# Moderne Methoden der Datenanalyse:

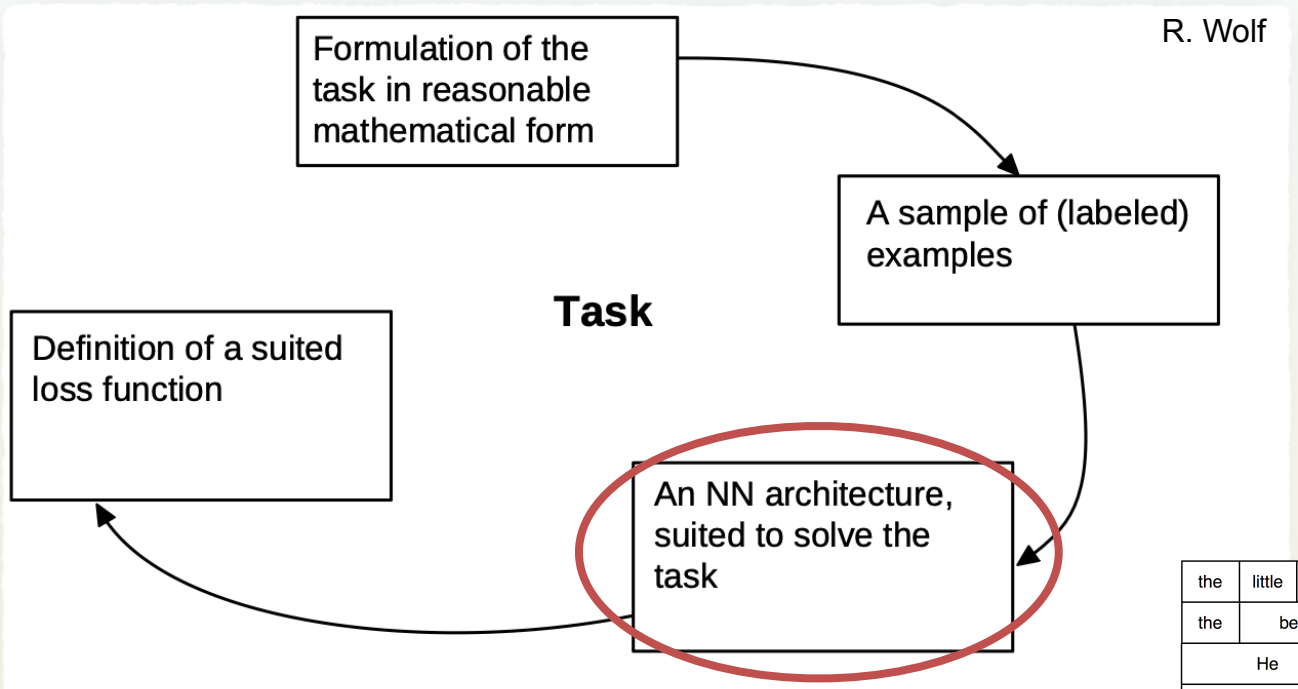
## - Attention and transformer -

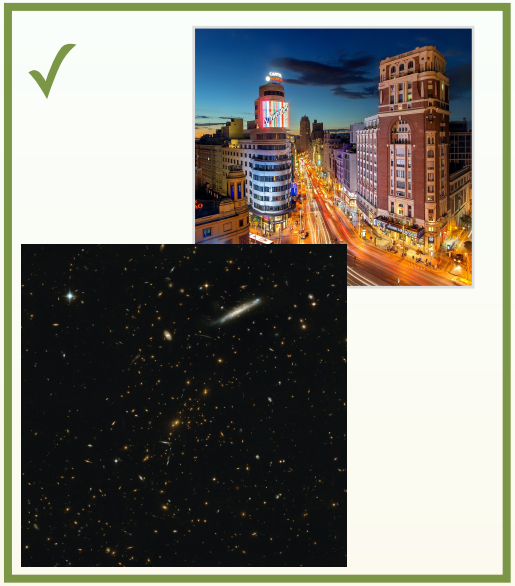
## - Graph neural networks -

28.7.2023
Jan Kieseler

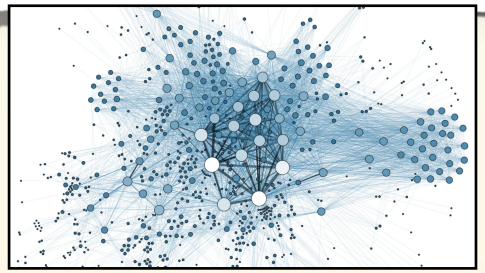
This is a very rich topic, with enough content for whole courses.
Please consider the following teasers

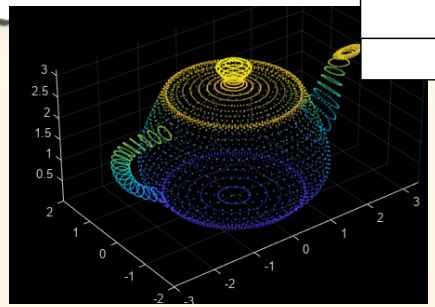
# Recap

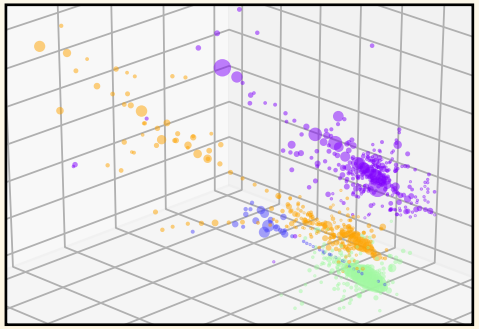


| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| the | little | bear | saw | the | fine | fat | trout | in | the | brook |
| the | bear | | saw | the | trout | | | in | it | |
| He | | | saw | it | | | | there | | |
| He | | | ran | | | | | there | | |
| He | | | ran | | | | | | | |

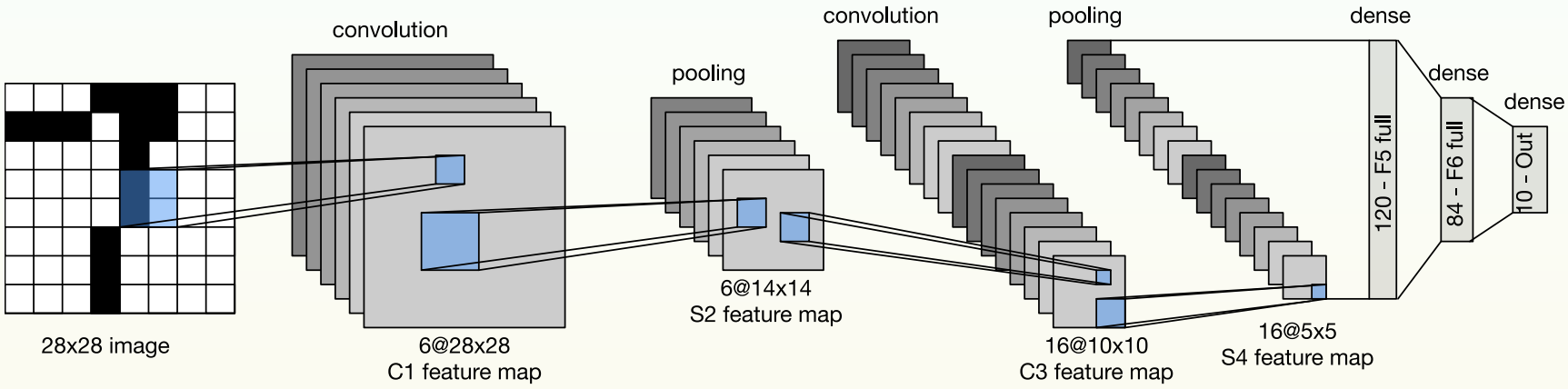






- Architecture needs to fit the desired output ✓
- Architecture needs to fit the **input data**

# Recap: CNNs



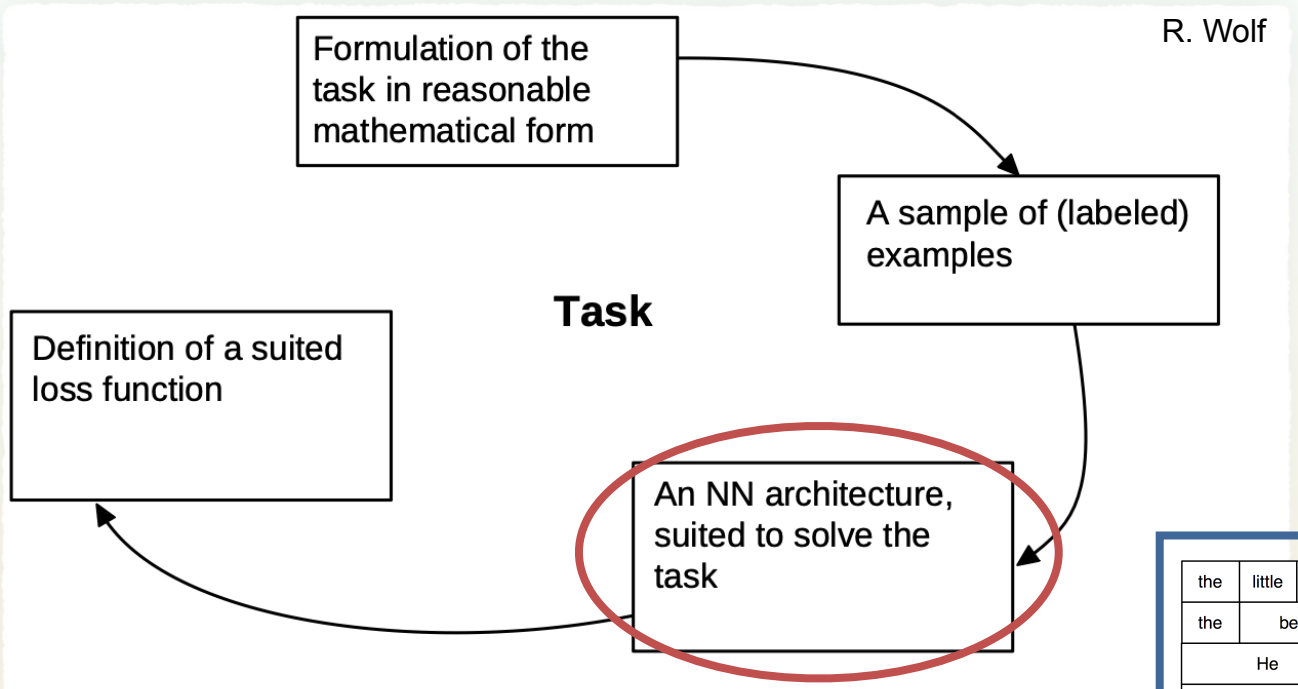

- CNNs exploit image-structured data (and abstractions of it)
  - Translations

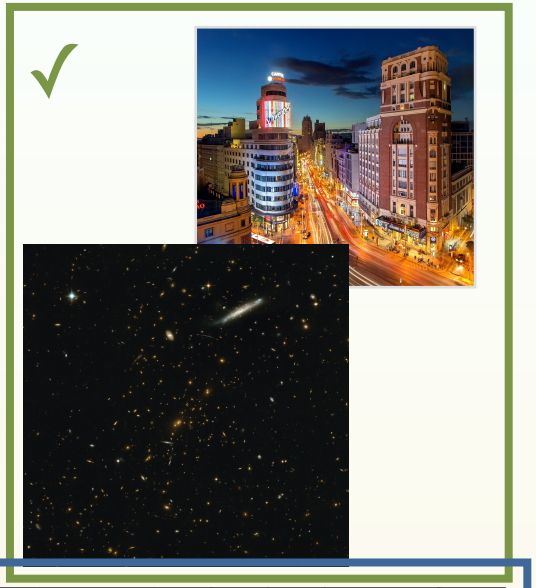
$$y_{j\alpha} = \theta\left(\sum_{\beta}^{N_F}\sum_{i}^{N_k} \omega_{i\alpha\beta}\ x_{I(j,i)\beta} - T_\alpha\right)$$

- CNN kernels are learned filters
  - Weights strictly relative to central position
  - Commute with translation
  - Effectively exploit training sample

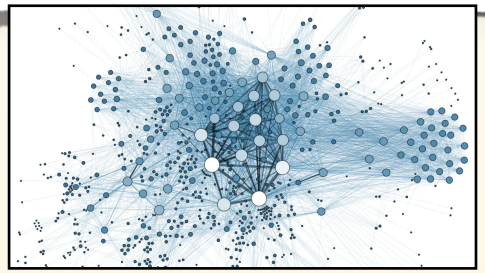
- Very powerful combined with pooling and (learned) abstraction

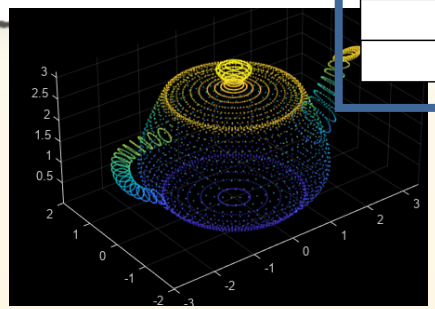
# Recap

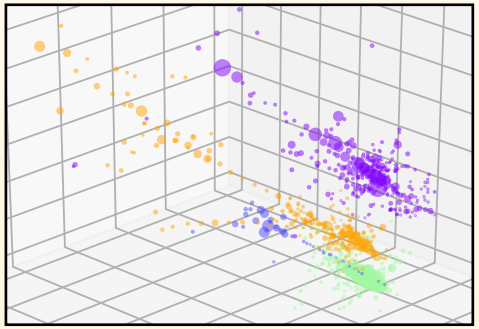


| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| the | little | bear | saw | the | fine | fat | trout | in | the | brook |
| the | bear | | saw | the | trout | | | in | it | |
| He | | | saw | it | | | | there | | |
| He | | | ran | | | | | there | | |
| He | | | ran | | | | | | | |



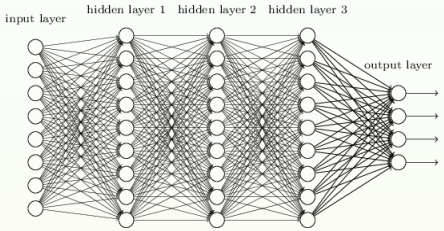


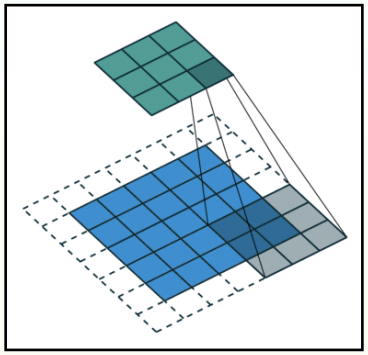


- Architecture needs to fit the **input data**
- ***Structure is the key***

# Main building blocks of architectures

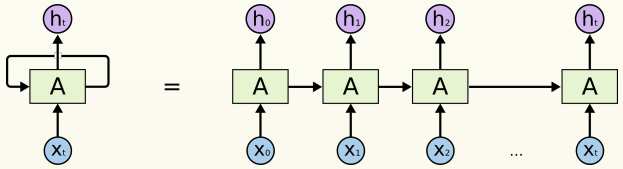
- MLP / Feed forward ✓



- CNNs ✓

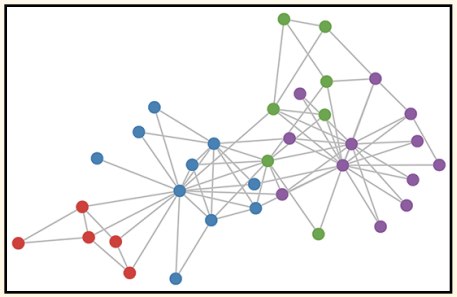
- RNNs As there was some interest in it last time, I prepared some backup slides on RNNs now




- Attention

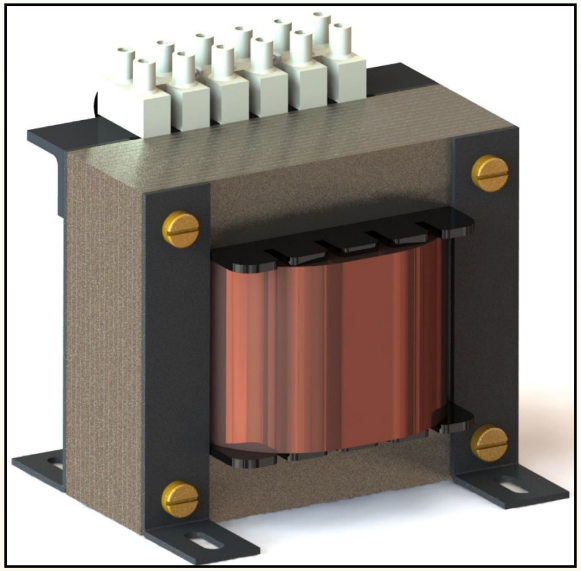
This time

- GNNs



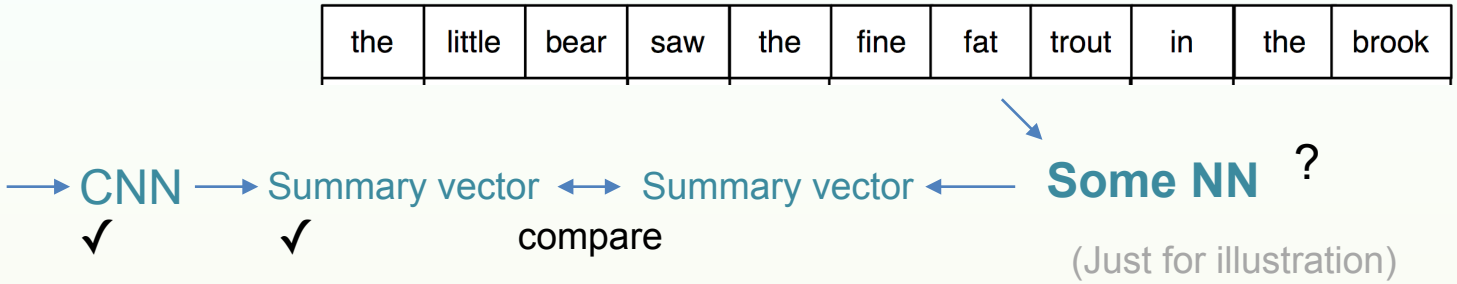
# Two perspectives


Evolution from RNN
to transformer



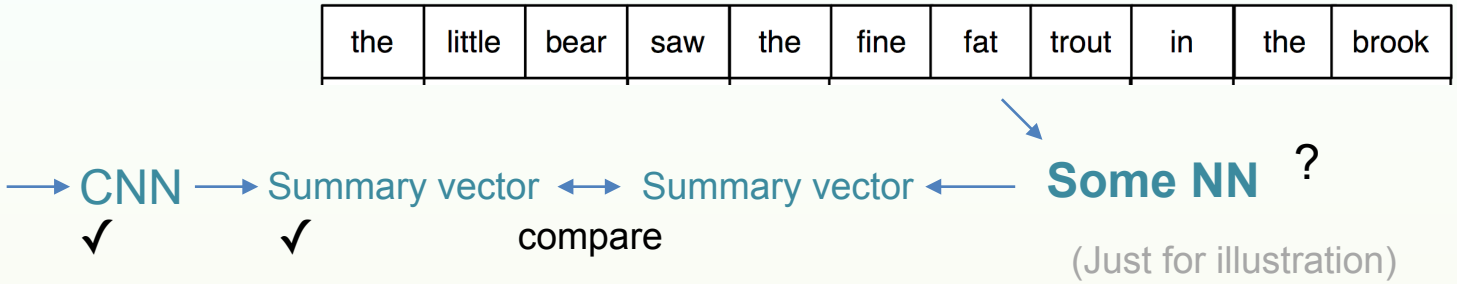
A direct approach

# Sentences and sequences




- Let's say a DNN should 'understand' the sentence
- For that, the structure is the key
- What can we say about the structure (e.g. compared to images)?

