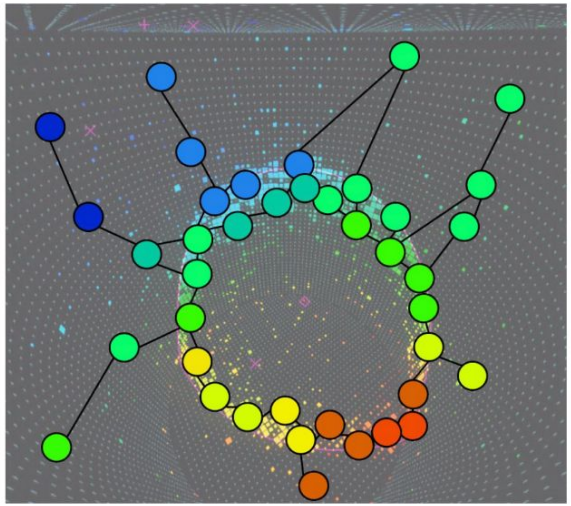
Networks on 3D point-clouds or graphs

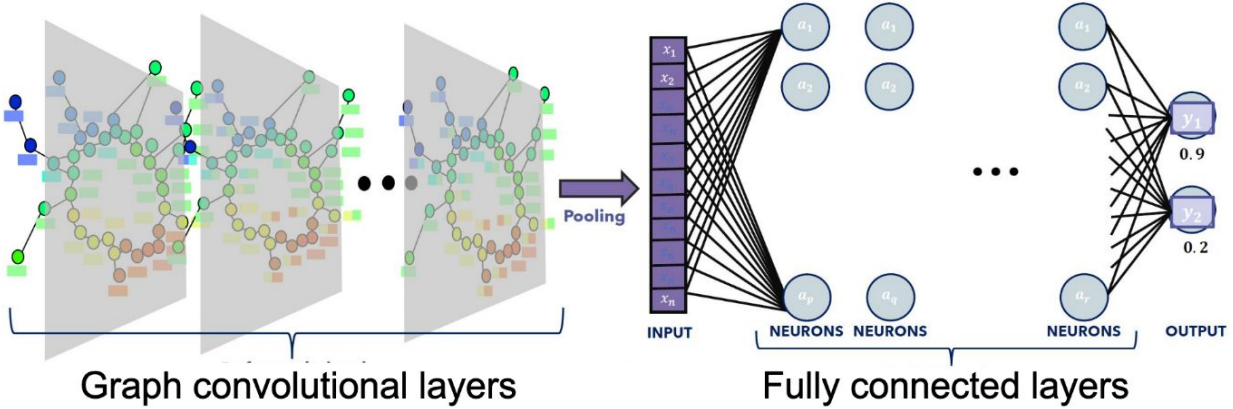




- Retain physical 3D detector geometry
- More complex and challenging to explore

Example: GNN for particle reconstruction





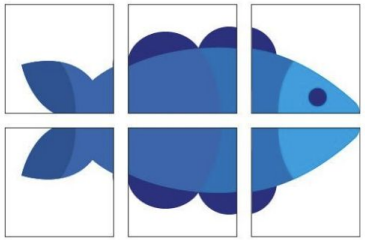
Transformers

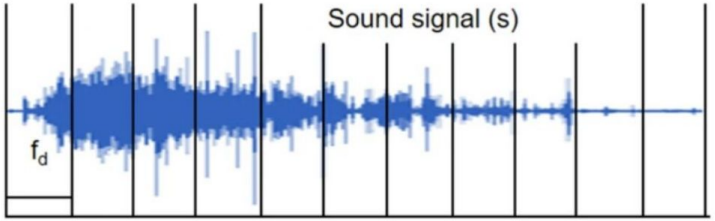
tutorial

Transformers are working with composites (objects made up of elements)

- Each of these parts (represented numerically for computers) is called **token**

Text: The elements of a sentence are words

Images:  Patches

Sound: 

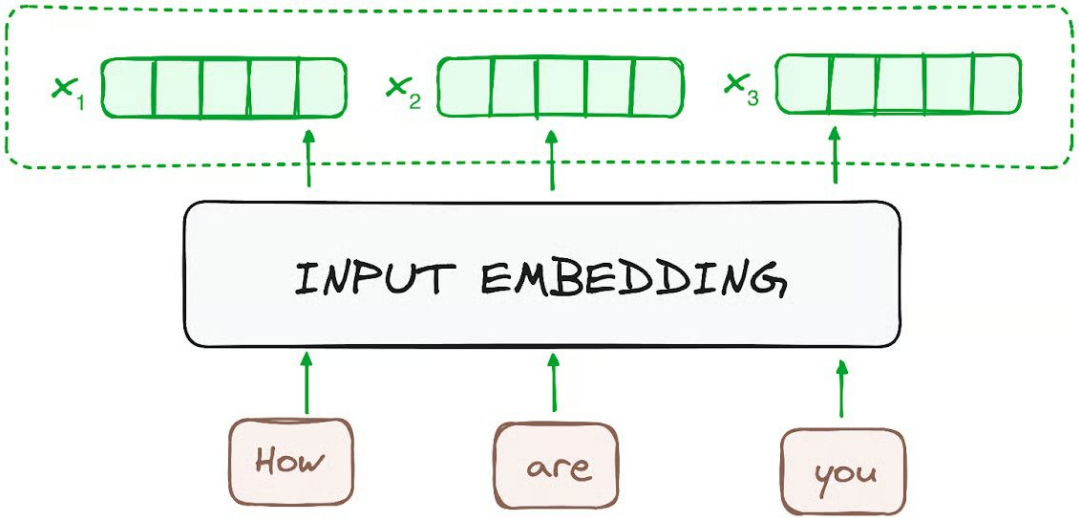
Frames

How to learn the relation between embeddings?

- Each of these parts (represented numerically for computers) is called **token**
- Obtain a representation of the composite. It does not have to be “good” or semantic. An initial representation.

How to learn the relation between embeddings?

- Each of these parts (represented numerically for computers) is called **token**
- Obtain a representation of the composite. It does not have to be “good” or semantic. An initial representation.



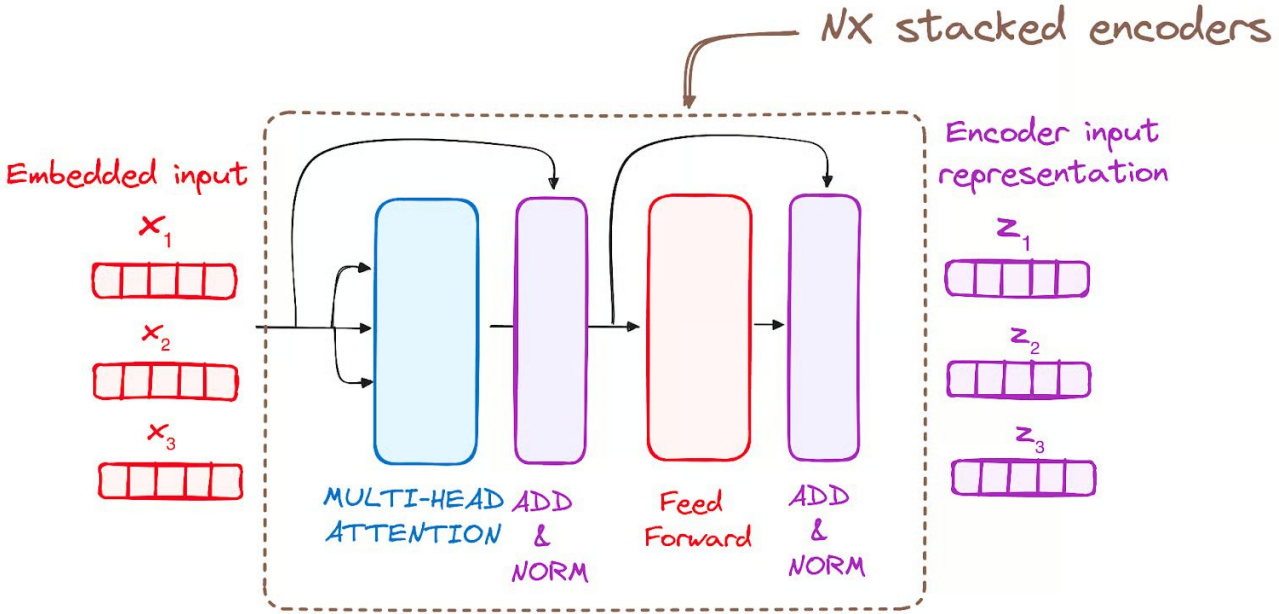
Each token is embedded into a vector of a fixed size.