# Sentences and sequences




- Let's say a DNN should 'understand' the sentence
- For that, the structure is the key
- What can we say about the structure (e.g. compared to images)?

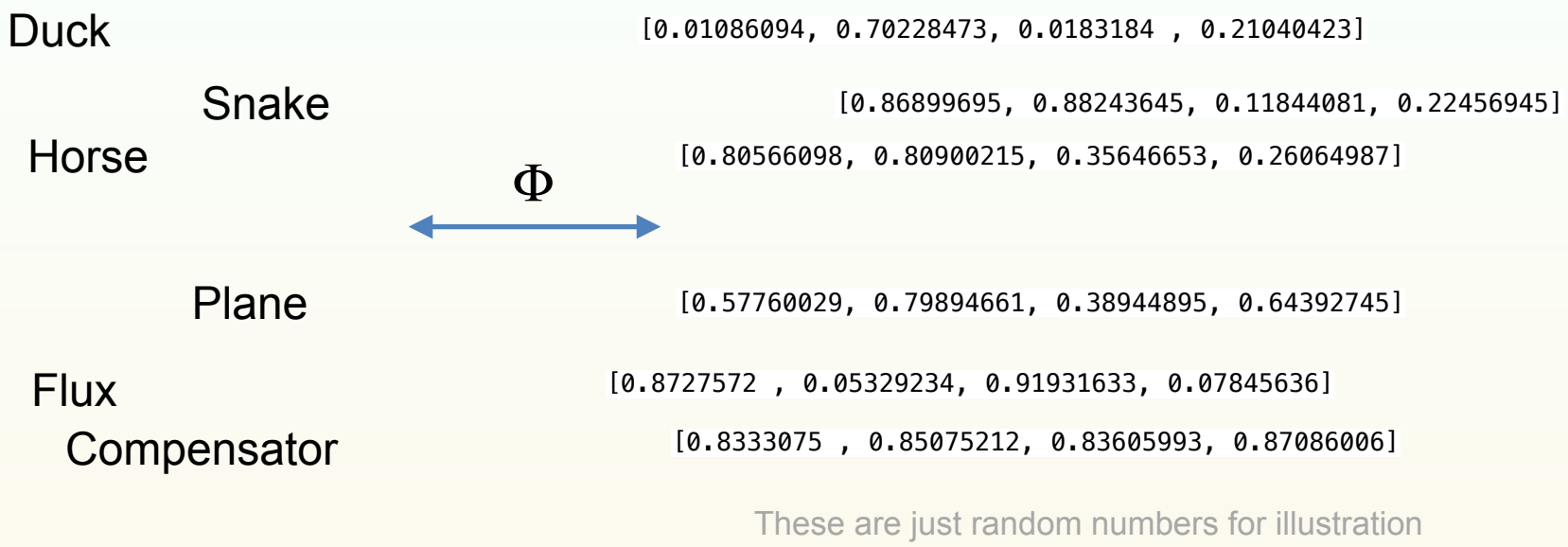
- Words, not numbers (e.g. RGB)

- Length is not fixed

- Context, relations matter (but that's also true for images)

- There is a direction

# Words to numbers: Embeddings

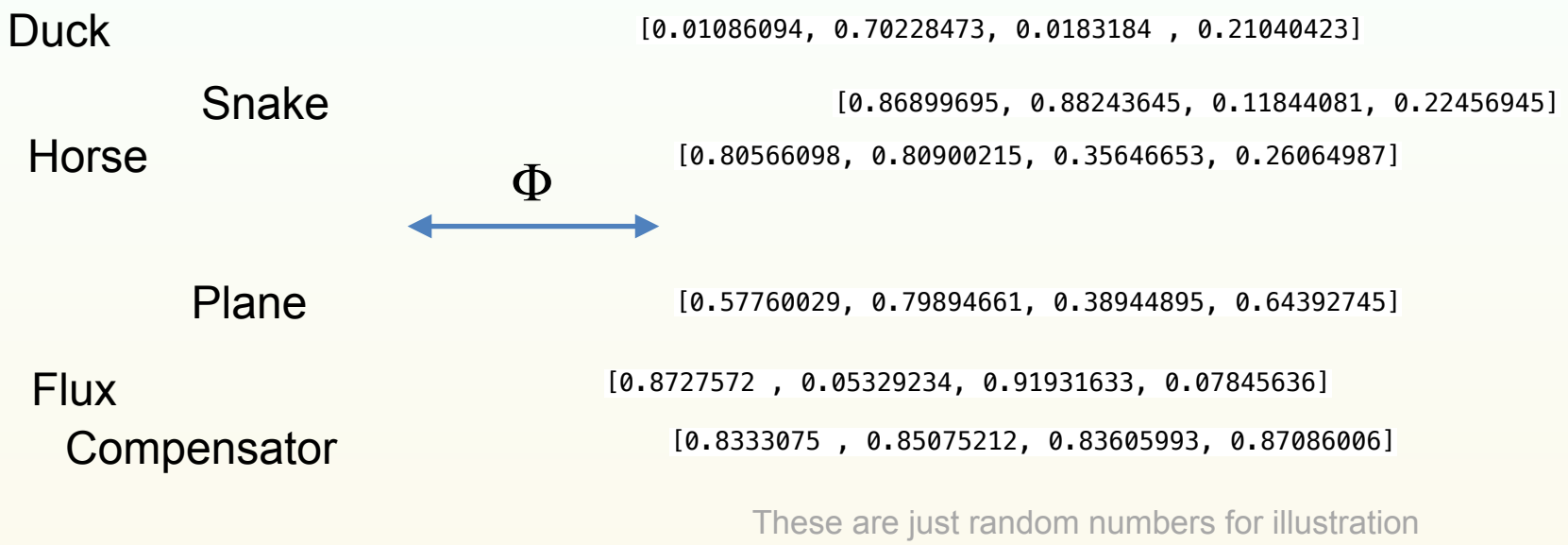



- Translate each human-readable word into an *embedding* vector

Used a lot in ML

- The easiest way to do that in a way optimal to the task?

# Words to numbers: Embeddings

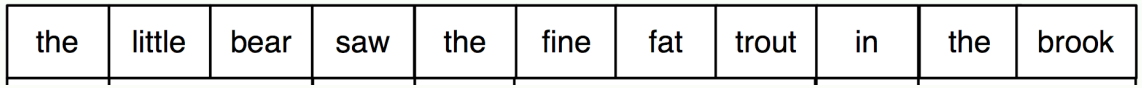



- Translate each human-readable word into an *embedding* vector

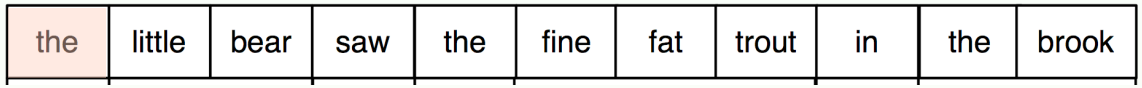
Used a lot in ML

- The easiest way to do that in a way optimal to the task?
  - Train a DNN to do it. (together with the 'main' model)
  - Words are now simple vectors ✓

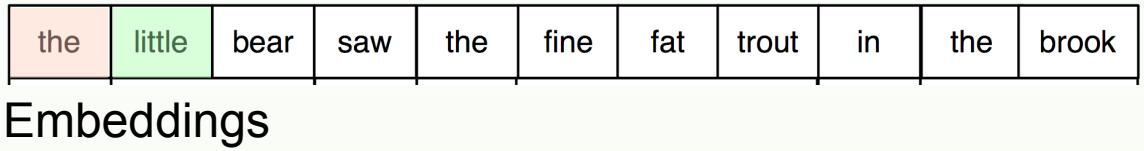
# Sentences and sequences



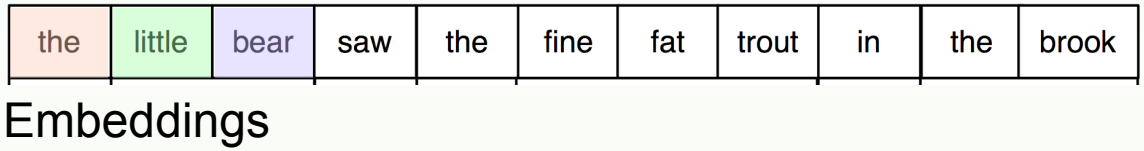
# Sentences and sequences



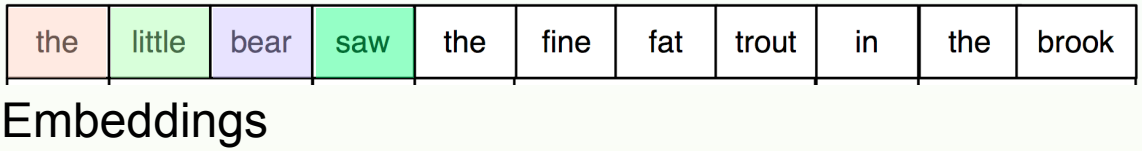
# Sentences and sequences



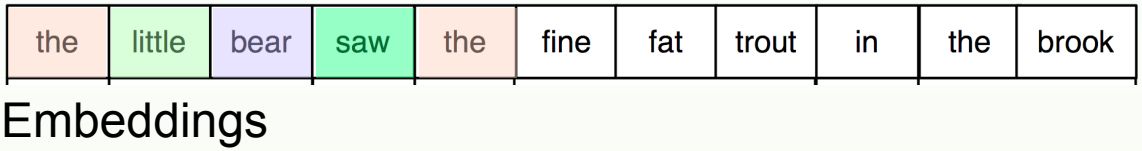
# Sentences and sequences



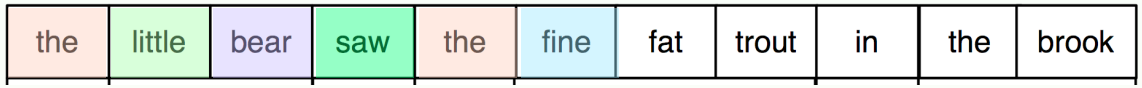
# Sentences and sequences



# Sentences and sequences

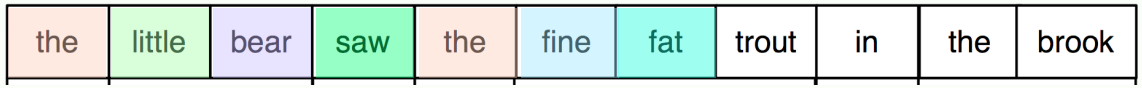


# Sentences and sequences



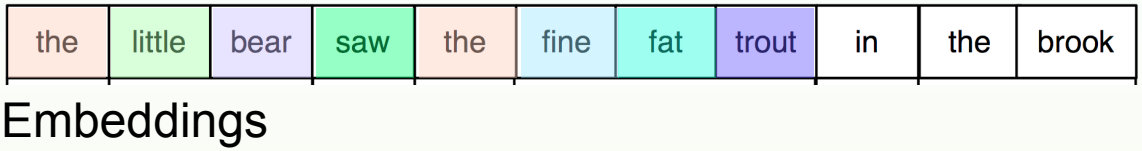

Embeddings

# Sentences and sequences

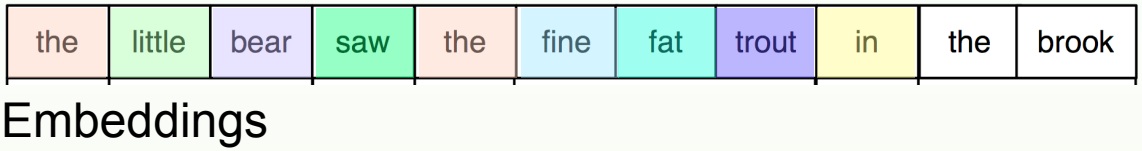



Embeddings

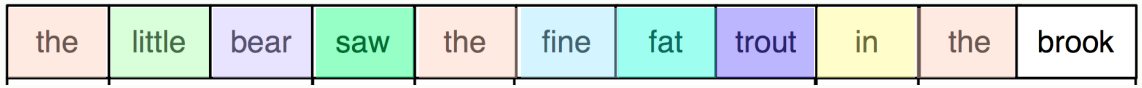
# Sentences and sequences



# Sentences and sequences

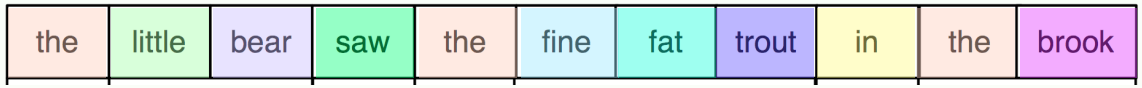


# Sentences and sequences



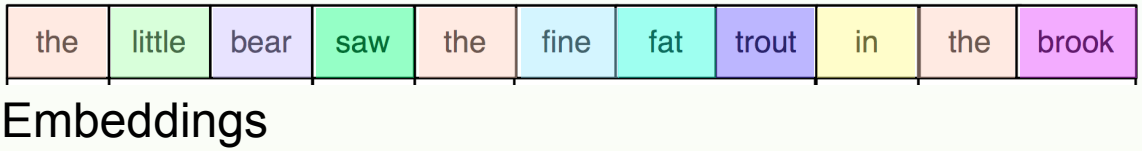

Embeddings

# Sentences and sequences




Embeddings

# Sentences and sequences



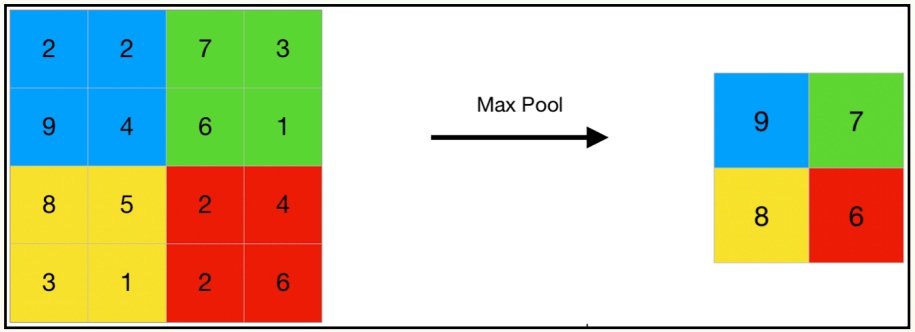

- It's different, but can we build something similar to a CNN?
  - Relations between neighbouring pixels
    → relations between neighbouring words
  - For the length, we can still zero-pad, right?
  - Let's give it a try

# Recap: Our CNN toolbox

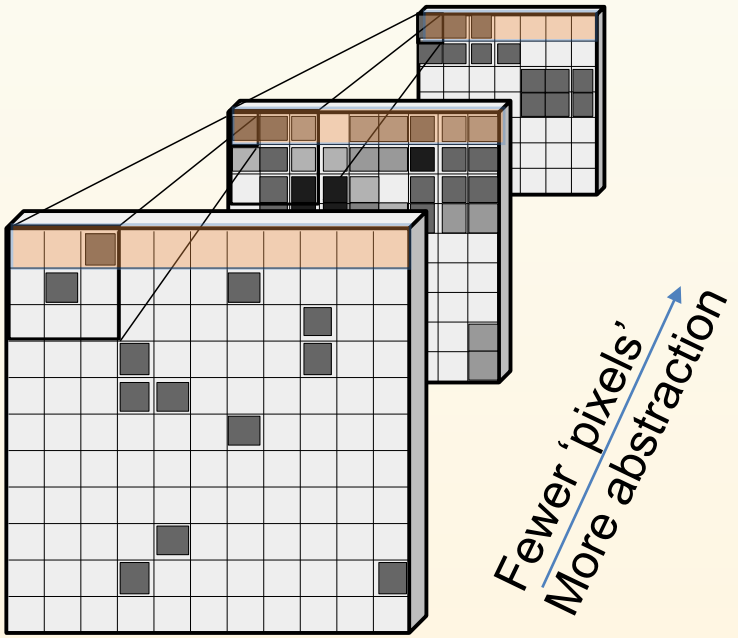
- CNN kernel
  - Learns filters

$$y_{j\alpha} = \theta\left(\sum_{\beta}^{N_F}\sum_{i}^{N_k}\omega_{i\alpha\beta}\ x_{I(j,i)\beta} - T_{\alpha}\right)$$

- Strides + Pooling
  - Build summaries

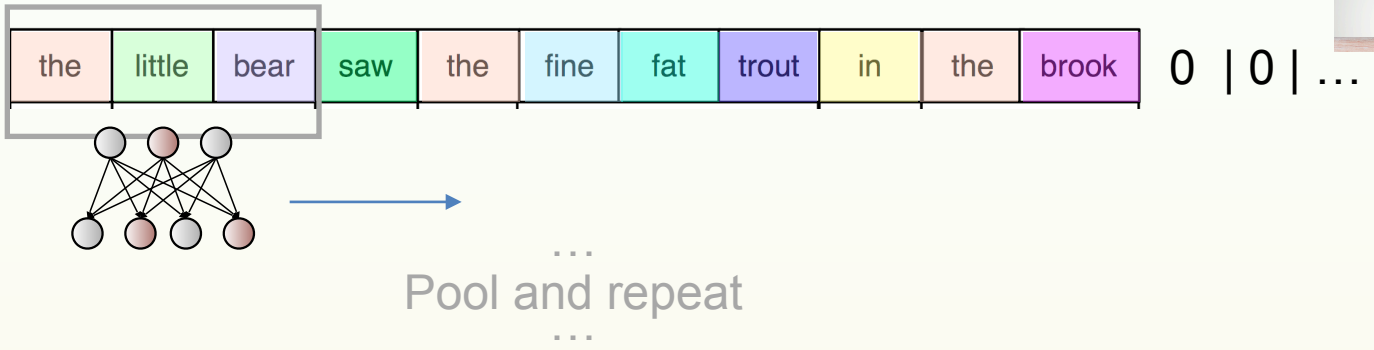



- Stack CNN layers
  - Abstraction



# A CNN-like approach

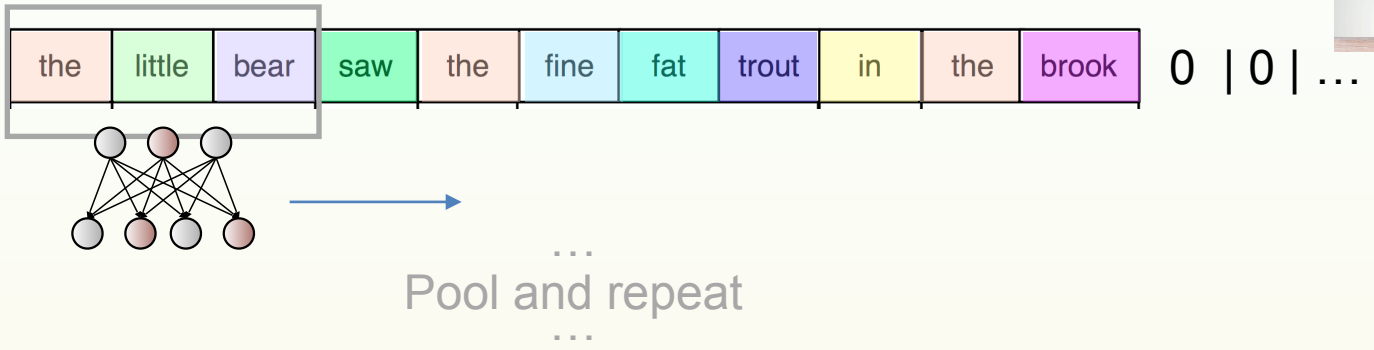
- We have a hammer, let's see if the nail fits…




- On such sentences, this could actually (more or less) work!
- But there are other sentences, e.g. in German

# A CNN-like approach

- We have a hammer, let's see if the nail fits…

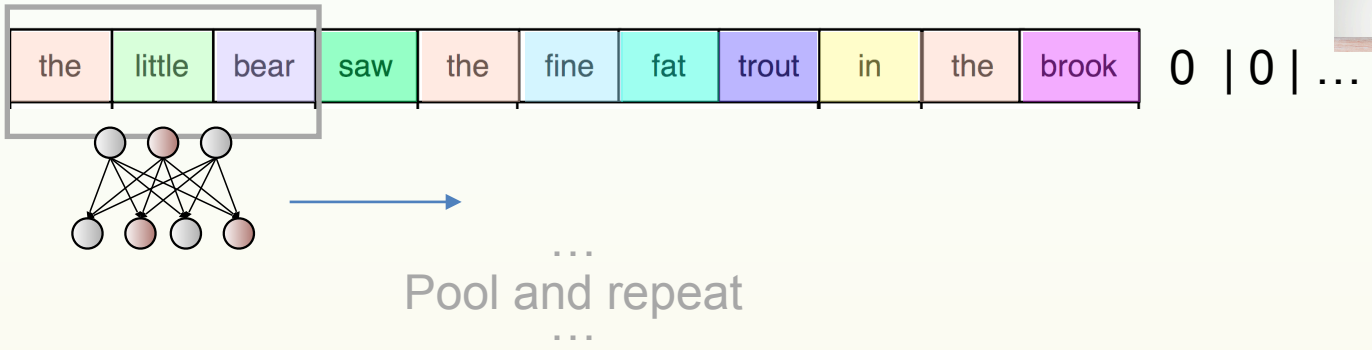



- On such sentences, this could actually (more or less) work!
- But there are other sentences, e.g. in German

**Die Versetzung und Beförderung** des deutschen Verfassungsschutzchefs Massen, der schon seit Jahren angeprangert wird und jetzt wegen einer als tendenziös bezeichneten Erklärung nach den Ereignissen von Chemnitz am 1. September nicht nur von der SPD und den Grünen, sondern auch von wichtigen Politikern der CDU kritisiert worden war und nach mehreren Krisensitzungen zwischen Merkel, Nahles und Seehofer von der Leitung des Verfassungsschutzes entbunden, wurde, **ist** von der SPD als ‚Desaster' und ‚Skandal' **bezeichnet worden.**

# A CNN-like approach

- We have a hammer, let's see if the nail fits…




- On such sentences, this could actually (more or less) work!
- But there are other sentences, e.g. in German

**Die Versetzung und Beförderung** des deutschen Verfassungsschutzchefs Massen, der schon seit Jahren angeprangert wird und jetzt wegen einer als tendenziös bezeichneten Erklärung nach den Ereignissen von Chemnitz am 1. September nicht nur von der SPD und den Grünen, sondern auch von wichtigen Politikern der CDU kritisiert worden war und nach mehreren Krisensitzungen zwischen Merkel, Nahles und Seehofer von der Leitung des Verfassungsschutzes entbunden, wurde, **ist** von der SPD als ‚Desaster' und ‚Skandal' **bezeichnet worden.**

- We would easily run into problems of an **insufficient receptive field**
- We would need to zero-pad to the **longest possible sentence**

# Extending the filter idea

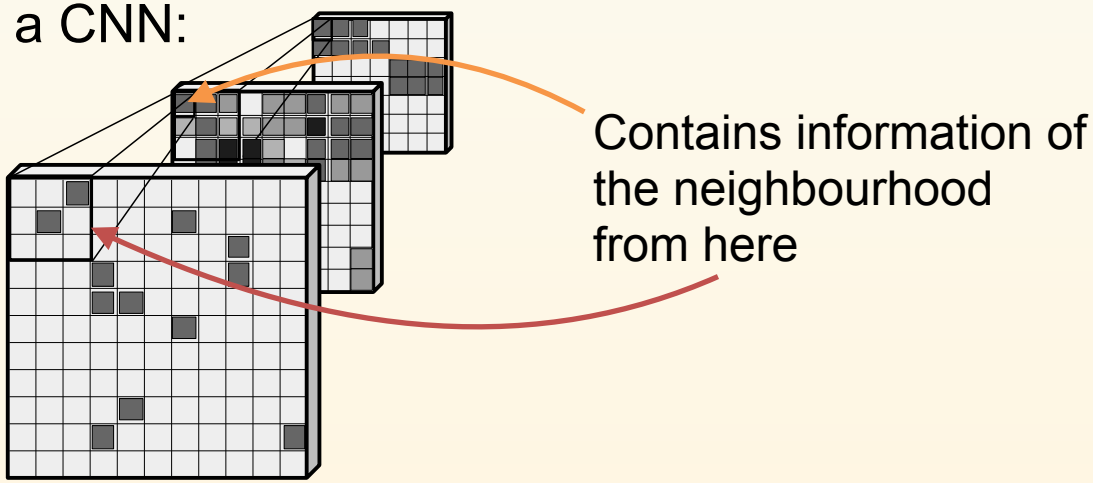
**Die Versetzung und Beförderung** des deutschen Verfassungsschutzchefs Massen, der schon seit Jahren angeprangert wird und jetzt wegen einer als tendenziös bezeichneten Erklärung nach den Ereignissen von Chemnitz am 1. September nicht nur von der SPD und den Grünen, sondern auch von wichtigen Politikern der CDU kritisiert worden war und nach mehreren Krisensitzungen zwischen Merkel, Nahles und Seehofer von der Leitung des Verfassungsschutzes entbunden, wurde, **ist** von der SPD als ‚Desaster' und ‚Skandal' **bezeichnet worden.**

- For a given word, the *best* 'neighbours' do not have a fixed position

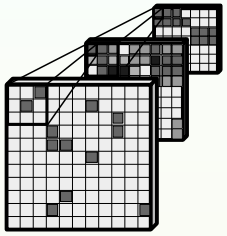
CNN

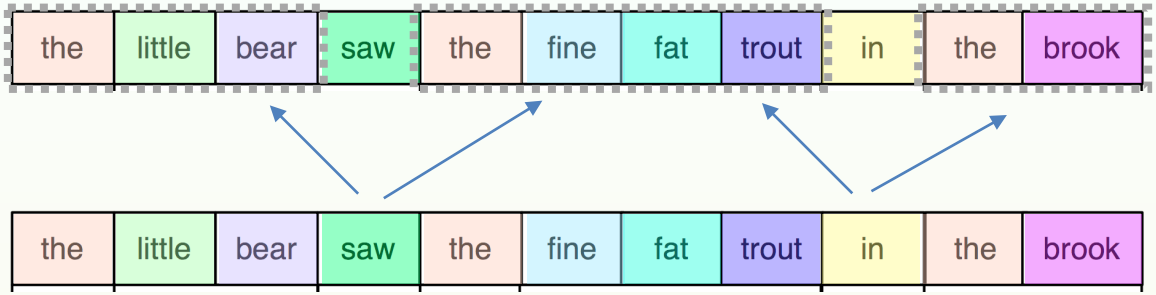

$$y_{j\alpha} = \theta\left(\sum_{\beta}^{N_F}\sum_{i}^{N_k} \omega_{i\alpha\beta}\, x_{I(j,i)\beta} - T_\alpha\right)$$

- We cannot assign a fixed weight to a fixed neighbour index as in CNNs
- But we should keep from a CNN:



# Translating this idea: redefining “neighbourhood”






Next slides are dense

- In principle, any position could be relevant
- But some more than others
- The *attention* depends on $i$ and $j$ (and in principle possibly $\beta$)

$$y_{j\alpha} = \theta\left(\sum_{\beta}^{N_F}\sum_{i}^{N_k} \omega_{i\alpha\beta}\; x_{I(j,i)\beta} - T_\alpha\right)$$

$$\omega_{i\alpha\beta} \rightarrow a(i,j,\beta)$$

Change of perspective

$$\overline{y}_{j\alpha} = \theta\left(\sum_{\beta}^{N_F}\sum_{i}^{N} a(j,i,\beta)\; x_{i\beta} - T_\alpha\right)$$

The ‘weight’ becomes a function of the words it connects

# 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

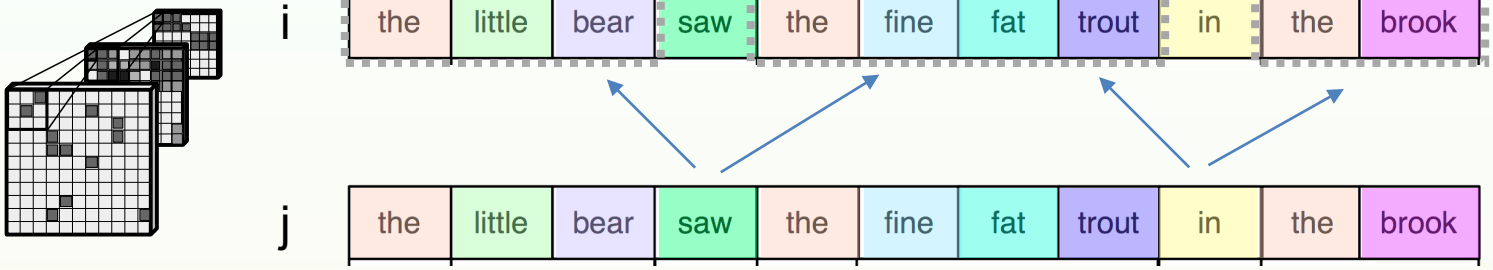
**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu

**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com

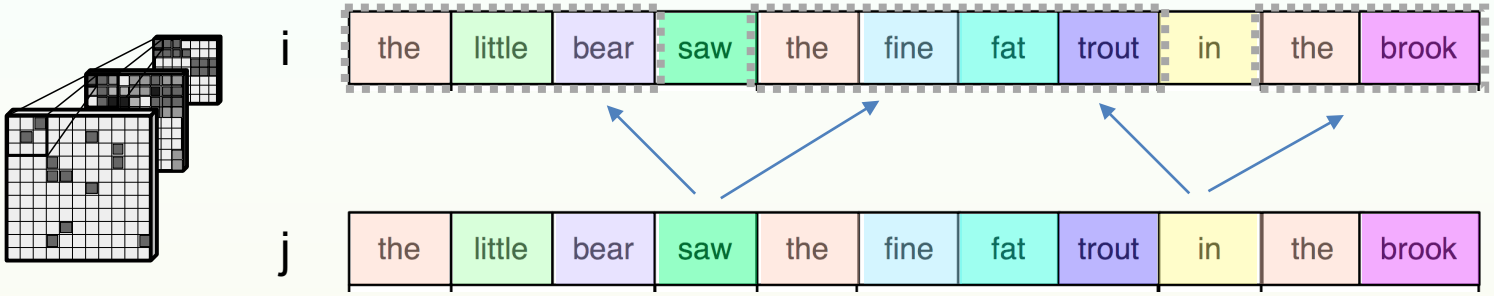
**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com

82k citations!

# Defining the attention



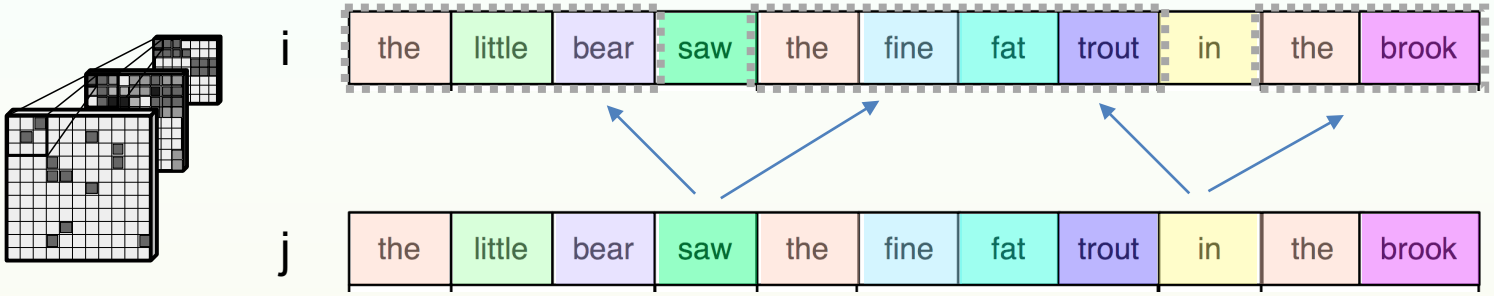
# Defining the attention




- Should depend on $\vec{x}_i$ and $\vec{x}_j$ only (vector indicates features $0, \ldots, \beta, \ldots, N_F$)

  $a_{ij} = a(\vec{x}_i, \vec{x}_j)$ now the sequence length doesn't matter anymore, it's just pair-wise comparisons

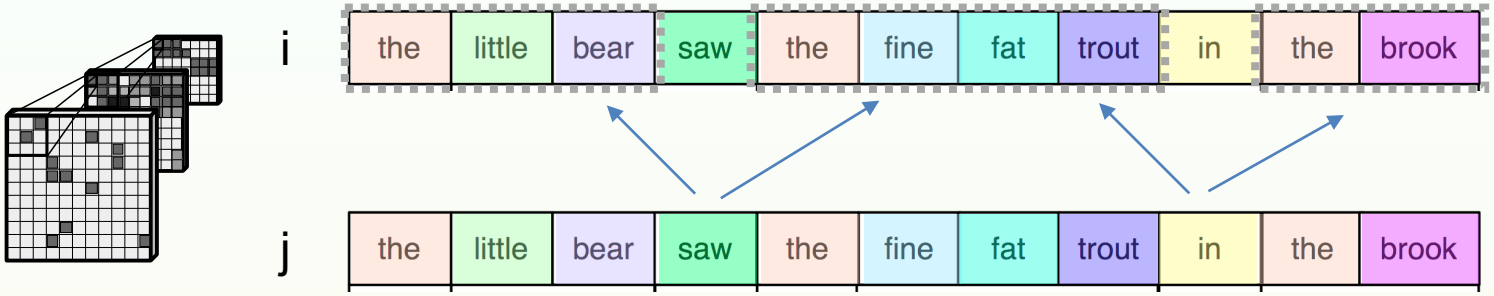
# Defining the attention




- Should depend on $\vec{x}_i$ and $\vec{x}_j$ only (vector indicates features $0, \ldots, \beta, \ldots, N_F$)
  $a_{ij} = a(\vec{x}_i, \vec{x}_j)$ now the sequence length doesn't matter anymore, it's just pair-wise comparisons

- Should be directed: $a(i,j) \neq a(j,i)$
  $a_{ij} = a(\vec{k}(\vec{x}_i), \vec{q}(\vec{x}_j))$