Can use pretrained layers

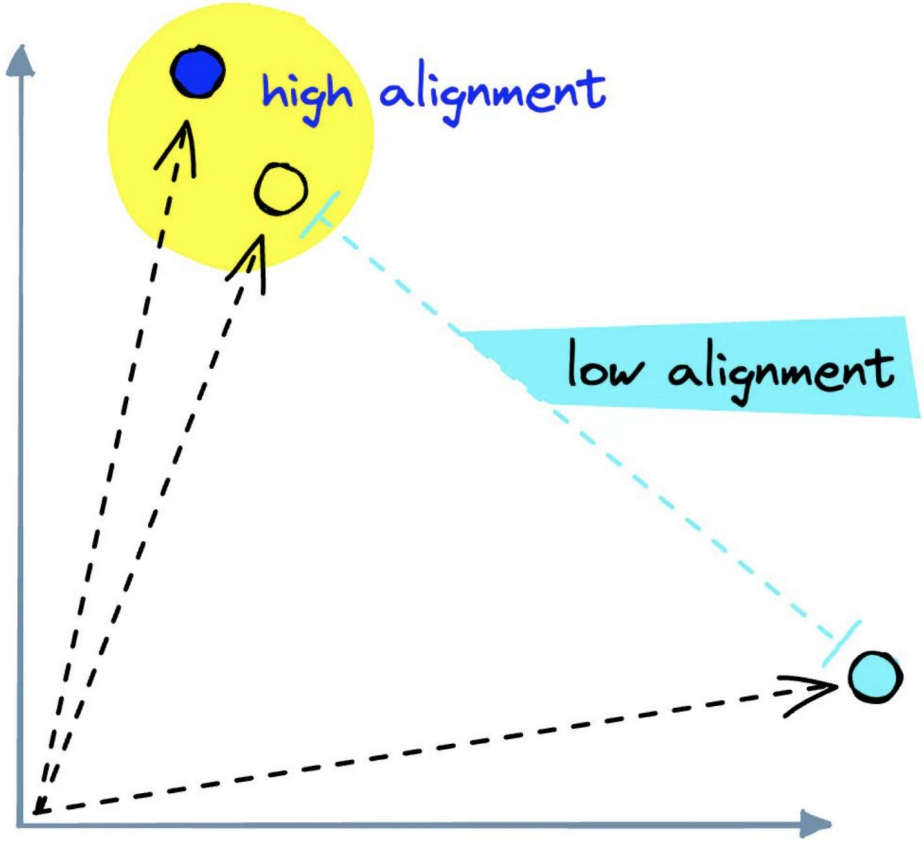
image credit for transformer architecture

How to learn the relation between embeddings?

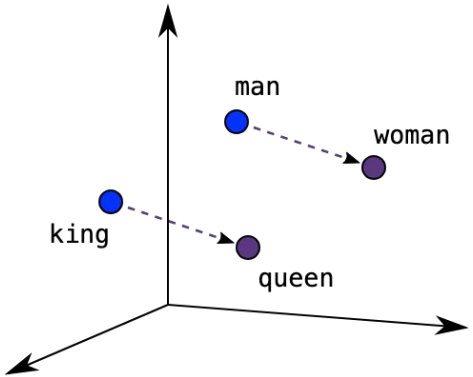
- Each of these parts (represented numerically for computers) is called **token**
- Obtain a representation of the composite. It does not have to be “good” or semantic. An initial representation.
- Feed the representation through several transformer layers to obtain “context-aware” embeddings.



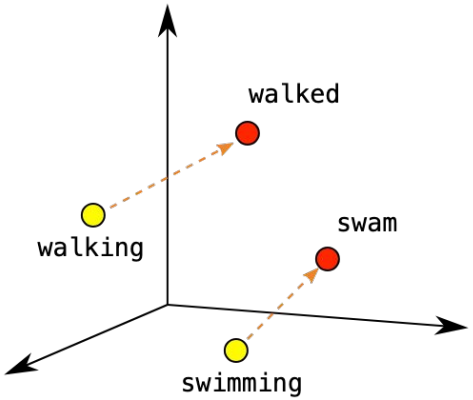
Key idea: learn a “good” representation of each composite = embedding



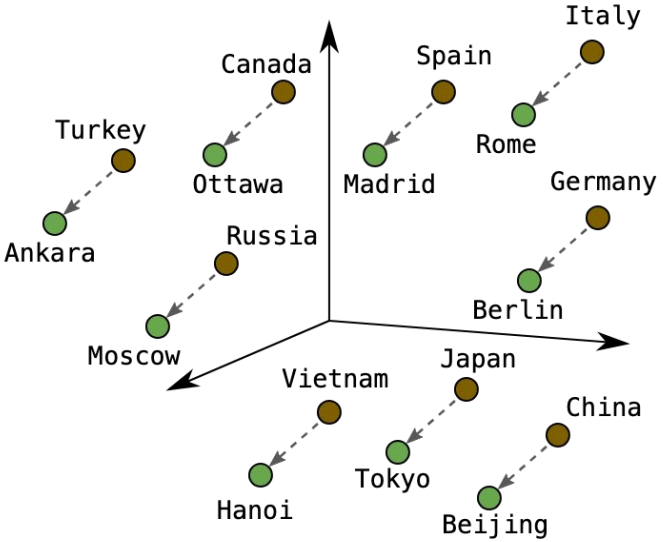
Words embedding



Male-Female

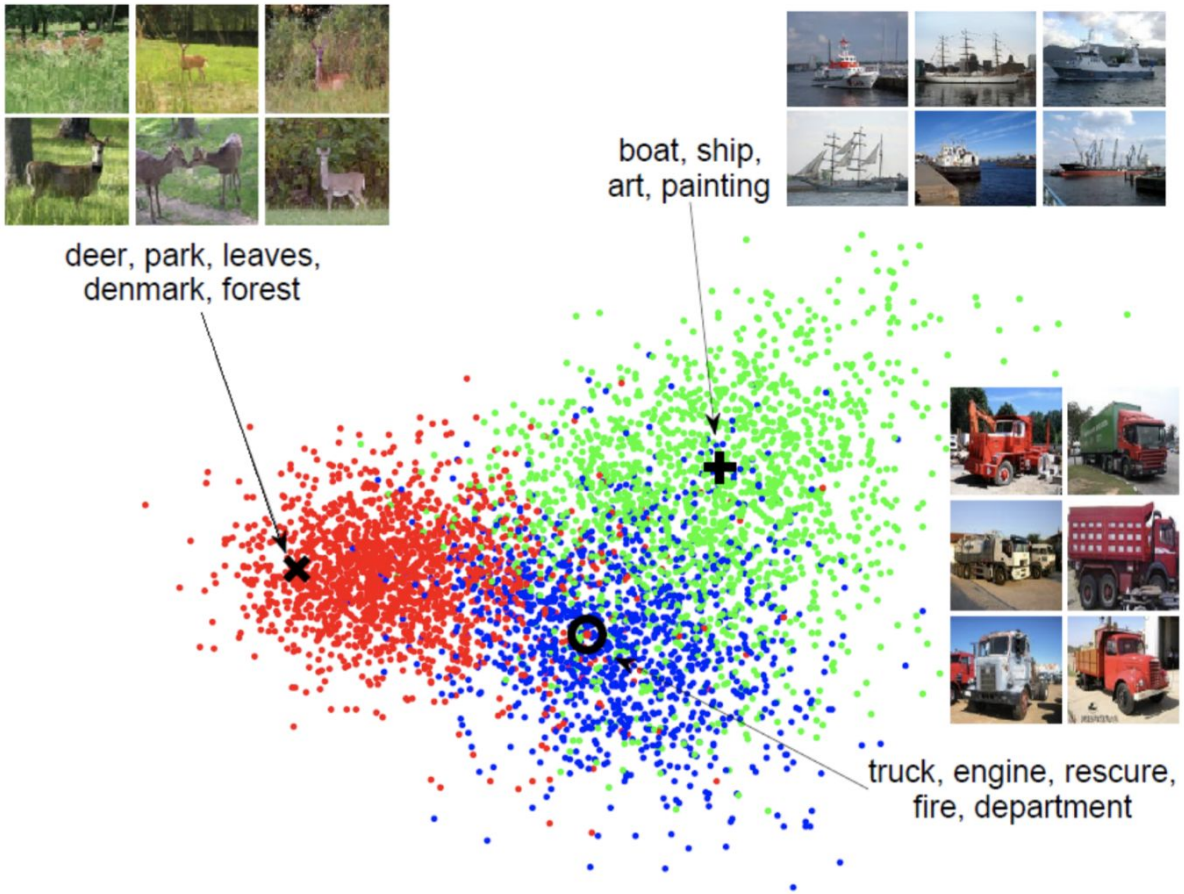


Verb Tense



Country-Capital

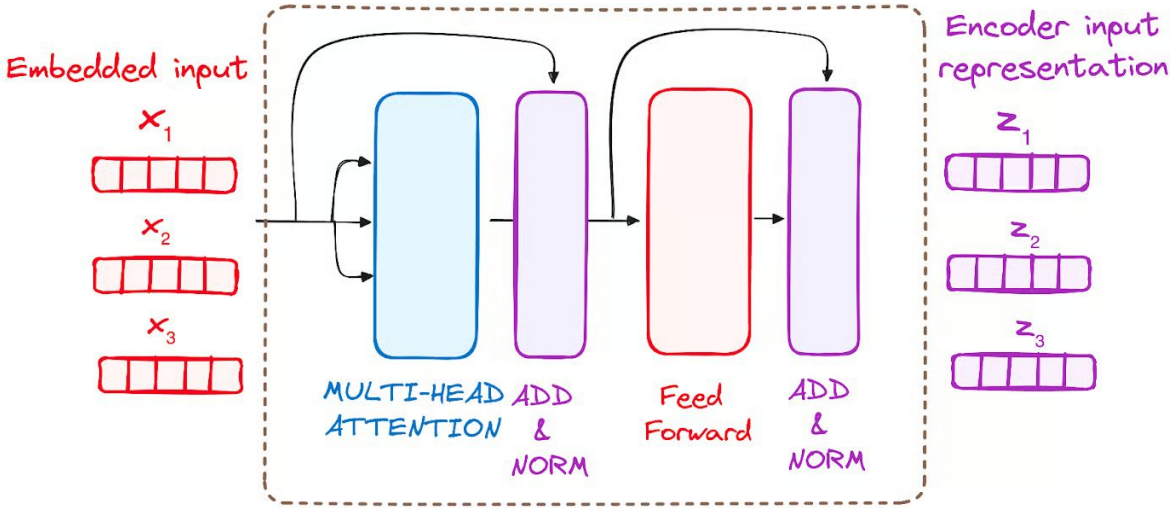
Images embedding



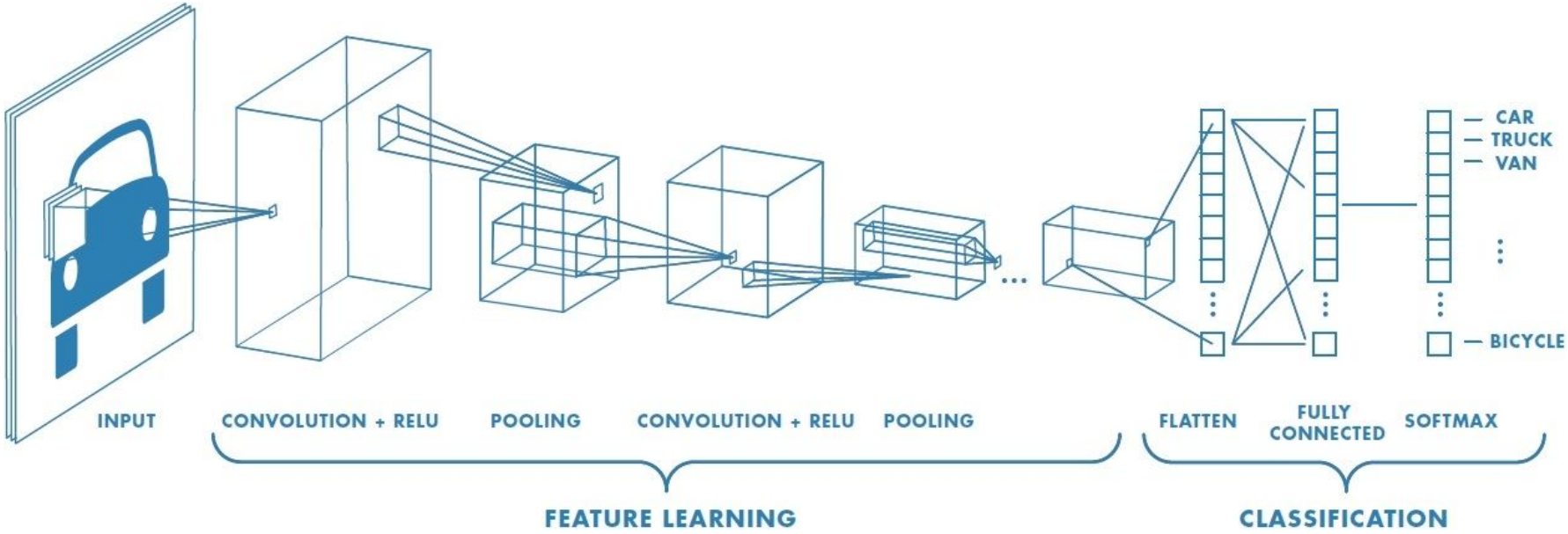
Context-aware embeddings

The humanoid robot did not cross the road, as it was very dangerous.

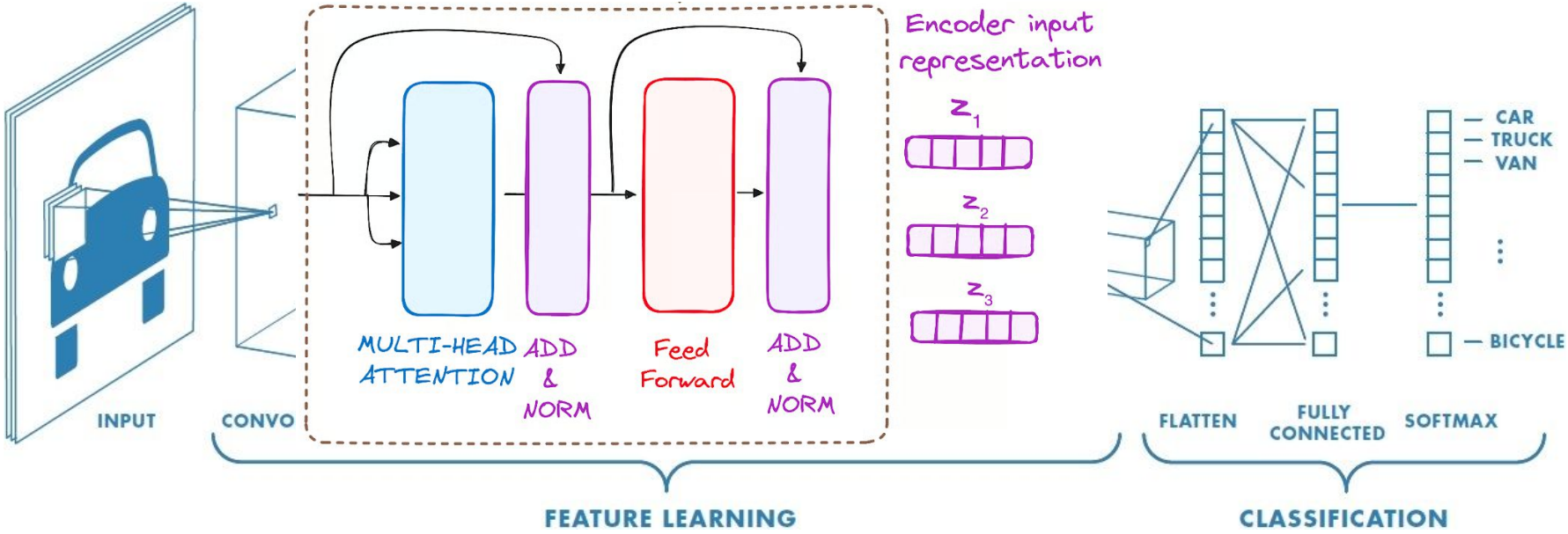
The humanoid robot did not cross the road because it was tired.