# Defining the attention

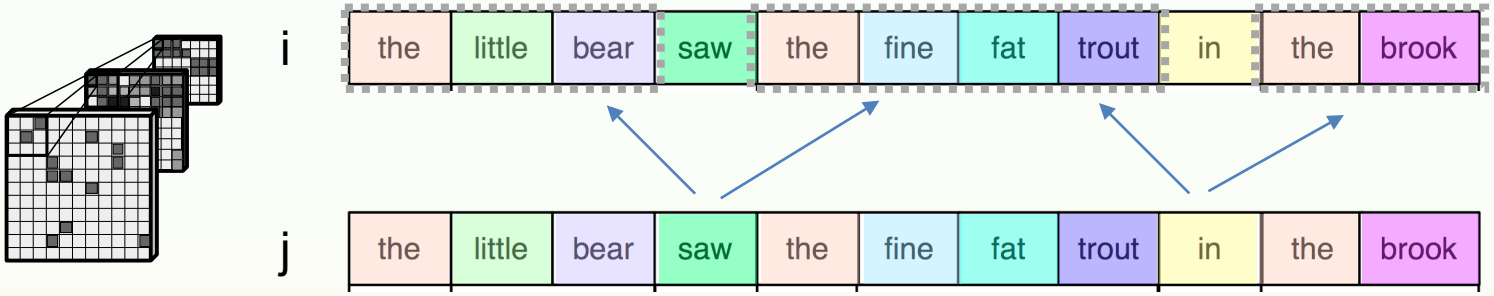



- Should depend on $\vec{x}_i$ and $\vec{x}_j$ only (vector indicates features $0, \ldots, \beta, \ldots, N_F$)
  $a_{ij} = a(\vec{x}_i, \vec{x}_j)$ now the sequence length doesn't matter anymore, it's just pair-wise comparisons

- Should be directed: $a(i,j) \neq a(j,i)$
  $a_{ij} = a(\vec{k}(\vec{x}_i), \vec{q}(\vec{x}_j))$

- Should be a scalar, simplest scalar relation between two vectors: scalar product *(alignment)*
  $a_{ij} = \ldots(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j))$

# Defining the attention



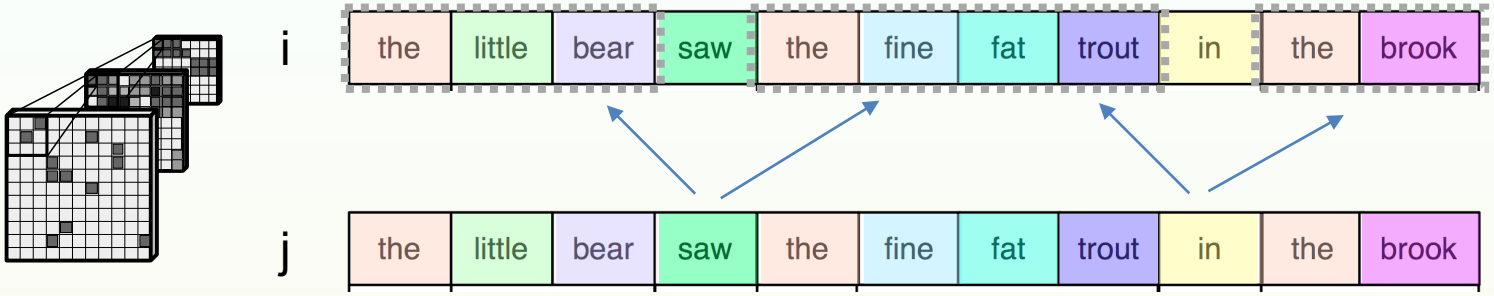

- Should depend on $\vec{x}_i$ and $\vec{x}_j$ only (vector indicates features $0, \ldots, \beta, \ldots, N_F$)
  $a_{ij} = a(\vec{x}_i, \vec{x}_j)$ now the sequence length doesn't matter anymore, it's just pair-wise comparisons

- Should be directed: $a(i,j) \neq a(j,i)$
  $a_{ij} = a(\vec{k}(\vec{x}_i), \vec{q}(\vec{x}_j))$

- Should be a scalar, simplest scalar relation between two vectors: scalar product *(alignment)*
  $a_{ij} = \ldots (\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j))$

- Should be bound between 0 and 1, such that it acts like a multiplicative filter on the inputs
  $\sum_i a_{ij} = 1$ so
  $a_{ij} = \underset{i}{\sigma}(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j))$ $\underset{i}{\sigma}$ : softmax over $i$

# Defining the attention




- Should depend on $\vec{x}_i$ and $\vec{x}_j$ only (vector indicates features $0, \ldots, \beta, \ldots, N_F$)
  $a_{ij} = a(\vec{x}_i, \vec{x}_j)$ now the sequence length doesn't matter anymore, it's just pair-wise comparisons

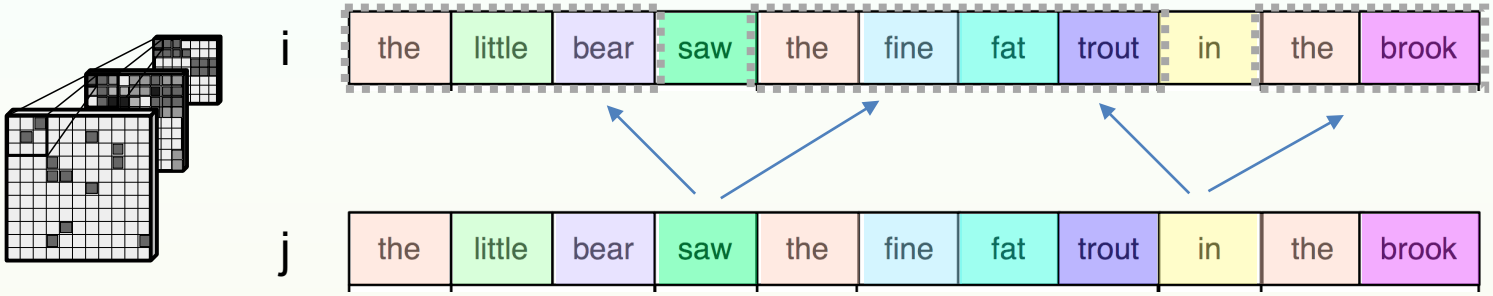
- Should be directed: $a(i,j) \neq a(j,i)$
  $a_{ij} = a(\vec{k}(\vec{x}_i), \vec{q}(\vec{x}_j))$

- Should be a scalar, simplest scalar relation between two vectors: scalar product *(alignment)*
  $a_{ij} = \ldots(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j))$

- Should be bound between 0 and 1, such that it acts like a multiplicative filter on the inputs
  $\sum_i a_{ij} = 1$ so
  $a_{ij} = \underset{i}{\sigma}(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j))$ $\underset{i}{\sigma}$ : softmax over $i$

- One last thing: the variance of $\vec{k} \cdot \vec{q}$ scales with $d_k$ , so correct for it
  $a_{ij} = \underset{i}{\sigma}\left(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j)/\sqrt{d_k}\right)$ how do we chose the best $\vec{k}$ and $\vec{q}$ ?

# Defining the attention




- Should depend on $\vec{x}_i$ and $\vec{x}_j$ only (vector indicates features $0, \ldots, \beta, \ldots, N_F$)
  $a_{ij} = a(\vec{x}_i, \vec{x}_j)$ now the sequence length doesn't matter anymore, it's just pair-wise comparisons

- Should be directed: $a(i,j) \neq a(j,i)$
  $a_{ij} = a(\vec{k}(\vec{x}_i), \vec{q}(\vec{x}_j))$

- Should be a scalar, simplest scalar relation between two vectors: scalar product *(alignment)*
  $a_{ij} = \ldots (\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j))$

- Should be bound between 0 and 1, such that it acts like a multiplicative filter on the inputs
  $\sum_i a_{ij} = 1$ so
  $a_{ij} = \underset{i}{\sigma}(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j))$ $\underset{i}{\sigma}$ : softmax over $i$

- One last thing: the variance of $\vec{k} \cdot \vec{q}$ scales with $d_k$ , so correct for it
  $a_{ij} = \underset{i}{\sigma}\left(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j)/\sqrt{d_k}\right)$ how do we chose the best $\vec{k}$ and $\vec{q}$ ?

We learn them!

# The full picture

$$a_{ij} = \underset{i}{\sigma}\left(\vec{k}(\vec{x}_i) \cdot \vec{q}(\vec{x}_j)/\sqrt{d_k}\right)$$

- Write out the attention fully (just this once)

$$a_{ij} = \underset{i}{\sigma}\left(\sum_{\alpha}^{d_k}\left(\sum_{\beta}^{N_F}\omega_{\alpha\beta}x_{i\beta}\right)\left(\sum_{\gamma}^{N_F}\tilde{\omega}_{\alpha\gamma}x_{j\gamma}\right)/\sqrt{d_k}\right)$$

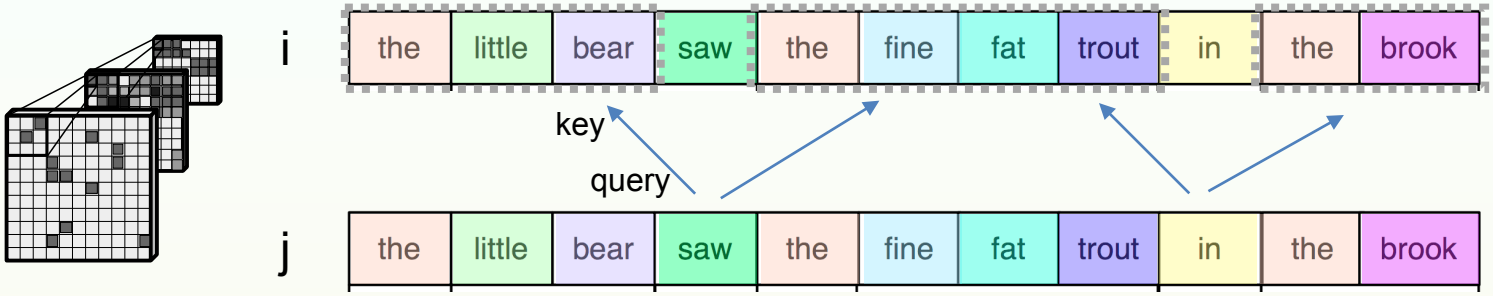
Scalar product

Learnable, creates *keys*
$\vec{k}(\vec{x}_i) = \vec{x}_i\underline{\omega}$

Learnable, creates *queries*
$\vec{q}(\vec{x}_i) = \vec{x}_j\underline{\tilde{\omega}}$

- In the end (simply) a learnable directed relation between words $i$ and $j$

# Translating this idea: redefining “neighbourhood”

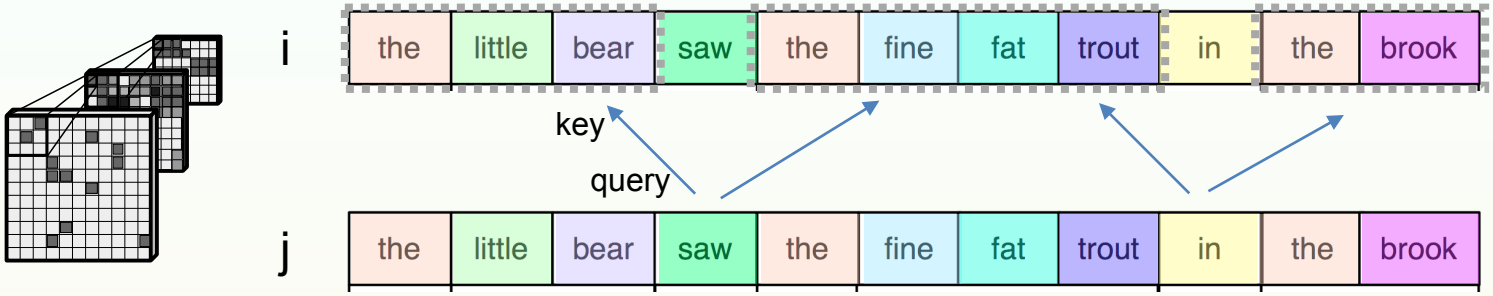



- Last step to a full *self-attention* layer: add expressivity

$$y_{j\alpha} = \cancel{\theta}\left( \sum_{\beta}^{N_F} \sum_{i}^{N} a_{ij}\, x_{i\beta} - \cancel{T_{\alpha}} \right)$$

We already have non-linearities (multiplication)

# Translating this idea: redefining “neighbourhood”




- Last step to a full *self-attention* layer: add expressivity

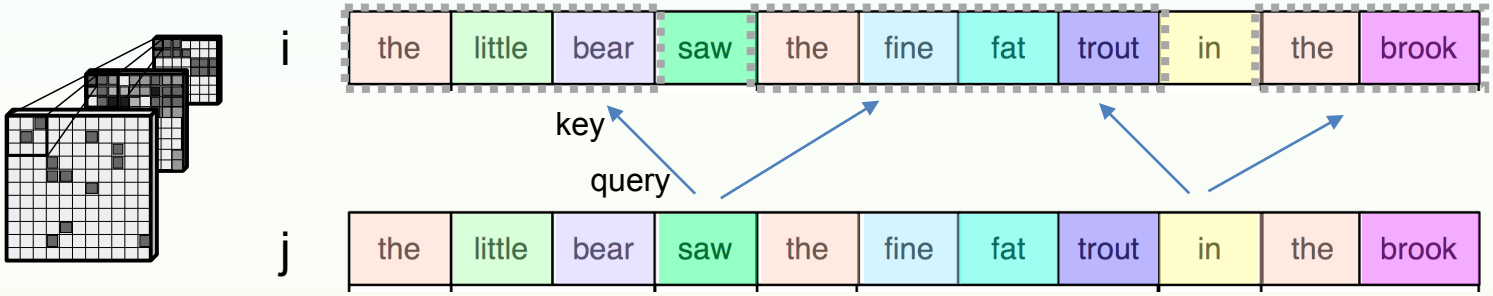
$$y_{j\alpha} = \cancel{\theta}\left( \sum_{\beta}^{N_F} \sum_{i}^{N} a_{ij}\, x_{i\beta} - \cancel{T_\alpha} \right)$$

We already have non-linearities (multiplication)

$$y_{j\alpha} = \sum_{i}^{N} a_{ij} \sum_{\beta}^{N_F} x_{i\beta}$$

sum swap

# Translating this idea: redefining “neighbourhood”




- Last step to a full *self-attention* layer: add expressivity

$$y_{j\alpha} = \cancel{\theta}\left( \sum_{\beta}^{N_F} \sum_{i}^{N} a_{ij}\, x_{i\beta} - \cancel{T_\alpha} \right)$$

We already have non-linearities (multiplication)

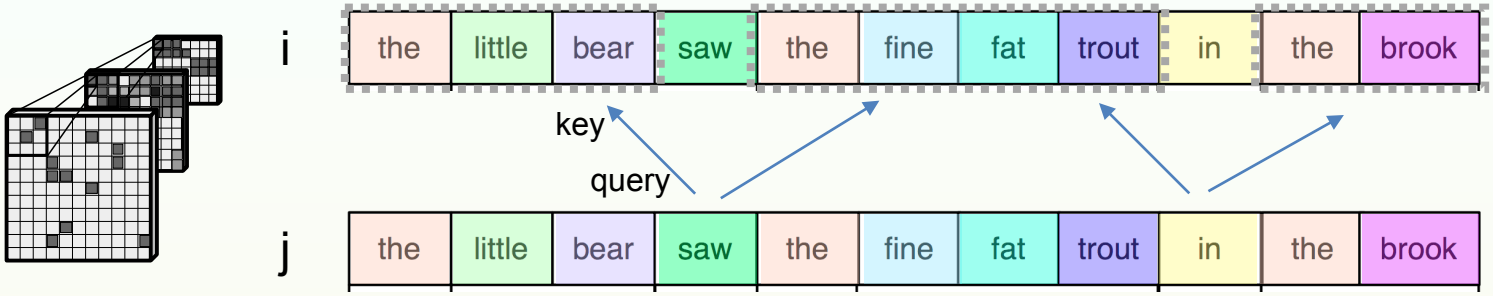
$$y_{j\alpha} = \sum_{i}^{N} a_{ij} \sum_{\beta}^{N_F} x_{i\beta}$$

sum swap

$$y_{j\alpha} := \sum_{i}^{N} a_{ij} \sum_{\beta}^{N_F} \hat{\omega}_{\alpha\beta} x_{i\beta}$$

no dangling $\alpha$, can change dimensions from $x$ to $y$

# Translating this idea: redefining “neighbourhood”




- Last step to a full *self-attention* layer: add expressivity

$$y_{j\alpha} = \cancel{\theta}\left( \sum_{\beta}^{N_F} \sum_{i}^{N} a_{ij}\ x_{i\beta} - \cancel{T_\alpha} \right)$$