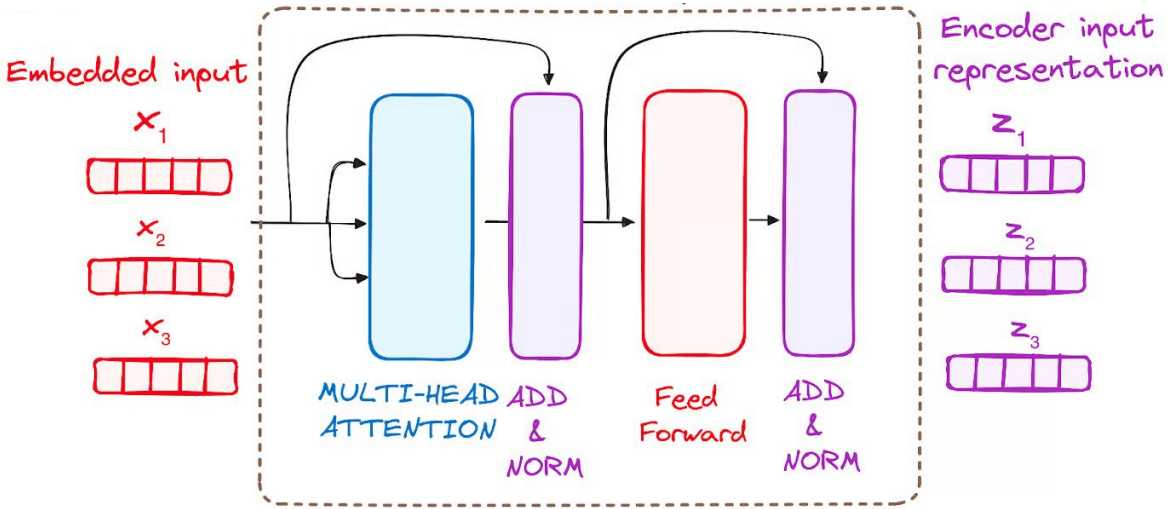
Something we've already seen



Something we've already seen



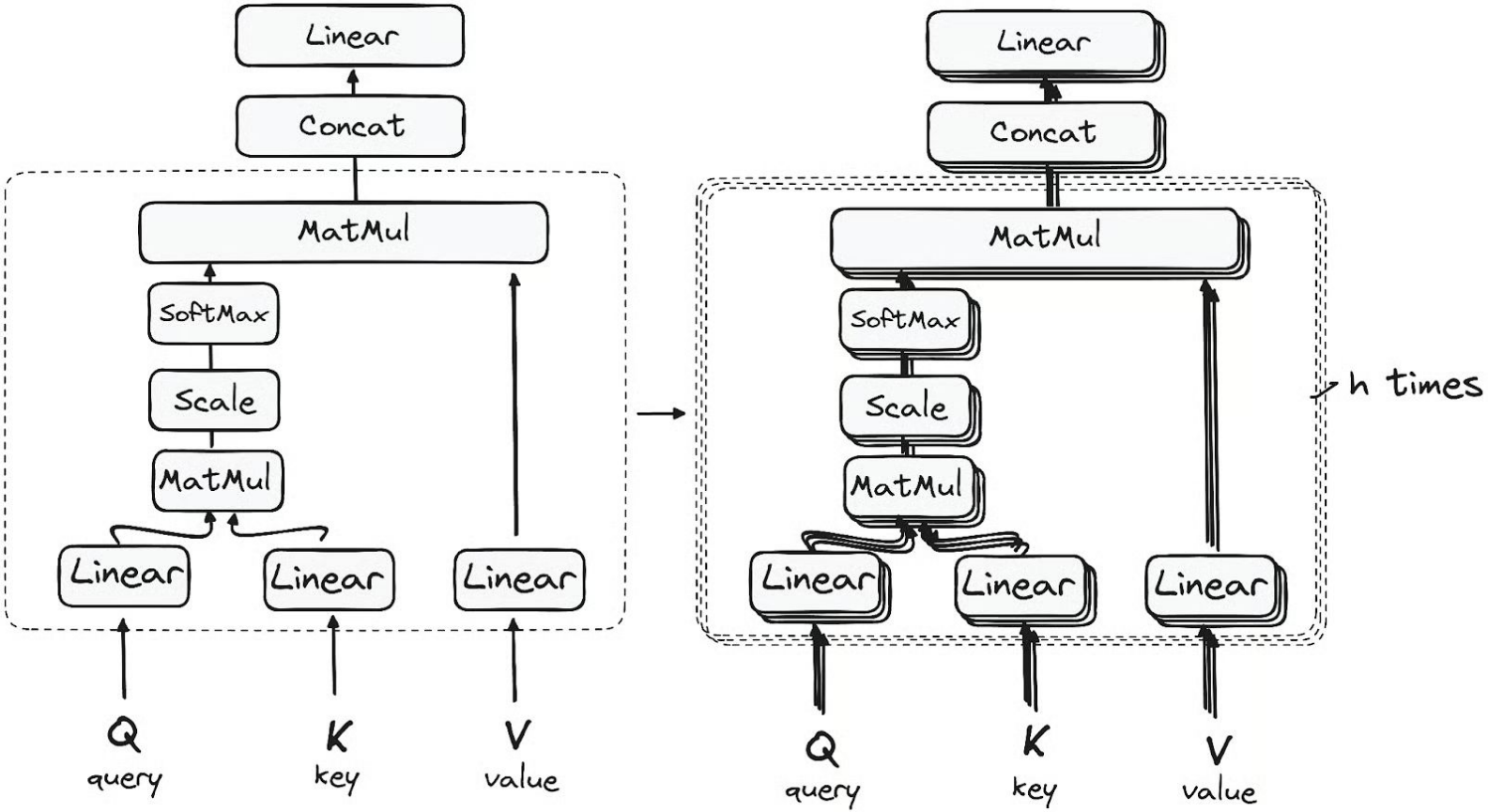
Attention is all we need



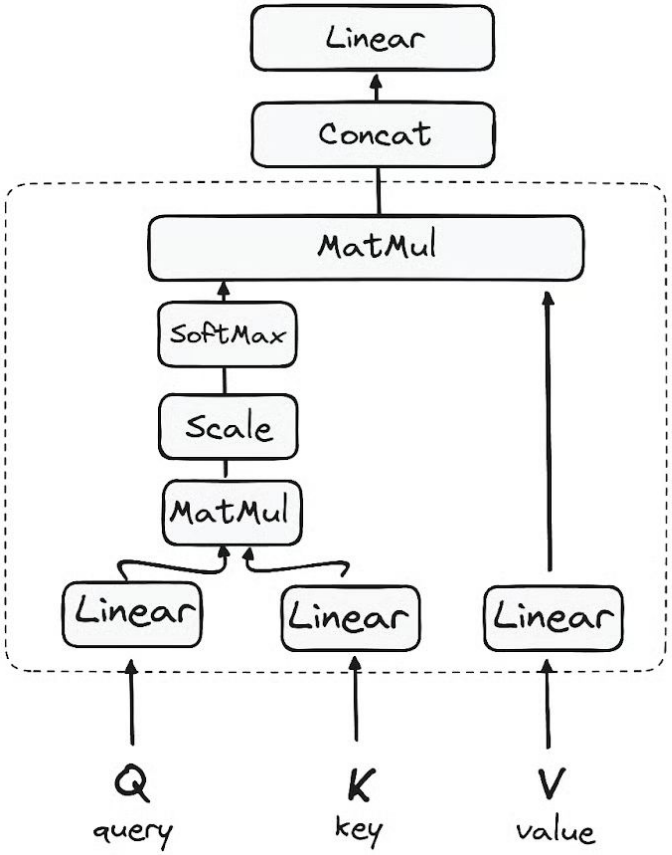
Transformer layers allows for the exchange of information between tokens.

- Each token sends a message to every other token.
- Each token computes how much does it want to take into account the messages of each other token. This is the idea of **attention**.
- Each token moves away from its embedding in the direction signaled by the message of each other token weighted by the attention that it is paying to that token.

Multi-head attention: attention mechanism performed in parallel



From input embeddings to query, key and value



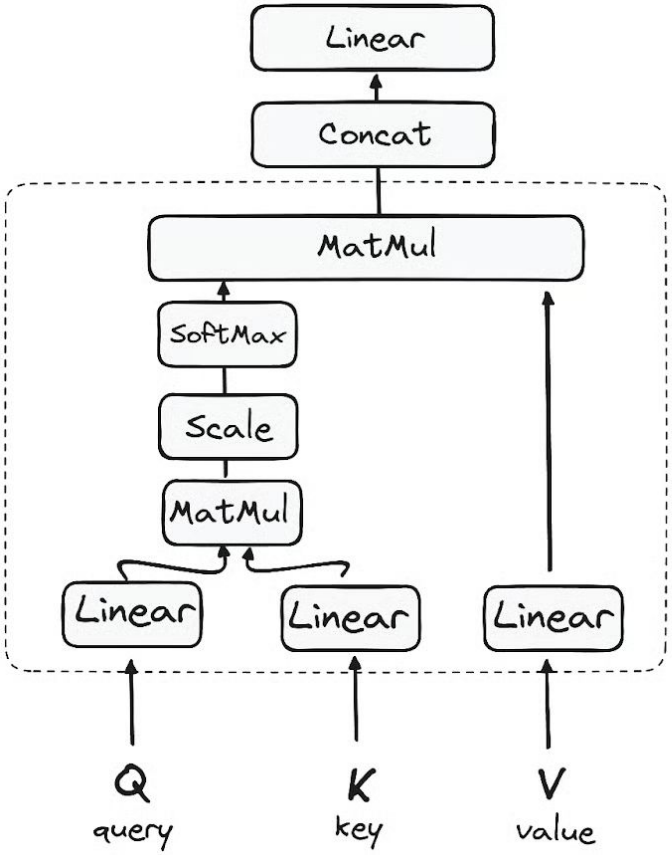
Processing different part of input embedding set while looking at a token:

- **Query**: what am I looking for? / what I want to pay attention to?
- **Key**: what do I contain? / what I offer for others to attend to?
- **Value**: what information I'll pass along if I'm selected?

Query, Key and **Value** are just matrices!

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{V}}$$

How does self-attention work

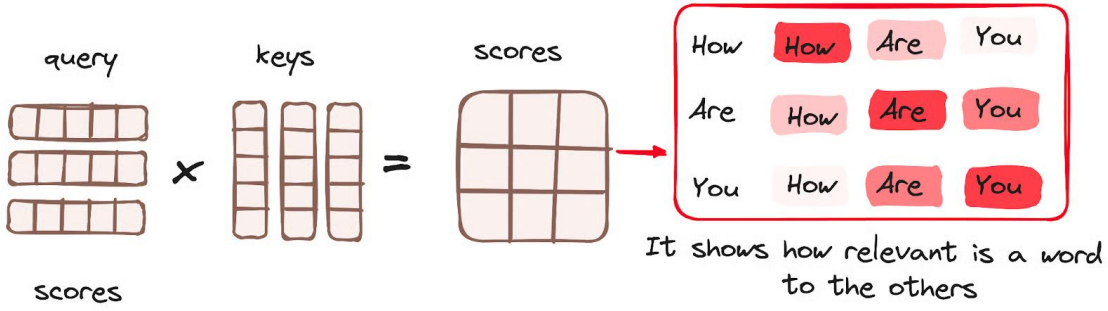


Processing different part of input embedding set while looking at a token:

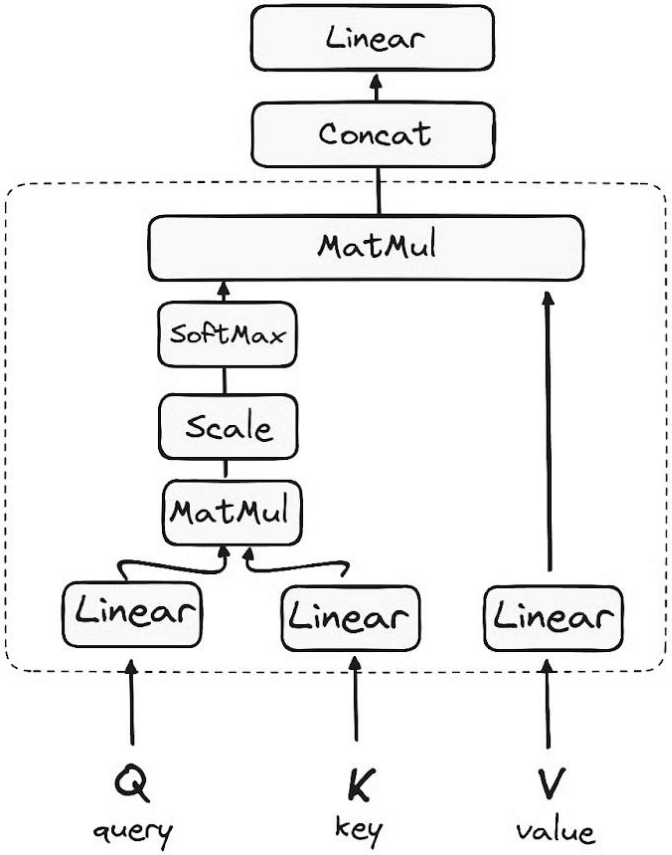
- **Query**: what am I looking for? / what I want to pay attention to?
- **Key**: what do I contain? / what I offer for others to attend to?
- **Value**: what information I'll pass along if I'm selected?

Query, Key and **Value** are just matrices!

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{V}}$$



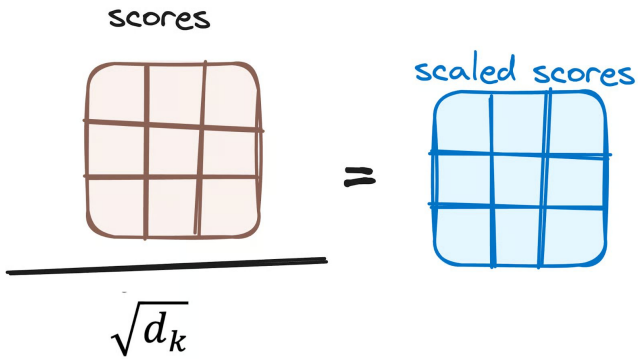
How does self-attention work



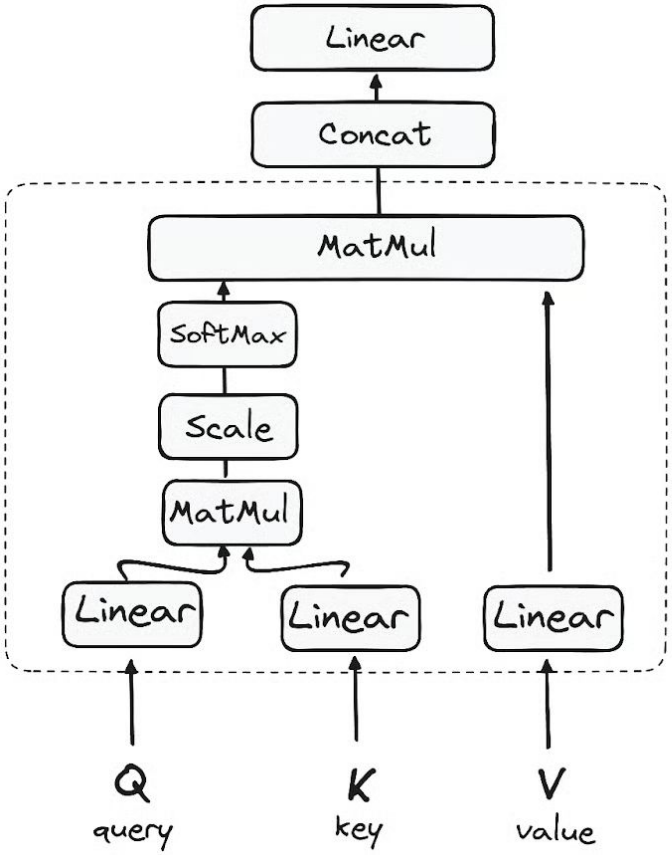
Processing different part of input embedding set while looking at a token:

- **Query**: what am I looking for? / what I want to pay attention to?
- **Key**: what do I contain? / what I offer for others to attend to?
- **Value**: what information I'll pass along if I'm selected?

Query, Key and **Value** are just matrices!



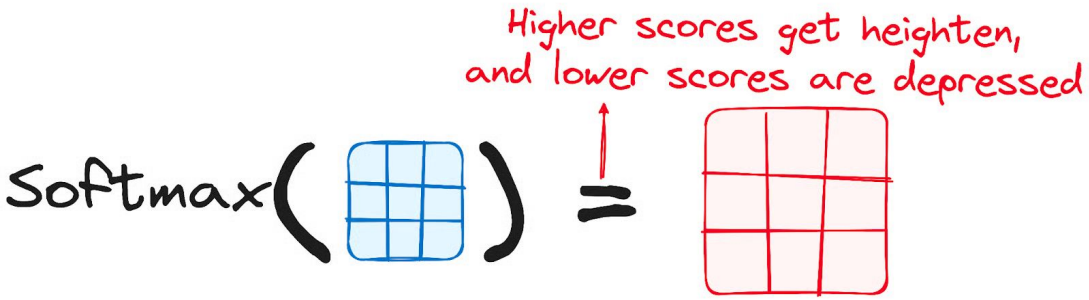
How does self-attention work



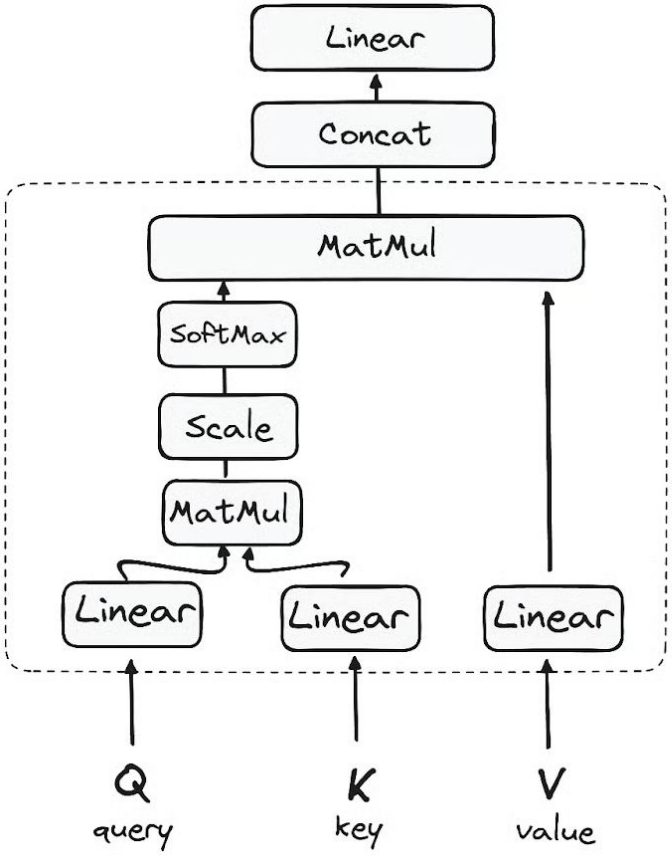
Processing different part of input embedding set while looking at a token:

- **Query**: what am I looking for? / what I want to pay attention to?
- **Key**: what do I contain? / what I offer for others to attend to?
- **Value**: what information I'll pass along if I'm selected?

Query, Key and **Value** are just matrices!



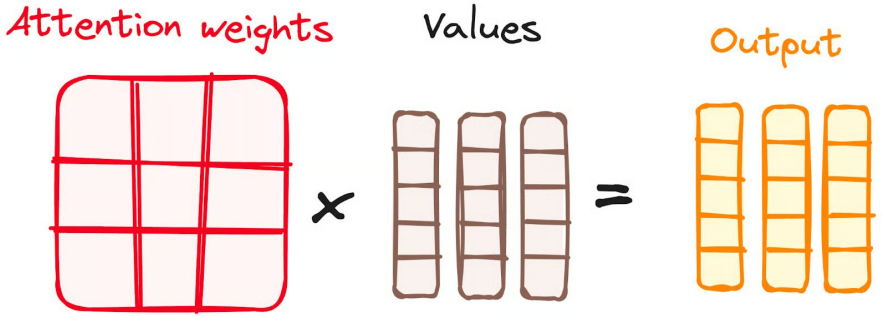
How does self-attention work



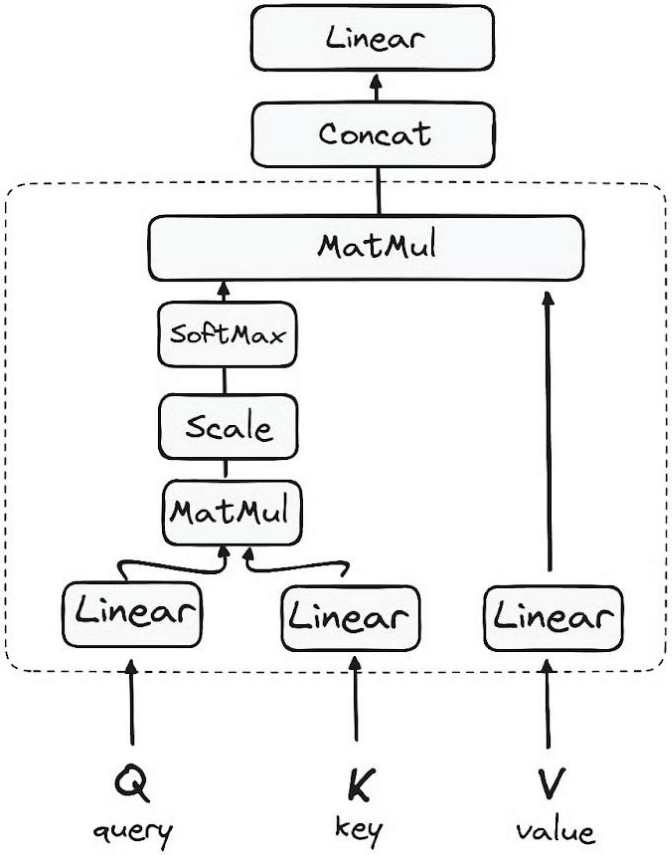
Processing different part of input embedding set while looking at a token:

- **Query**: what am I looking for? / what I want to pay attention to?
- **Key**: what do I contain? / what I offer for others to attend to?
- **Value**: what information I'll pass along if I'm selected?

Query, Key and **Value** are just matrices!