We already have non-linearities (multiplication)

$$y_{j\alpha} = \sum_{i}^{N} a_{ij} \sum_{\beta}^{N_F} x_{i\beta}$$

sum swap

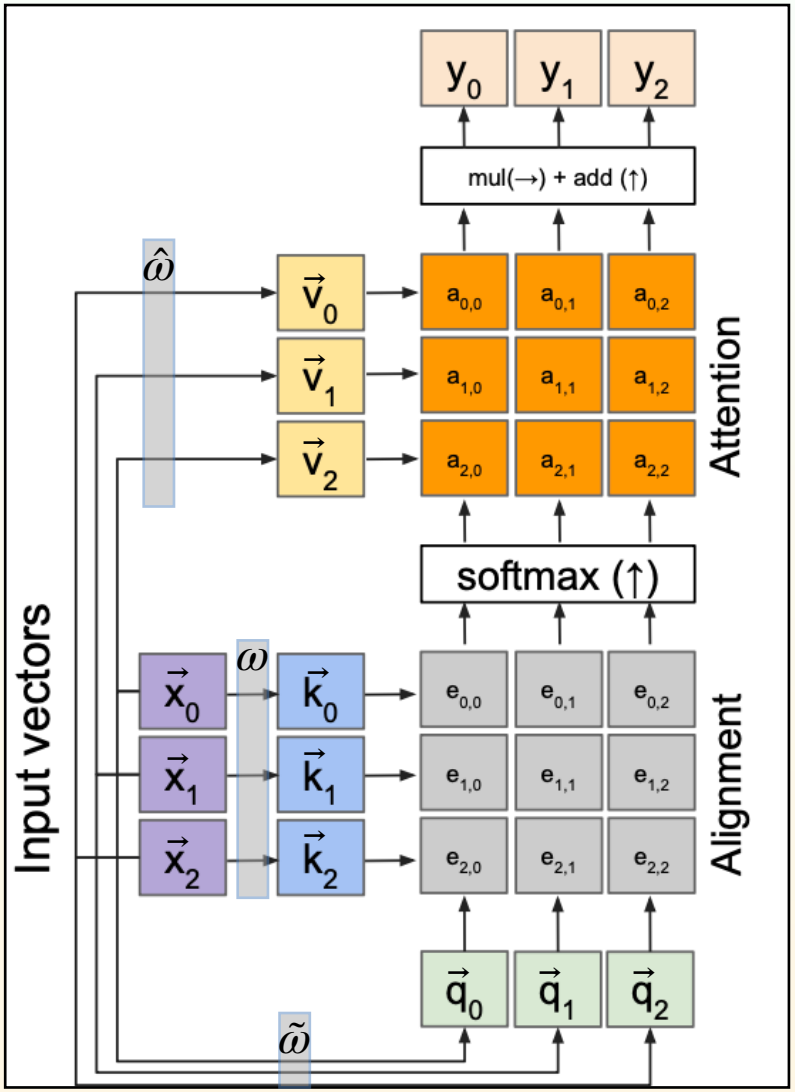
$$y_{j\alpha} := \sum_{i}^{N} a_{ij} \sum_{\beta}^{N_F} \hat{\omega}_{\alpha\beta} x_{i\beta}$$

no dangling $\alpha$, can change dimensions from $x$ to $y$

$$y_{j\alpha} = \sum_{i}^{N} a_{ij}\ v_{i\alpha}$$

v: *value vector* **this is it!**

# The final self-attention: overview




- From CNN point of view:
  instead of weights for each neighbour $i$ of pixel $j$ at a fixed position, attention weights are determined by the properties of word $j$ and word $i$

$$y_{j\alpha} = \sum_{i}^{N} a_{ij}\ v_{i\alpha}$$

$$a_{ij} = \underset{i}{\sigma}\left( \sum_{\alpha}^{d_k} \left( \sum_{\beta}^{N_F} \omega_{\alpha\beta} x_{i\beta} \right) \left( \sum_{\gamma}^{N_F} \tilde{\omega}_{\alpha\gamma} x_{j\gamma} \right) / \sqrt{d_k} \right)$$

Learnable, creates *keys*
$\vec{k}(\vec{x}_i) = \vec{x}_i \underline{\omega}$

Learnable, creates *queries*
$\vec{q}(\vec{x}_i) = \vec{x}_j \underline{\tilde{\omega}}$

Directionality

$$a_{ij} = \underset{i}{\sigma}\left(\sum_{\alpha}^{d_k}\left(\sum_{\beta}^{N_F}\omega_{\alpha\beta}x_{i\beta}\right)\left(\sum_{\gamma}^{N_F}\tilde{\omega}_{\alpha\gamma}x_{j\gamma}\right)/\sqrt{d_k}\right)$$

Attention

Non-linearity

# Time for some questions

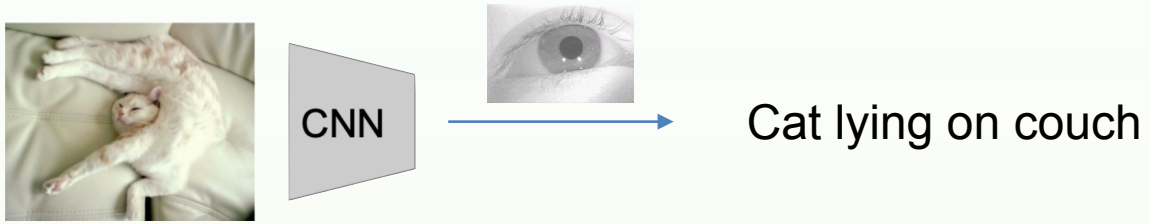
Queries

Values

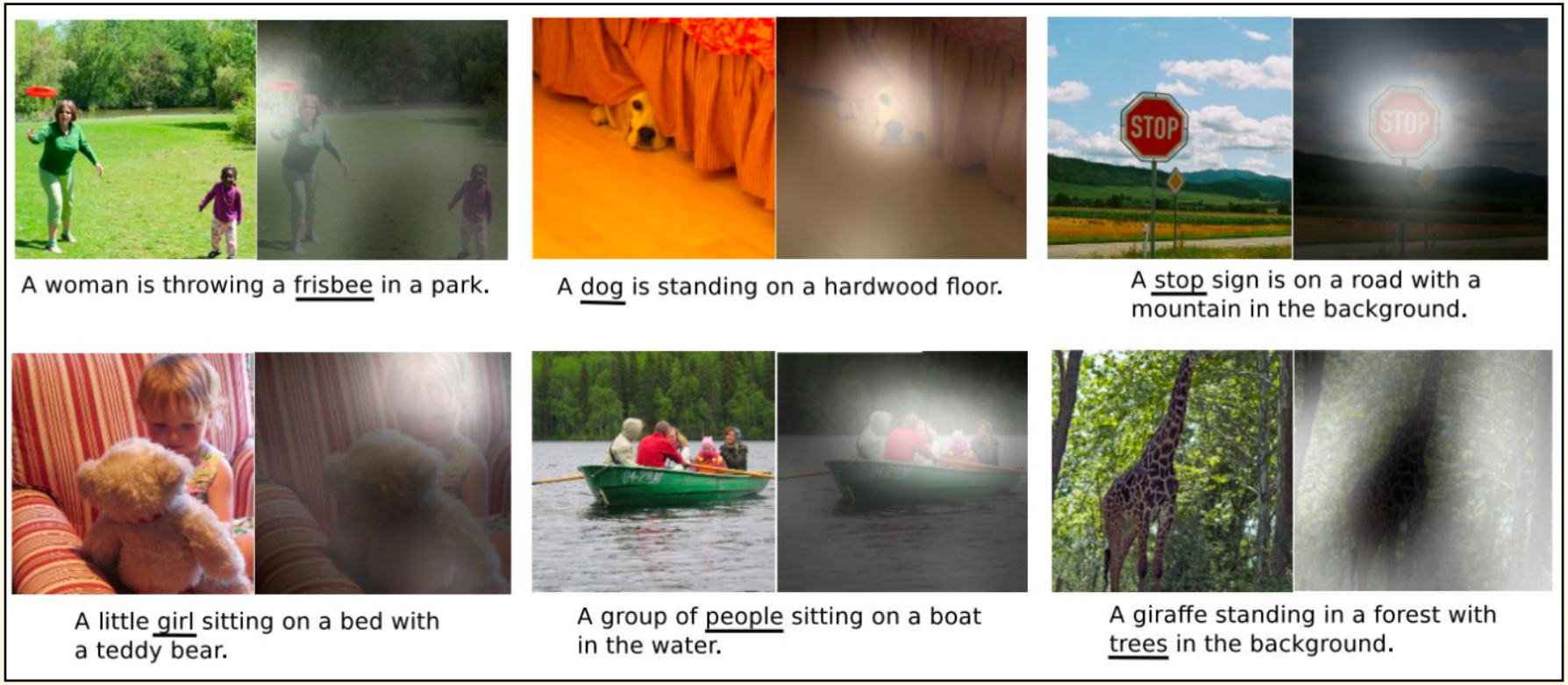
Keys

$$y_{j\alpha} = \sum_{i}^{N} a_{ij}\ v_{i\alpha}$$

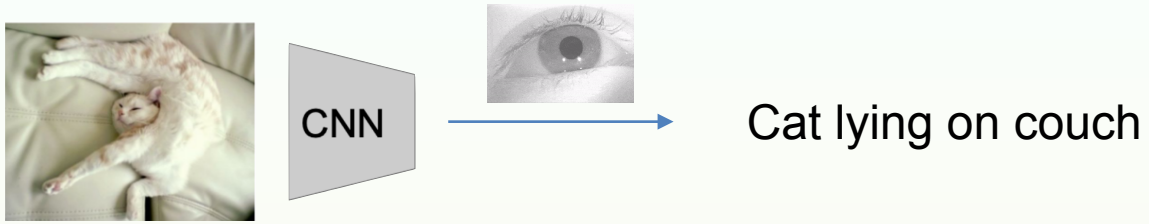
# Vision attention in action: image to text



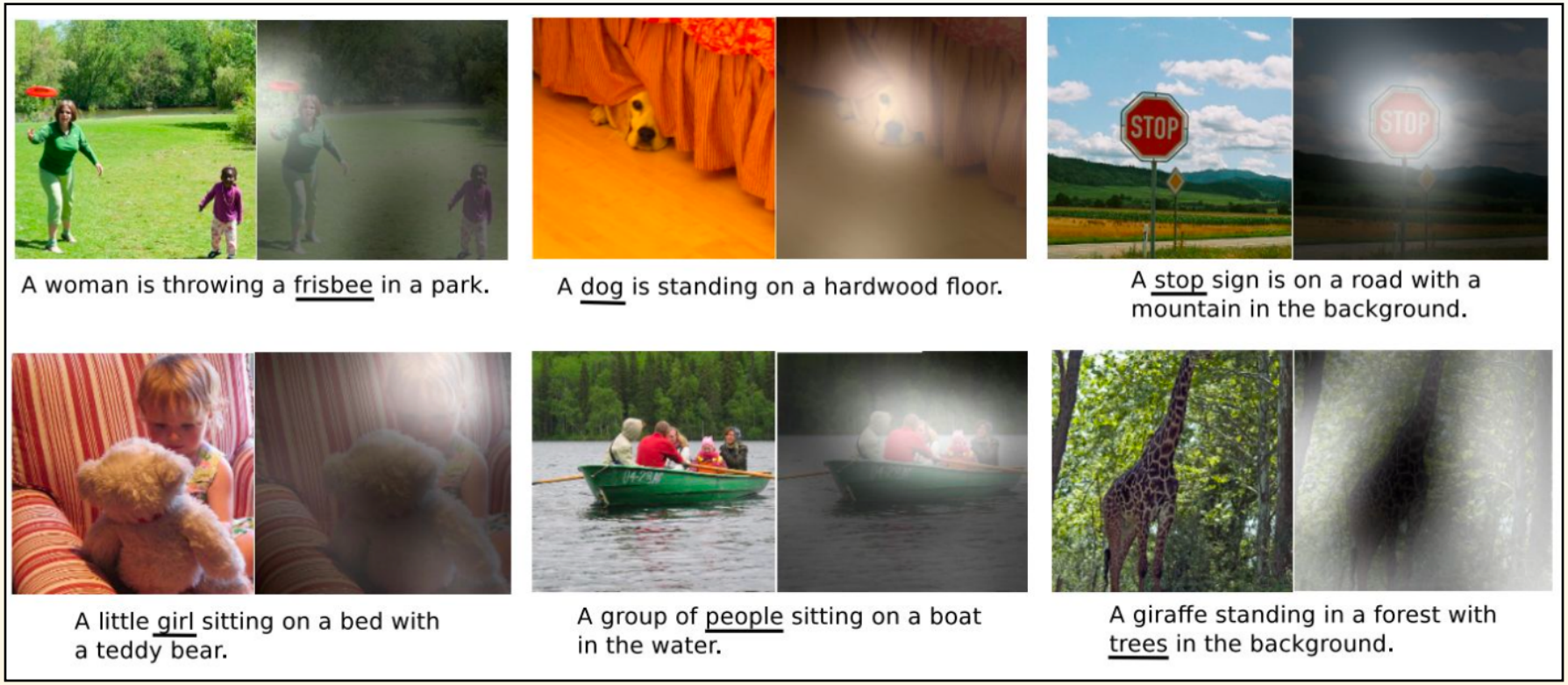

- Here, pixel-wise attention (arrow part in a few slides)



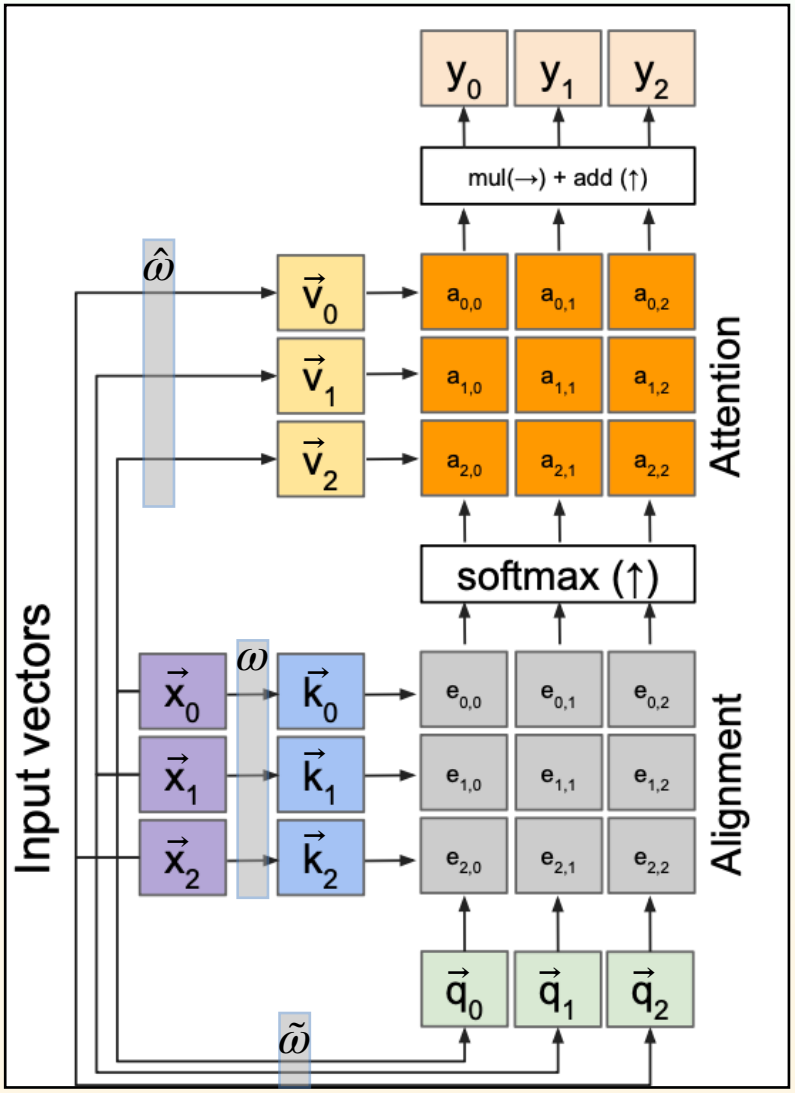
# Vision attention in action: image to text



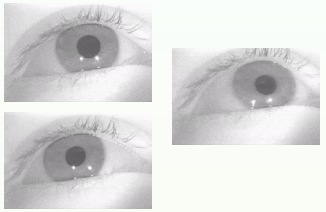

- Here, pixel-wise attention (arrow part in a few slides)

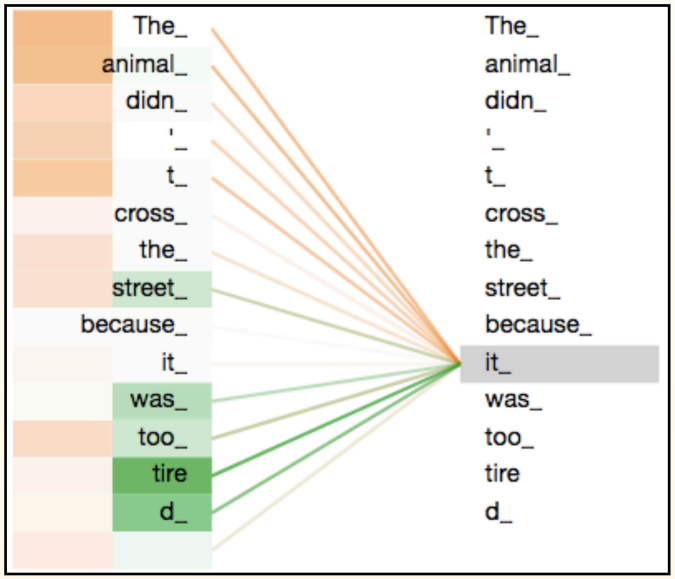


# Multi-head attention




- It might be useful to attend to other words for different kinds of information



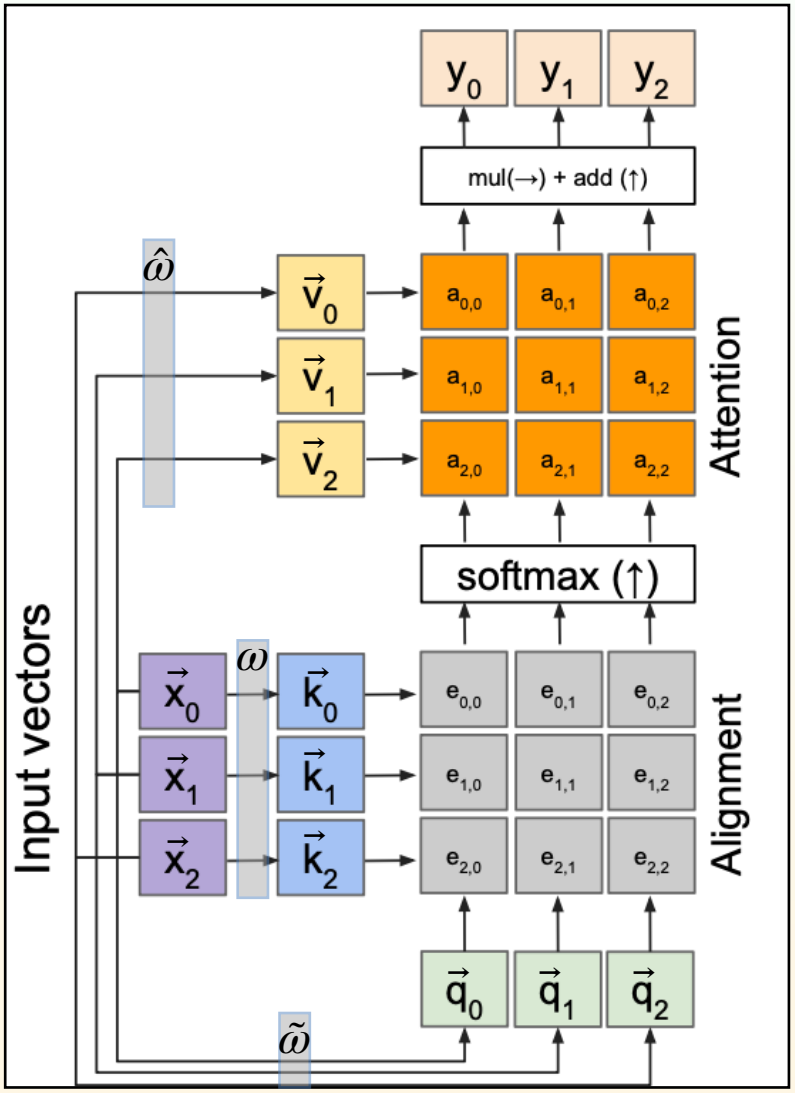



- Simply create multiple heads and concatenate their output:

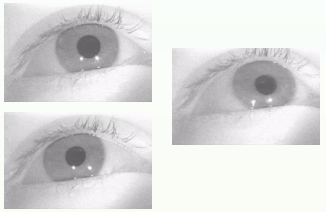
$$y_j = (y_j^1, y_j^2, \dots)$$

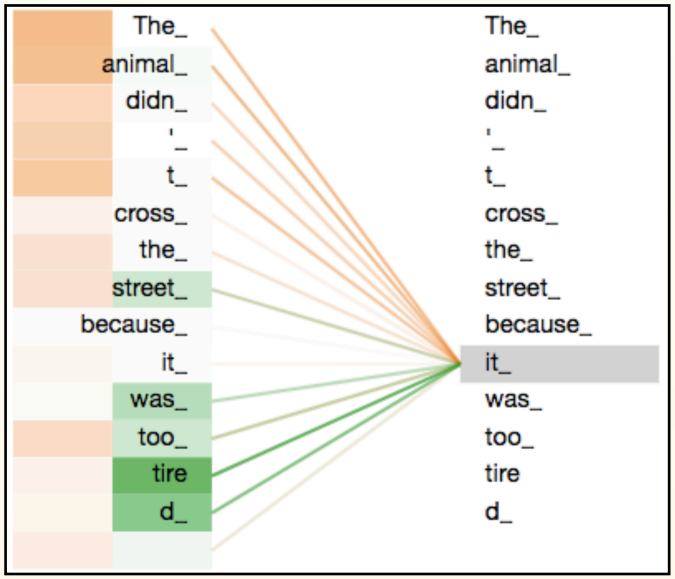
http://jalammar.github.io/illustrated-transformer/

# Multi-head attention




- It might be useful to attend to other words for different kinds of information





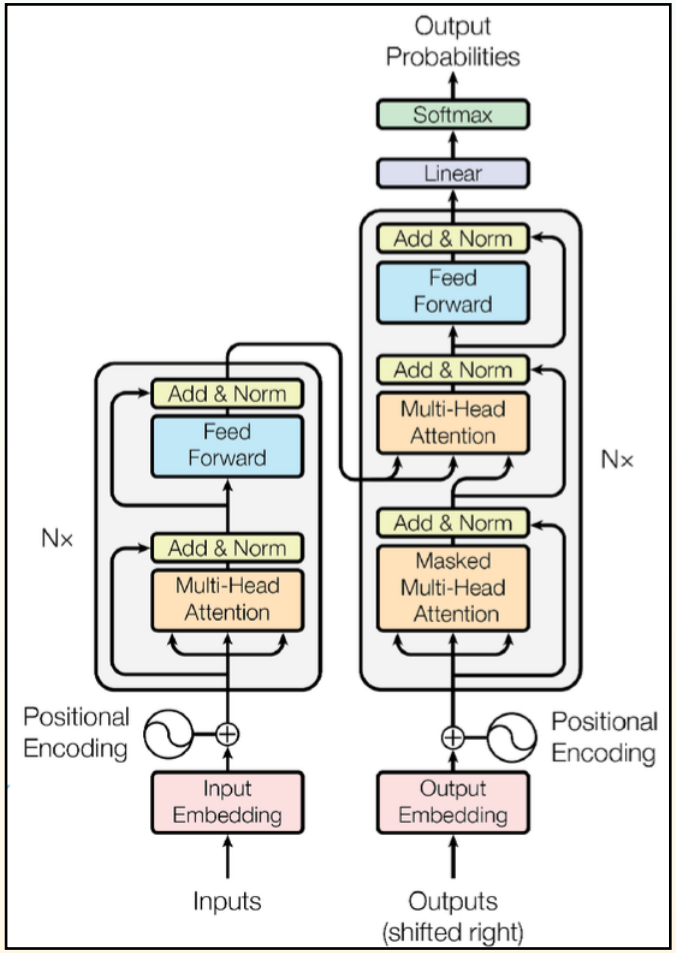

- Simply create multiple heads and concatenate their output:

$$y_j = (y_j^1, y_j^2, \dots)$$

http://jalammar.github.io/illustrated-transformer/

# The transformer model

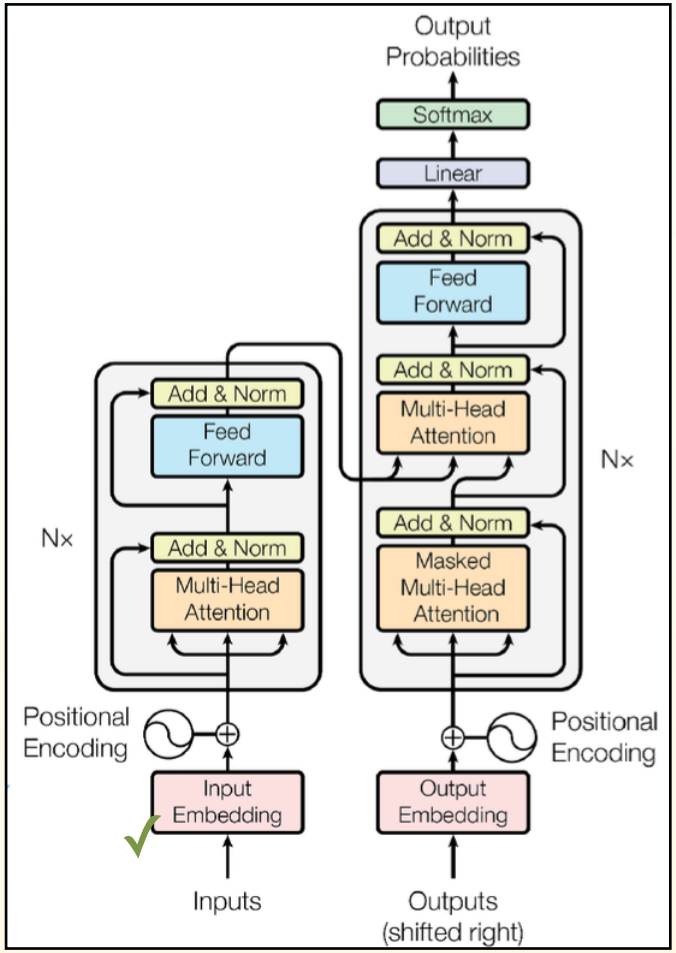
- Sequence to sequence processing (e.g. translation)
- Contains encoding and decoding block
- Will go though it step by step in the next slides




arxiv:1706.03762

# The transformer model

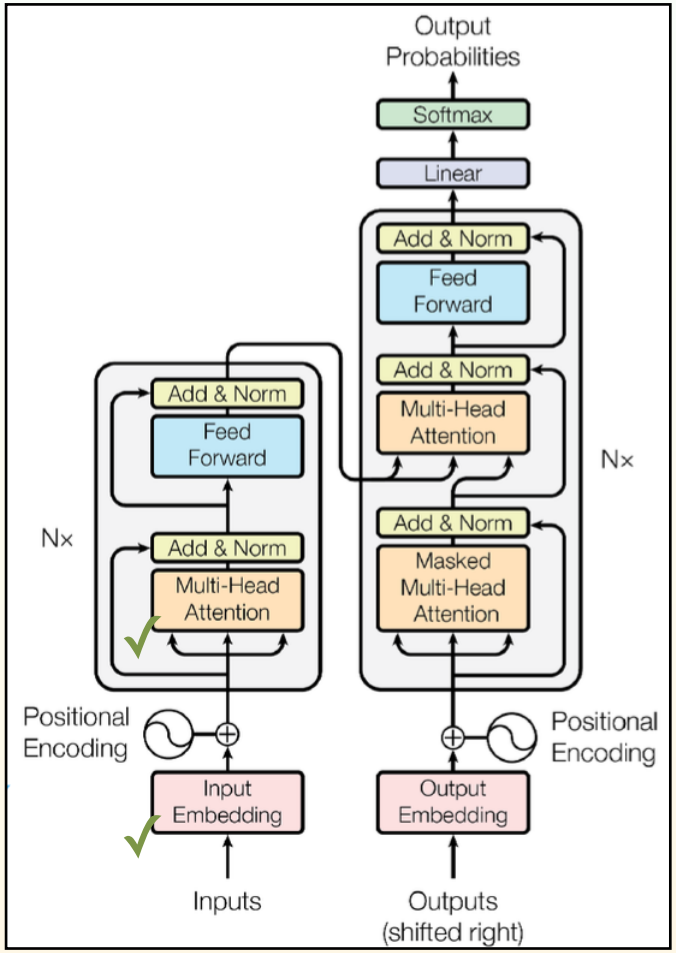
- Sequence to sequence processing (e.g. translation)
- Contains encoding and decoding block
- Will go though it step by step in the next slides




arxiv:1706.03762

# The transformer model

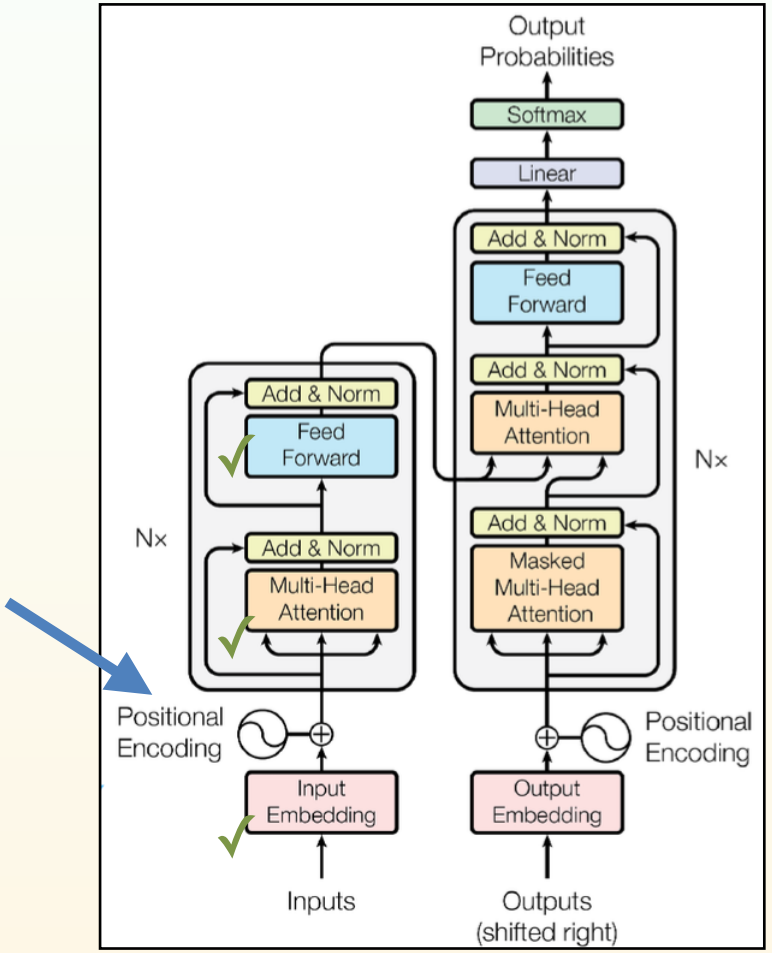
- Sequence to sequence processing (e.g. translation)
- Contains encoding and decoding block
- Will go though it step by step in the next slides




arxiv:1706.03762

# The transformer model

- Sequence to sequence processing (e.g. translation)
- Contains encoding and decoding block
- Will go though it step by step in the next slides



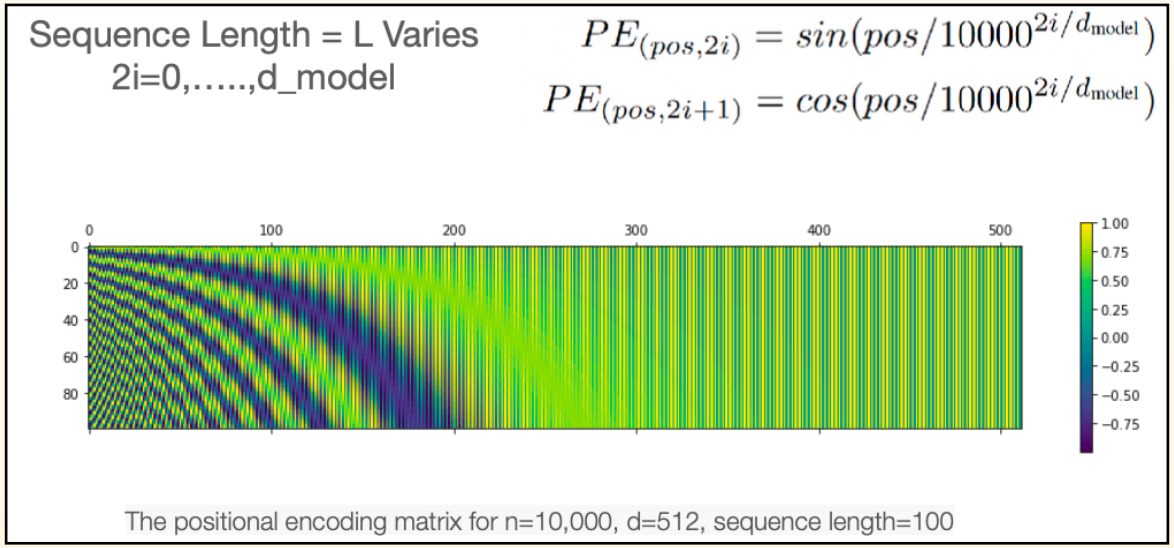

arxiv:1706.03762

# Positional encoding (brief, FYI)

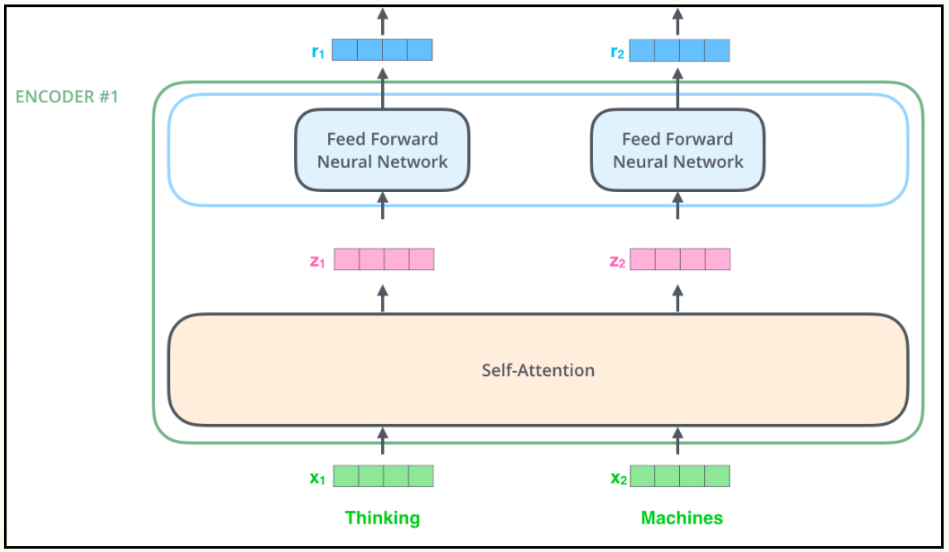
- So far, the attention mechanism is **ordering invariant** (by construction)
- This can be very useful for different physics tasks
- For text, the position matters

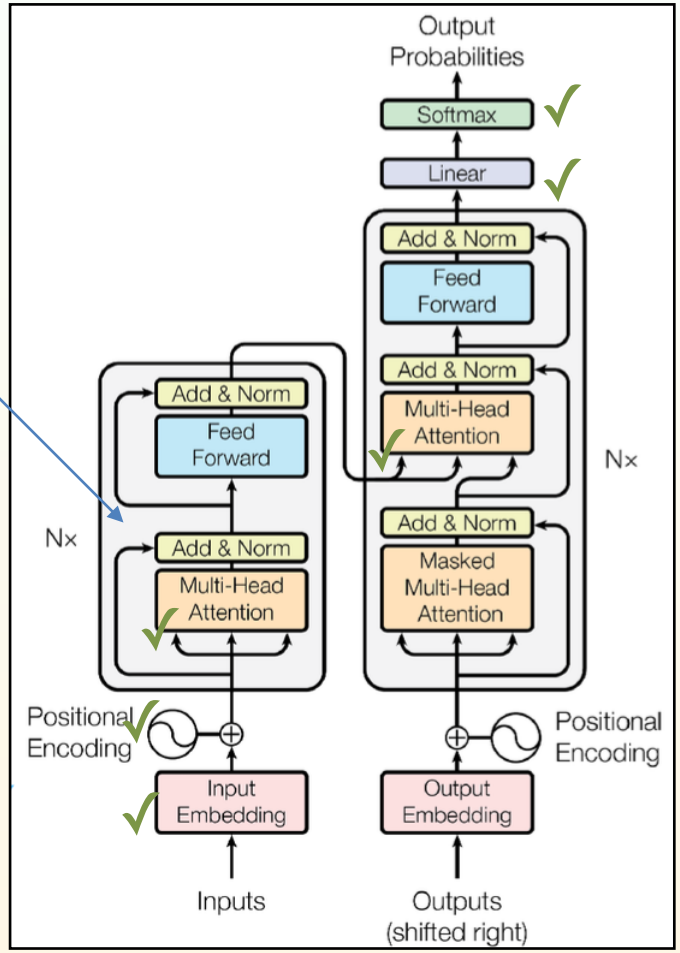
# Positional encoding (brief, FYI)