How does self-attention work



Processing different part of input embedding set while looking at a token:

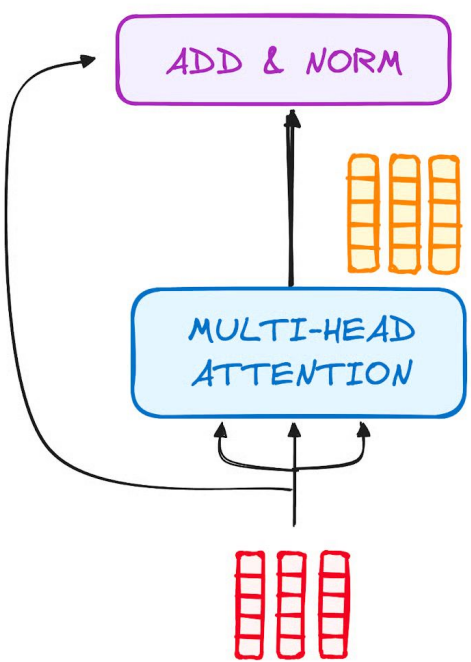
- **Query**: what am I looking for? / what I want to pay attention to?
- **Key**: what do I contain? / what I offer for others to attend to?
- **Value**: what information I'll pass along if I'm selected?

Query, Key and **Value** are just matrices!

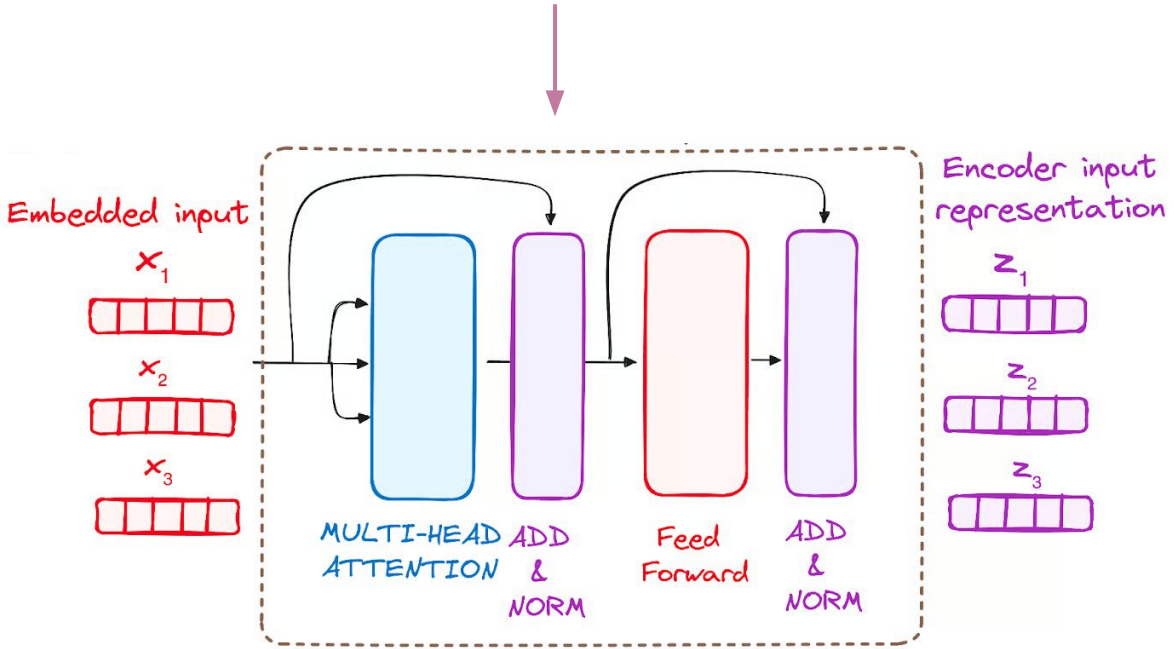
$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{V}}$$

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

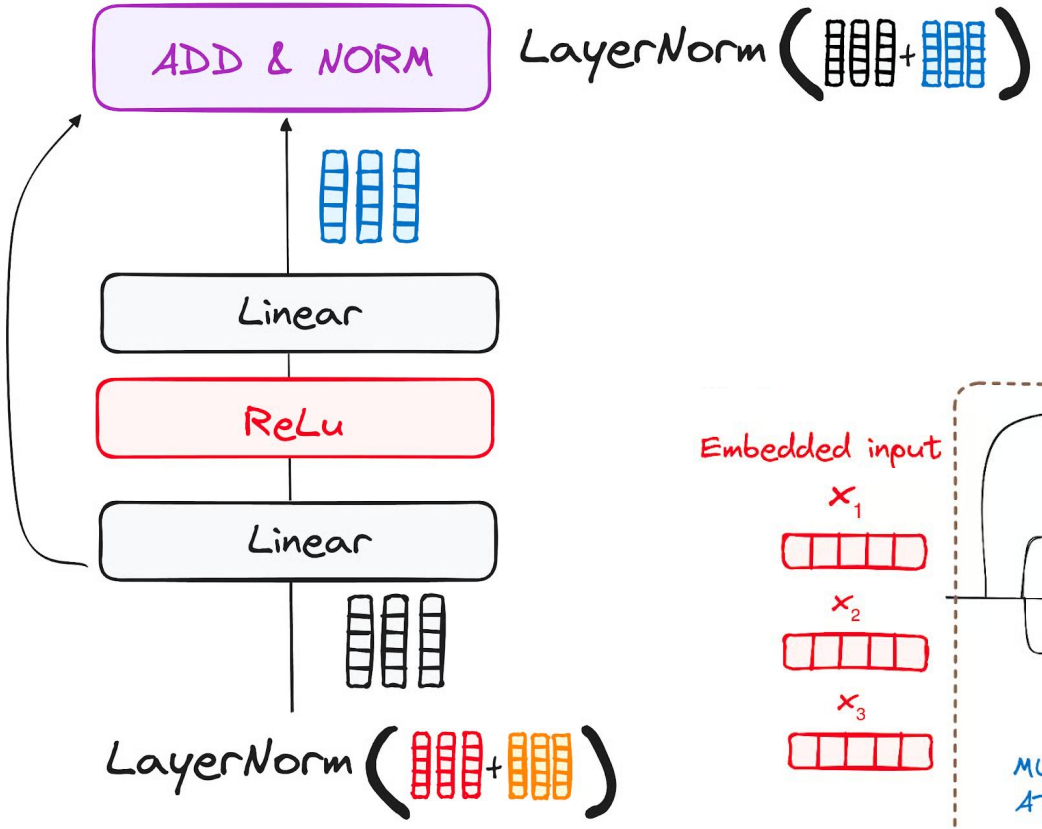
How does self-attention work

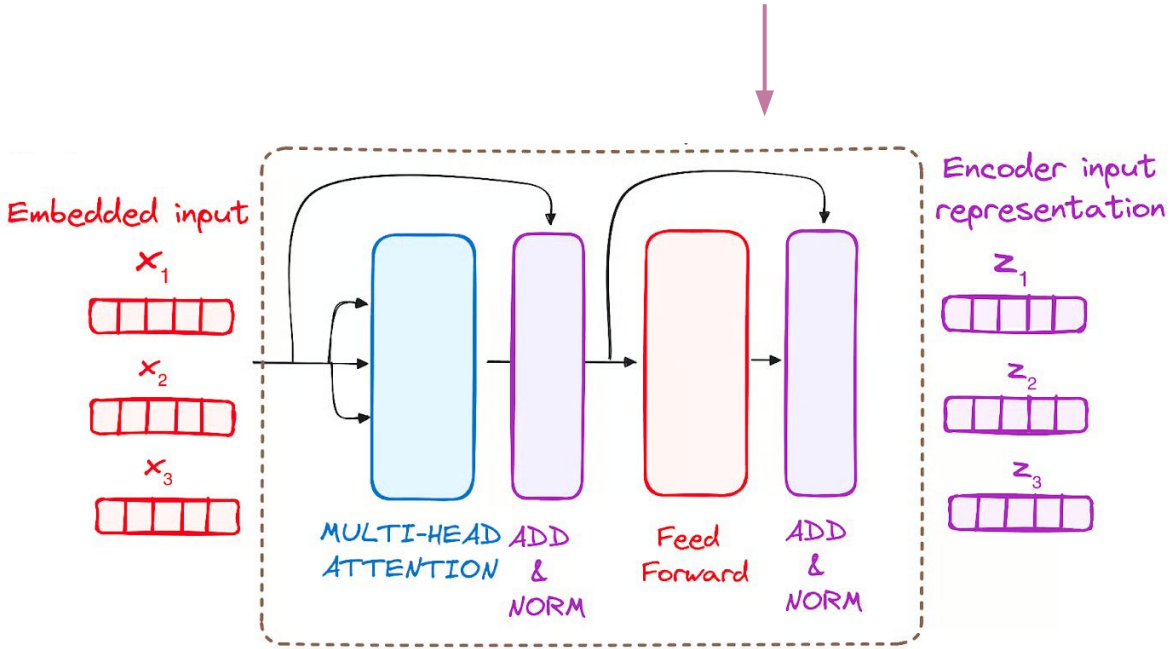


LayerNorm(+)

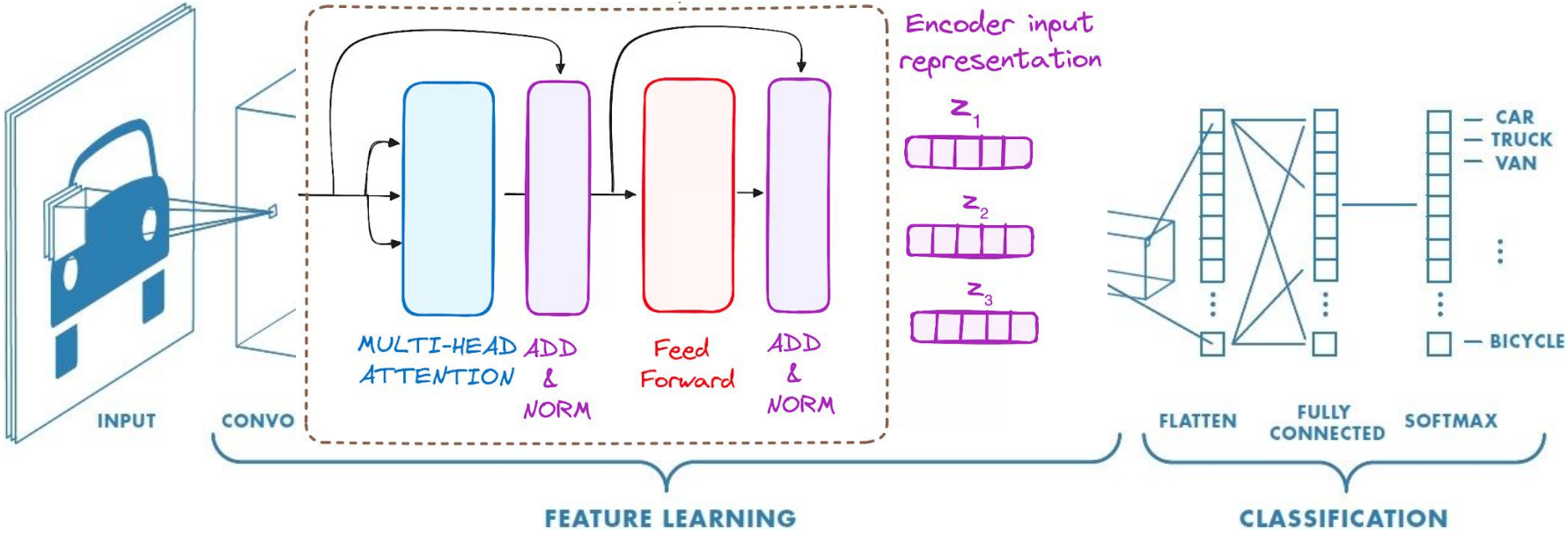


How does self-attention work

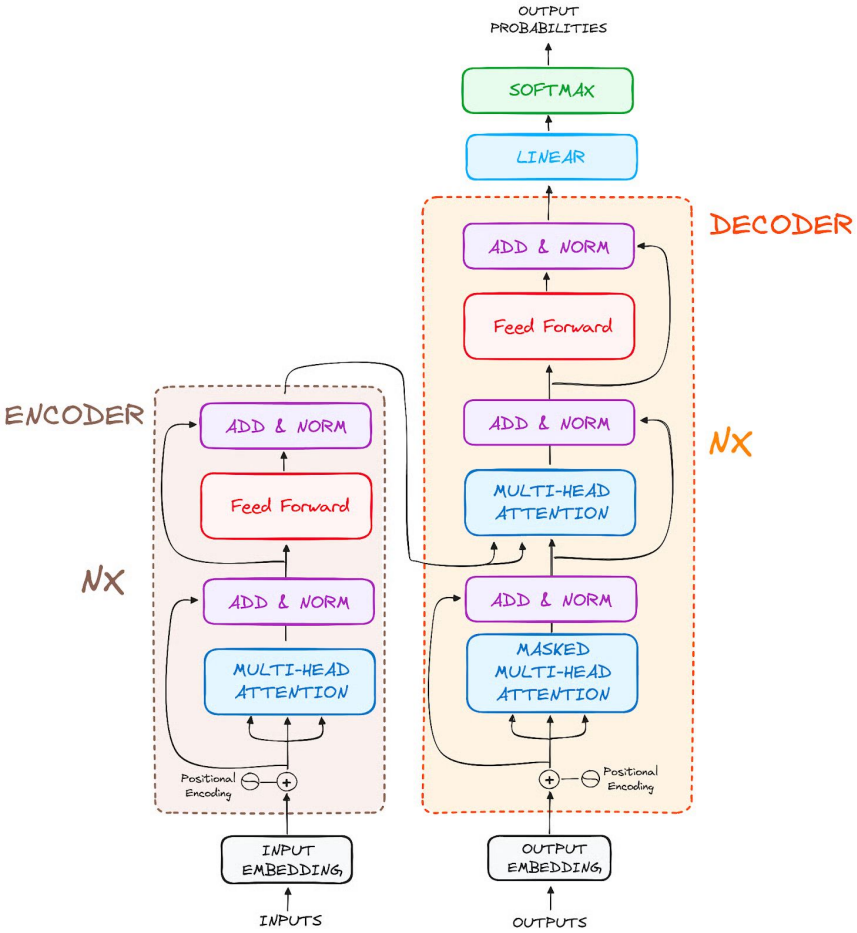




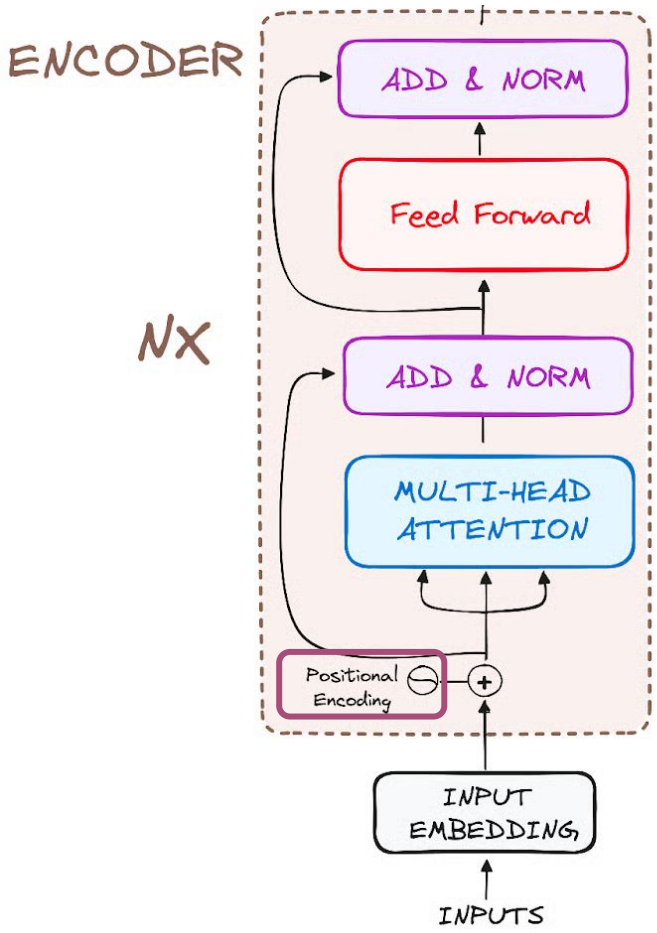
Transformer for feature extraction



Transformer for generative tasks

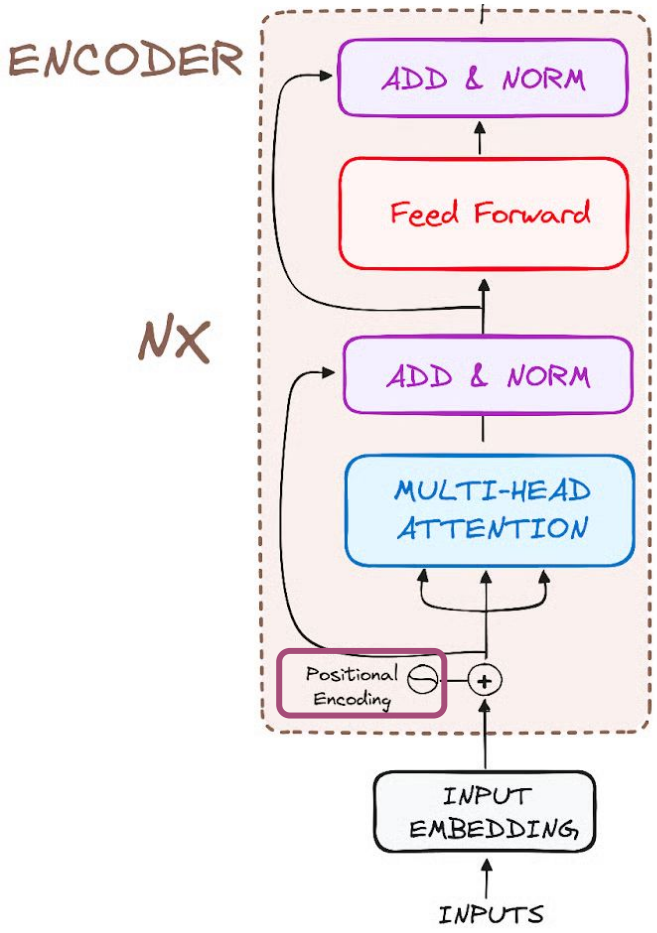


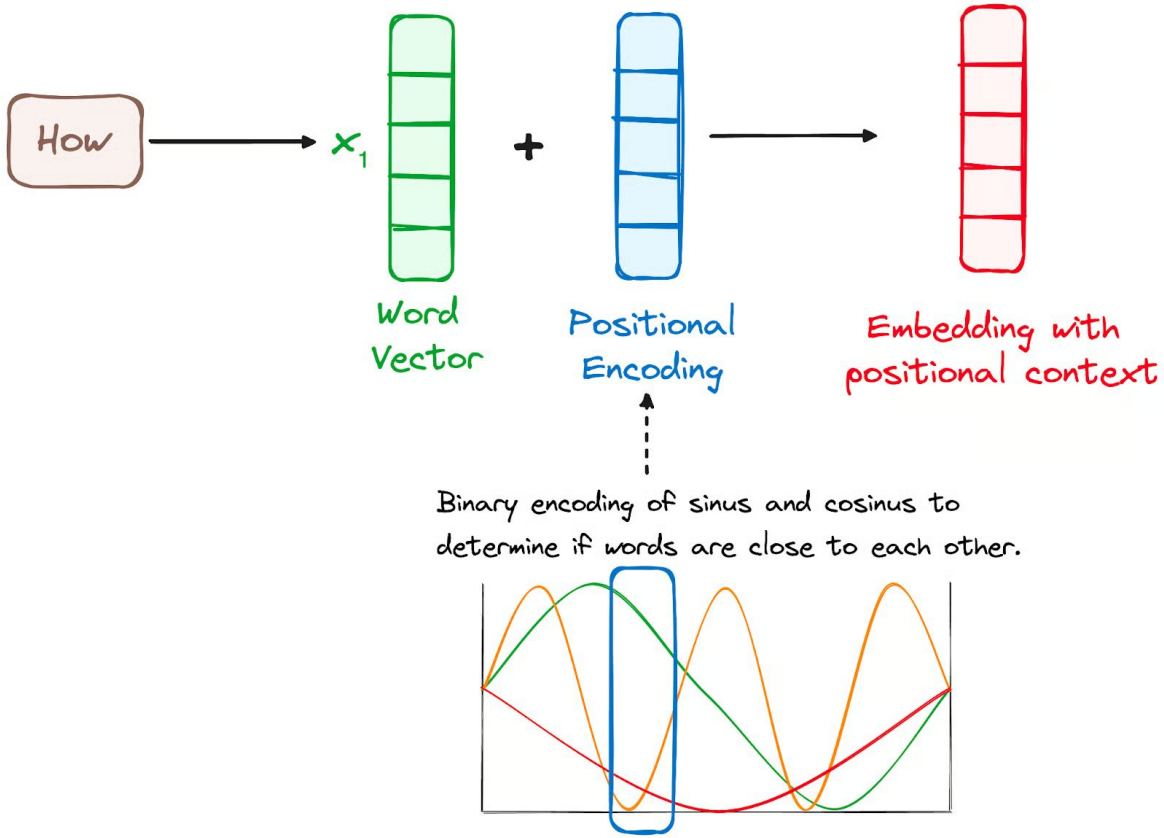
Self-attention is permutation equivariant



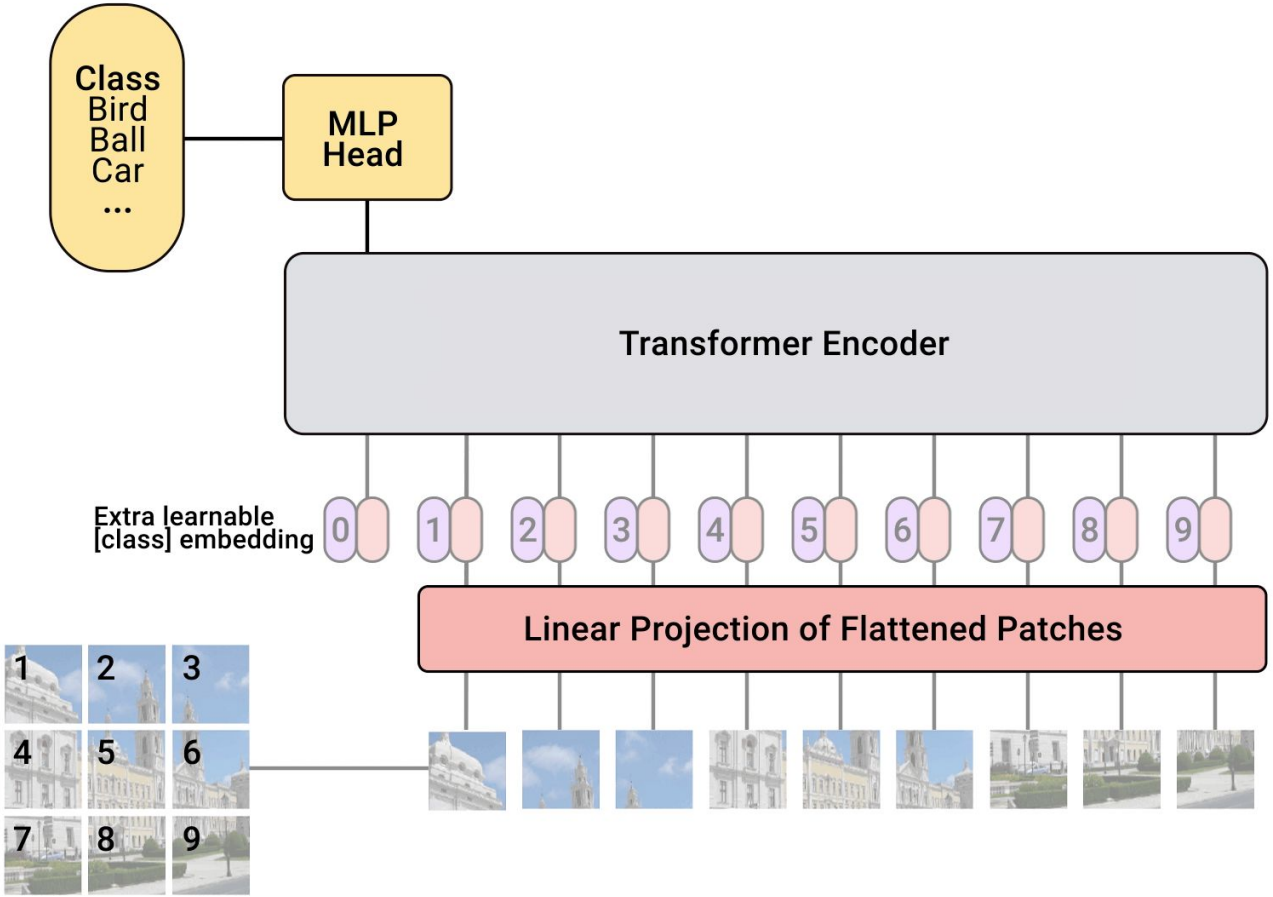
Each token embedding goes through the same MLP independently

Self-attention is permutation equivariant without positional encoding

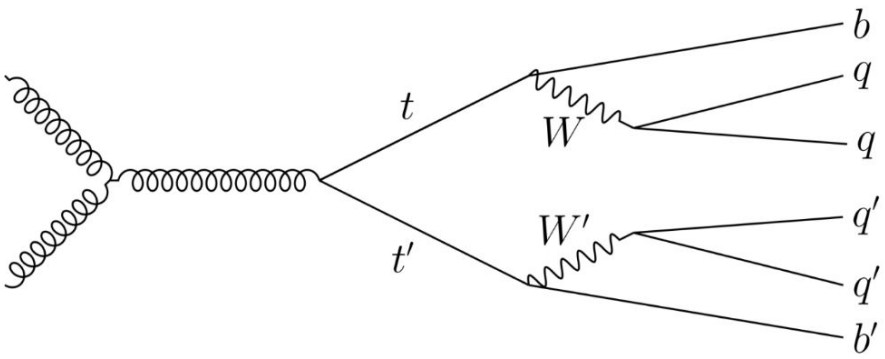




Transformer for image classification (ViT)



Transformer for jet matching (SPANet)



Set of Jets

$$\{j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8\}$$

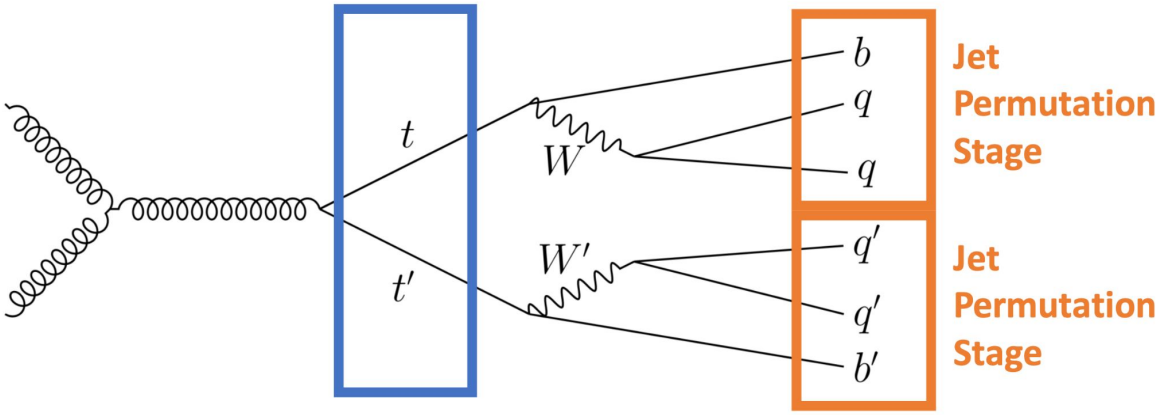
Match Jets to Parton Labels

$$\{b, q', \emptyset, q', b', \emptyset, q, q\}$$

Targets

Garbage Jets

Transformer for jet matching (SPANet)



Particle Classification Stage

$$q_1 q_2 b q_1' q_2' b' \leftrightarrow q_2 q_1 b q_1' q_2' b'$$
$$q_1 q_2 b q_1' q_2' b' \leftrightarrow q_1 q_2 b q_2' q_1' b'$$

← We call these **jet symmetries.** We will handle this with **attention**.

$$\underset{\mathcal{T}_1}{q_1} \underset{\mathcal{T}_2}{q_2} b \underset{\mathcal{T}_2}{q_1'} \underset{\mathcal{T}_2}{q_2'} \underset{\mathcal{T}_2}{b'} \leftrightarrow \underset{\mathcal{T}_2}{q_1'} \underset{\mathcal{T}_2}{q_2'} \underset{\mathcal{T}_2}{b'} \underset{\mathcal{T}_1}{q_1} \underset{\mathcal{T}_1}{q_2} b$$

← We call this **particle symmetries.** We will handle this with a **special loss function**.

Visualisation here