- So far, the attention mechanism is **ordering invariant** (by construction)
- This can be very useful for different physics tasks
- For text, the position matters
- Simple solution: add a 'position feature'
  $\vec{x}_i \rightarrow (x_{i0}, \ldots, x_{i\,N_F}, x_{i\,(p0)}, \ldots)$
- To keep them between -1 and 1, often a set of orthogonal sin or cos functions used



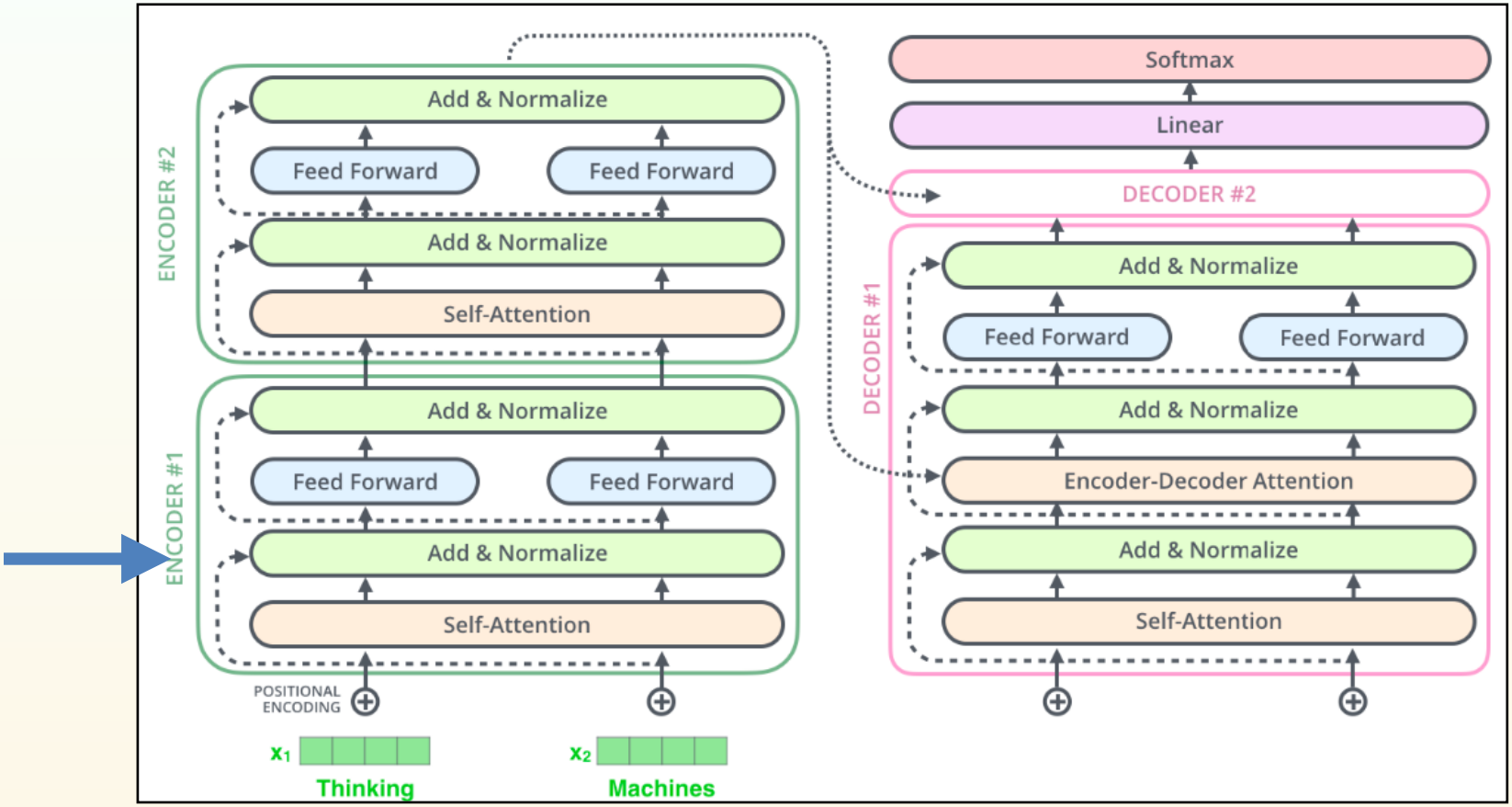
# Feed forward



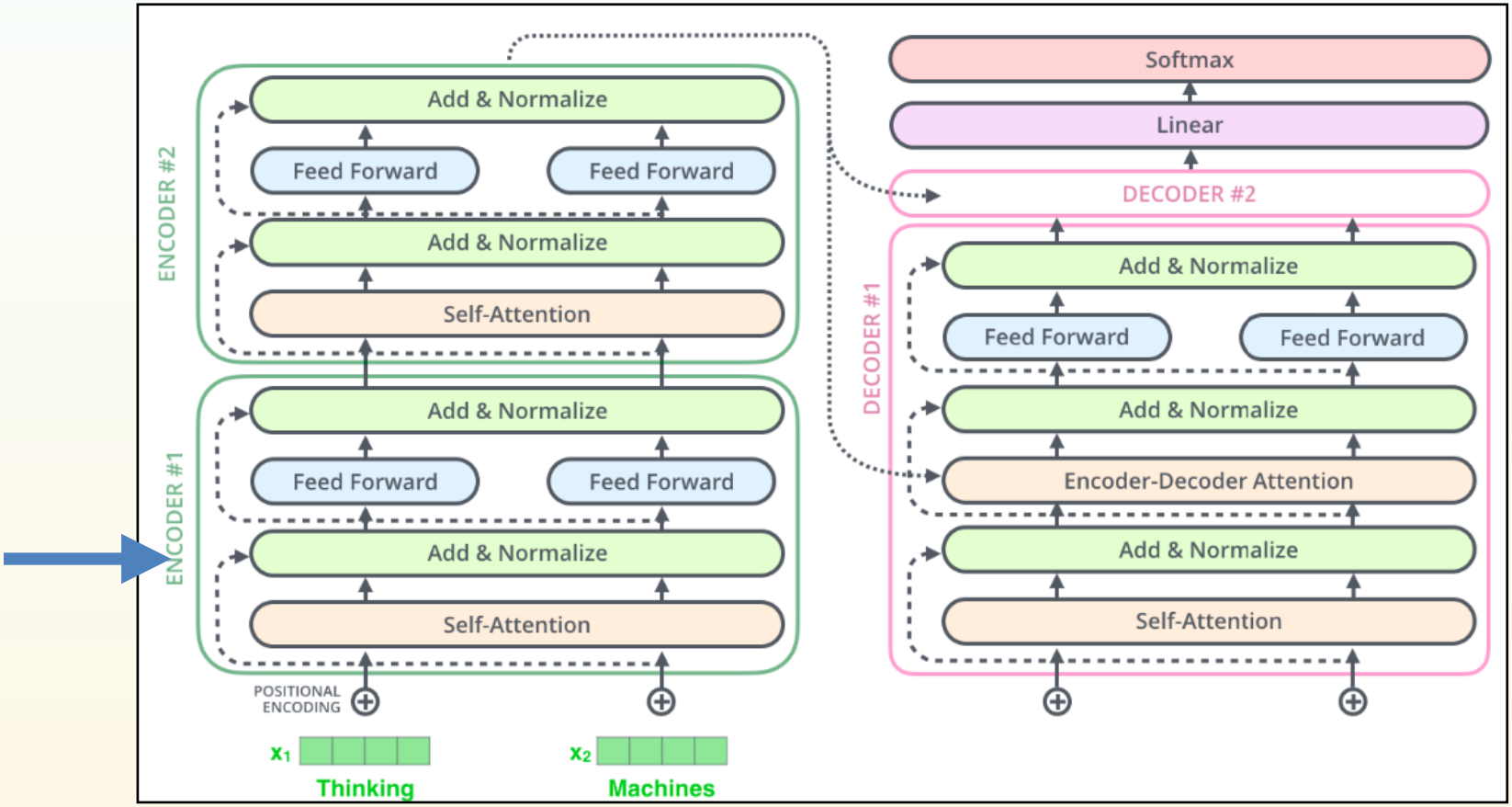

- The FF network is an MLP with shared weights for each ‘word’ (same as a 1D CNN)

# More details: skip / residual connections




- Each attention layer adds to the features
- Ensures non-vanishing gradients. Why?
  - **Useful in many contexts**

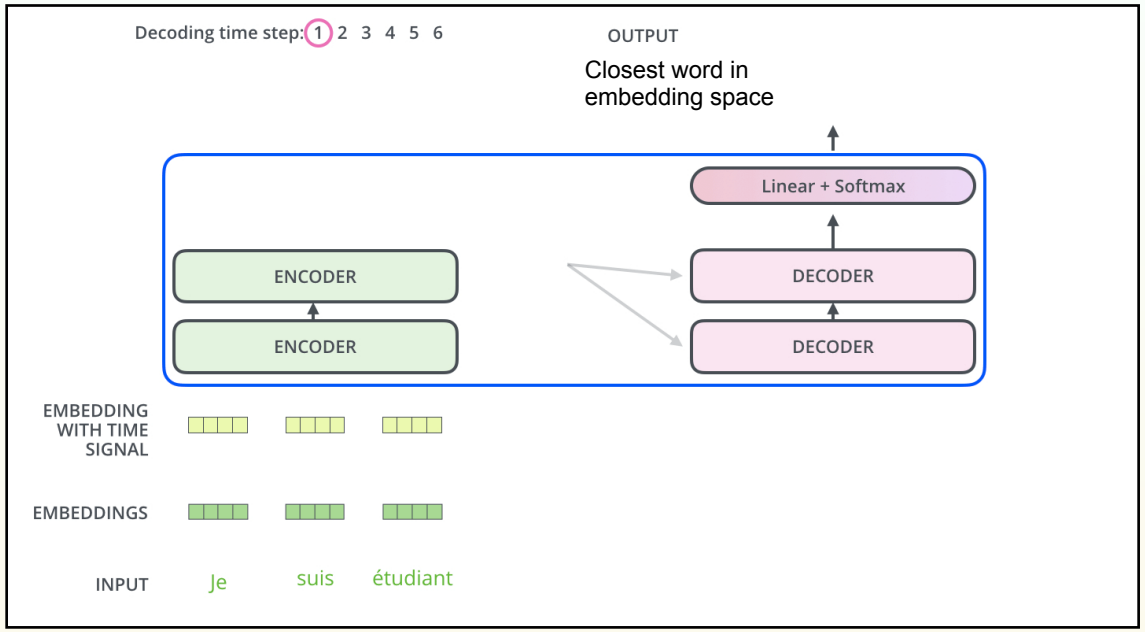
# More details: skip / residual connections

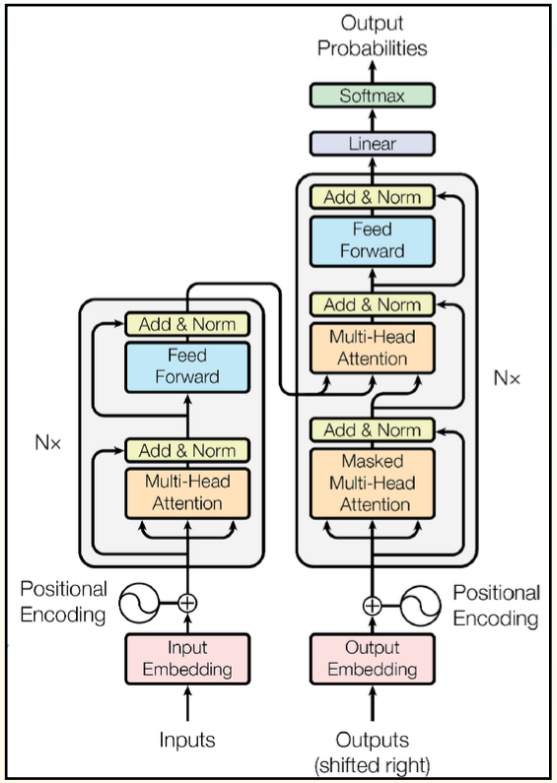



- Each attention layer adds to the features
- Ensures non-vanishing gradients. Why?
  - **Useful in many contexts**

Because there is always a short connection back to the inputs in the backward pass

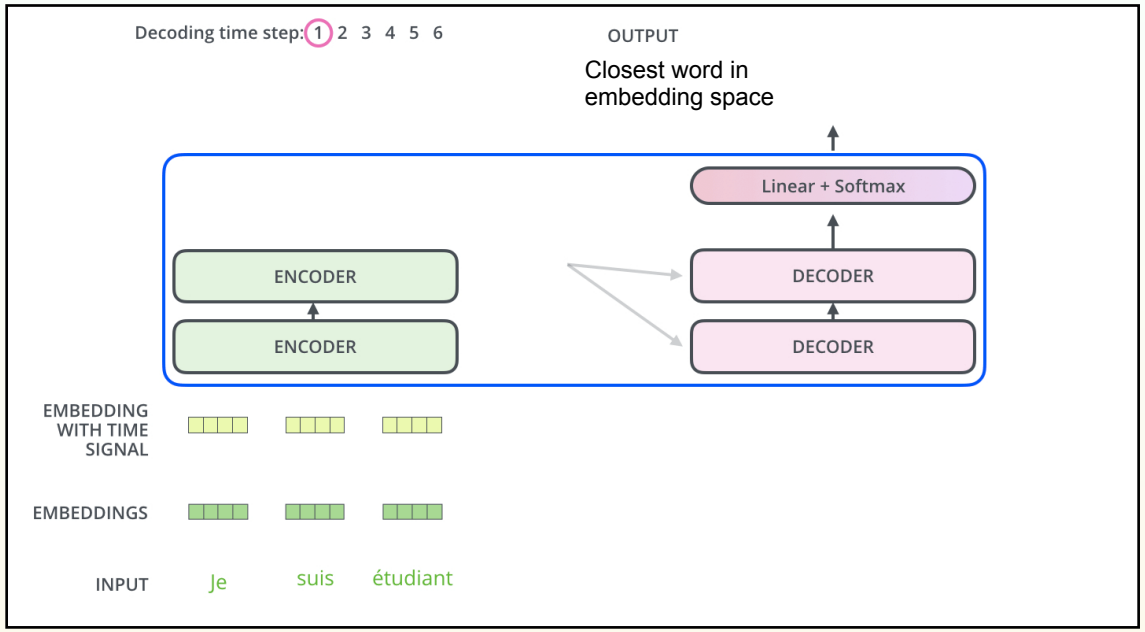
# Encoder / decoder interplay

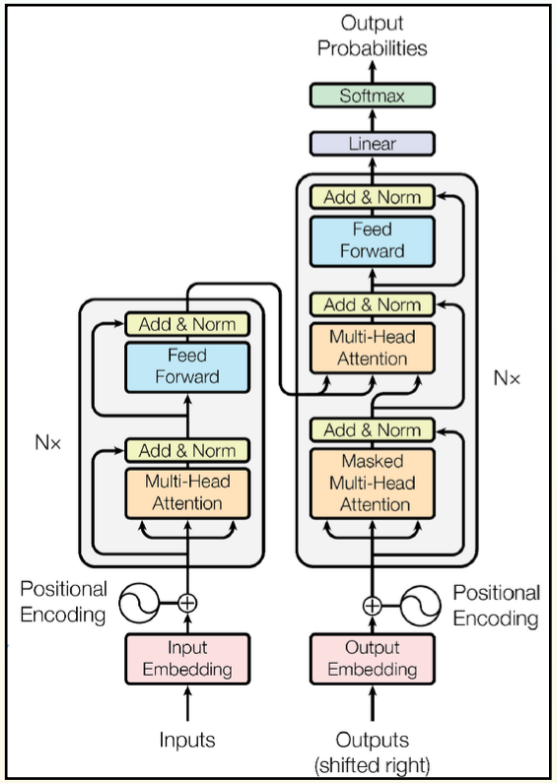



- The masking sets the attention for not yet known (future) fed-back outputs to zero
- Cross-attention: the decoder attends to the inputs with its own queries (similar to the image example) **generated from the previous output**

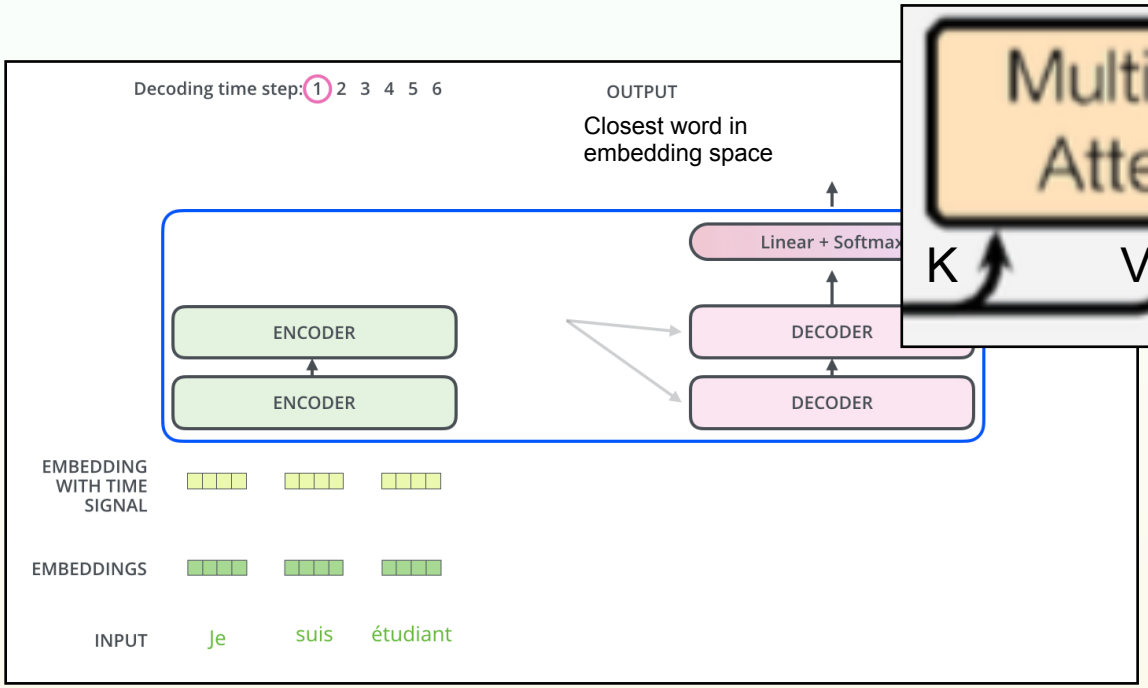
# Encoder / decoder interplay

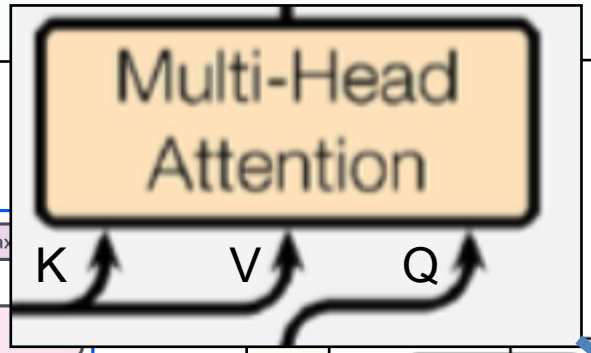


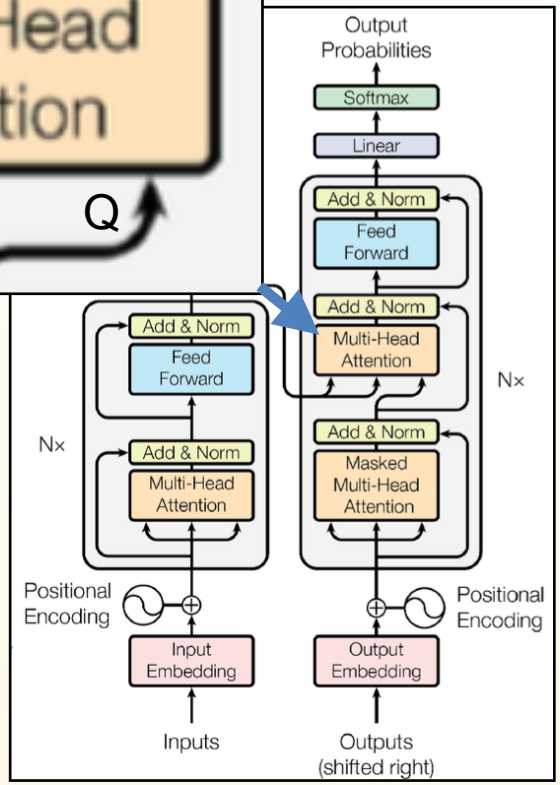




- The masking sets the attention for not yet known (future) fed-back outputs to zero
- Cross-attention: the decoder attends to the inputs with its own queries (similar to the image example) **generated from the previous output**

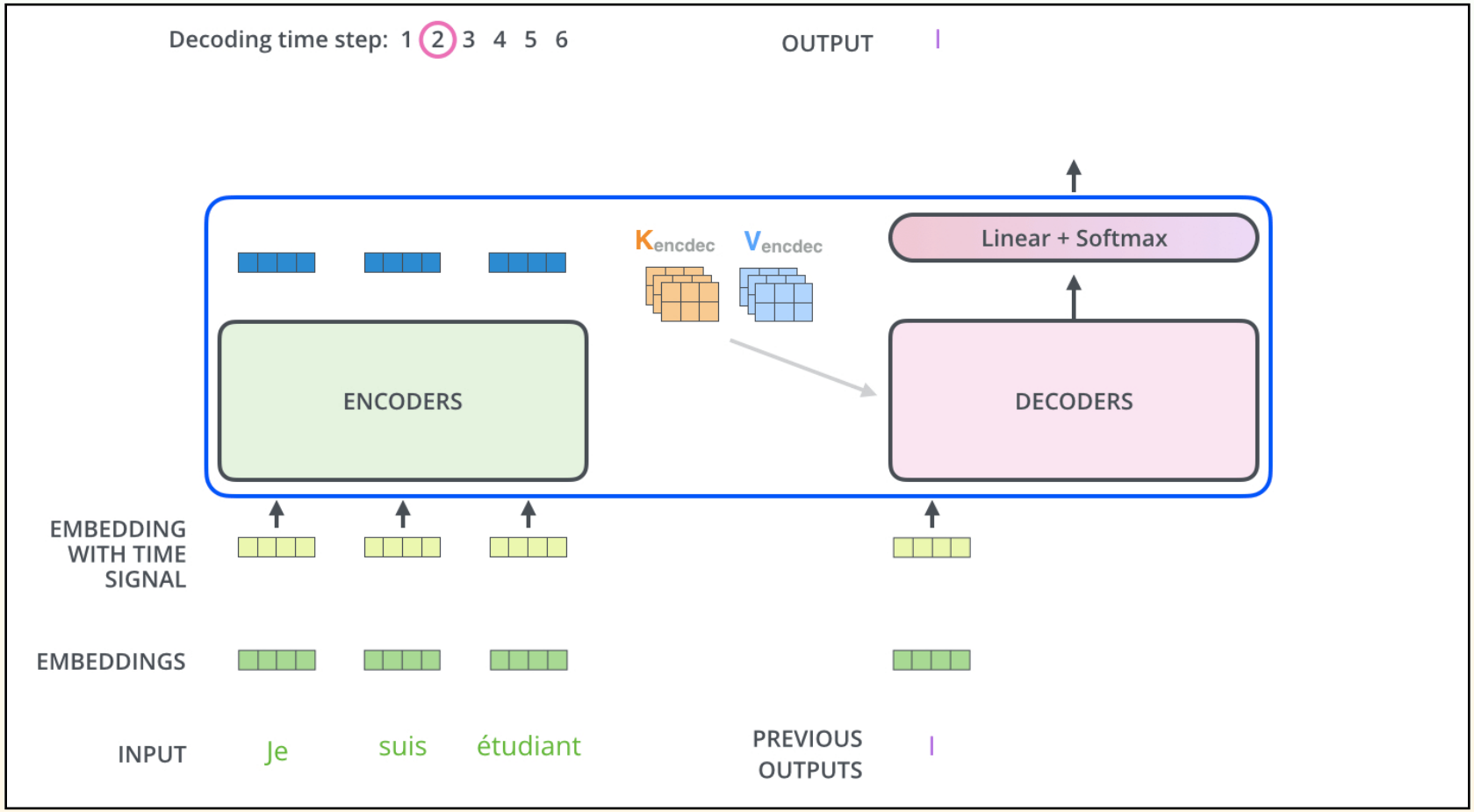
# Encoder / decoder interplay

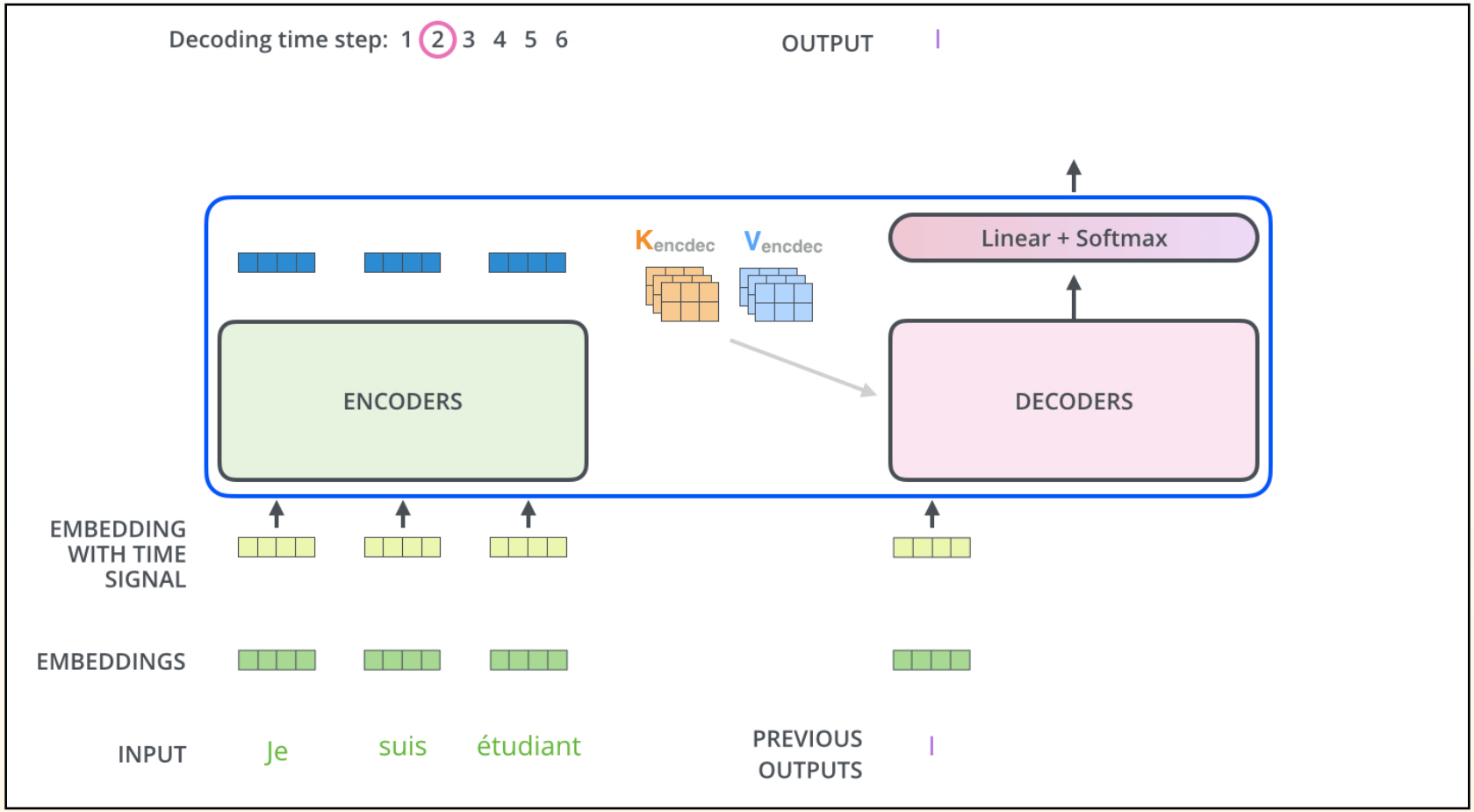



- The masking sets the attention for not yet known (future) fed-back outputs to zero
- Cross-attention: the decoder attends to the inputs with its own queries (similar to the image example) **generated from the previous output**


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K encdec
V encdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K encdec
V encdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Encoder

Multi-attention-head

Decoder

Keys

# Time for some questions

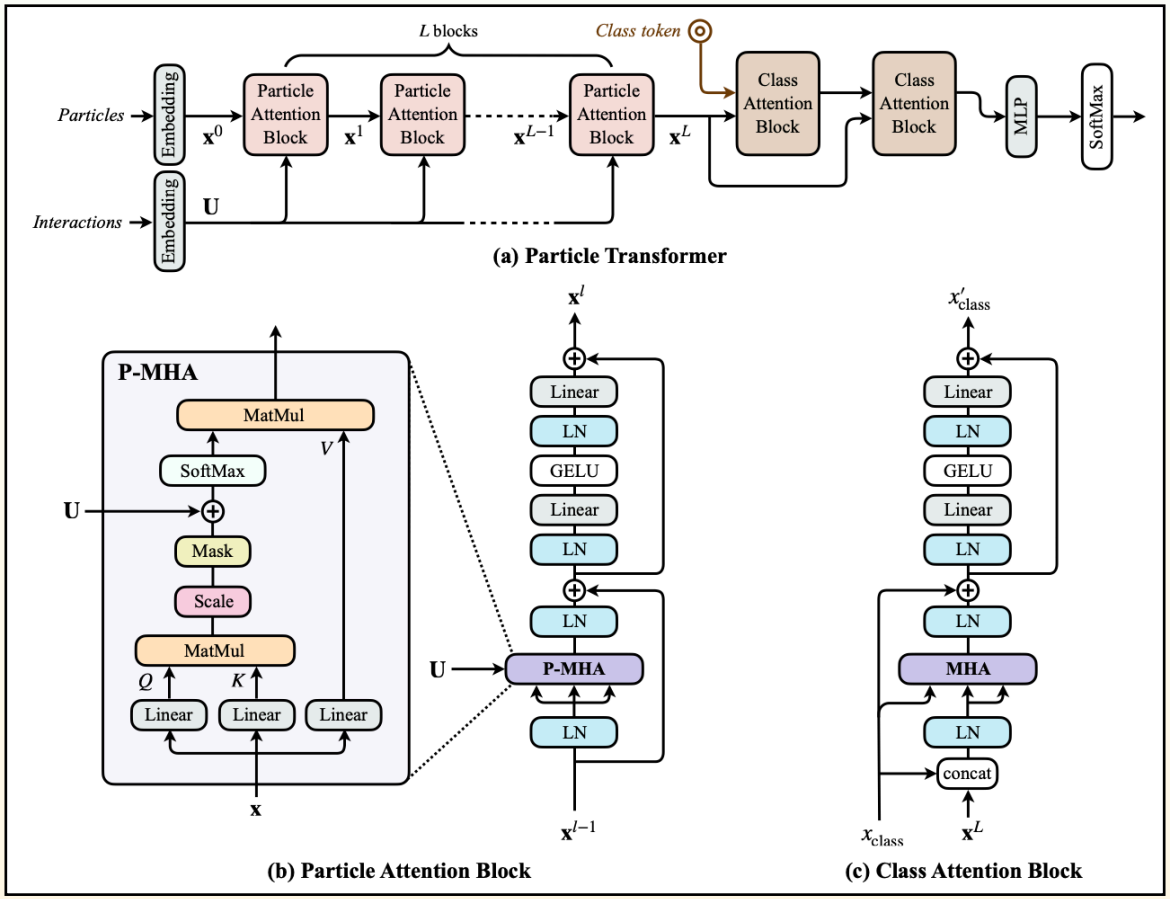
Queries

1D CNN

Output

# Transformers in physics

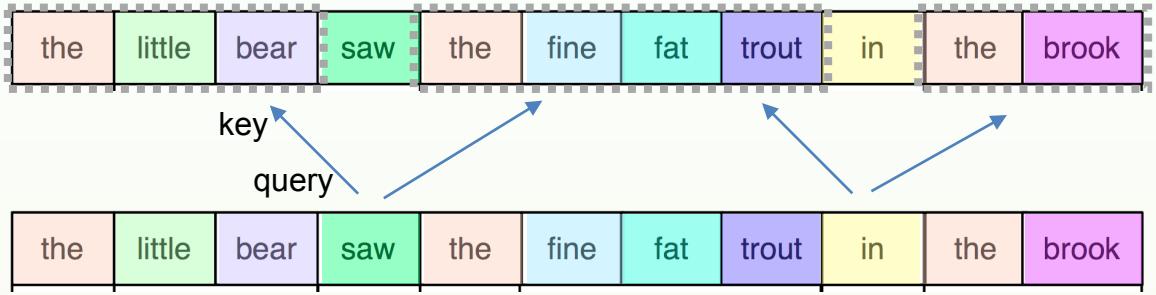
- Transformer applied to jet tagging, small modifications of the architecture
- Large improvement over other methods
  (on test data, not yet shown in analyses)




| | Accuracy | AUC | $\text{Rej}_{50\%}$ | $\text{Rej}_{30\%}$ |
|---|---|---|---|---|
| P-CNN | 0.930 | 0.9803 | $201 \pm 4$ | $759 \pm 24$ |
| PFN | — | 0.9819 | $247 \pm 3$ | $888 \pm 17$ |
| ParticleNet | 0.940 | 0.9858 | $397 \pm 7$ | $1615 \pm 93$ |
| JEDI-net (w/ $\sum O$) | 0.930 | 0.9807 | — | 774.6 |
| PCT | 0.940 | 0.9855 | $392 \pm 7$ | $1533 \pm 101$ |
| LGN | 0.929 | 0.964 | — | $435 \pm 95$ |
| rPCN | — | 0.9845 | $364 \pm 9$ | $1642 \pm 93$ |
| LorentzNet | 0.942 | 0.9868 | $498 \pm 18$ | $2195 \pm 173$ |
| ParT | 0.940 | 0.9858 | $413 \pm 16$ | $1602 \pm 81$ |
| ParticleNet-f.t. | 0.942 | 0.9866 | $487 \pm 9$ | $1771 \pm 80$ |
| **ParT-f.t.** | **0.944** | **0.9877** | $\mathbf{691 \pm 15}$ | $\mathbf{2766 \pm 130}$ |

arxiv:2202.03772

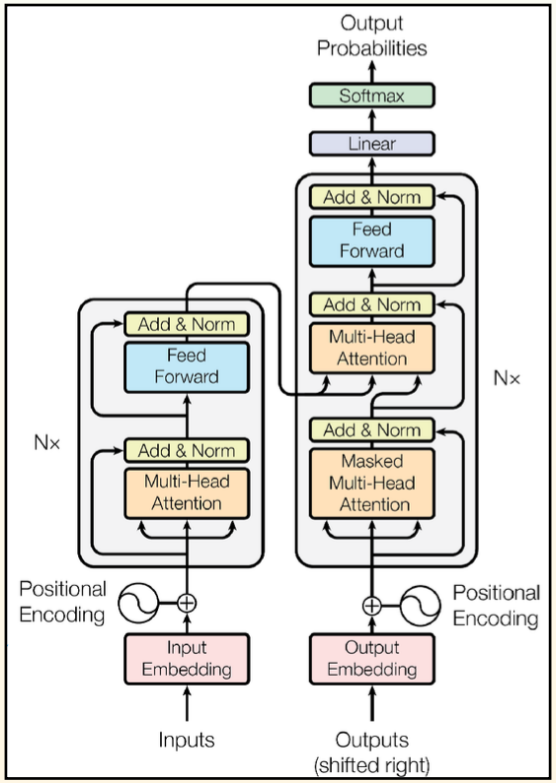
# Summary: attention and transformers



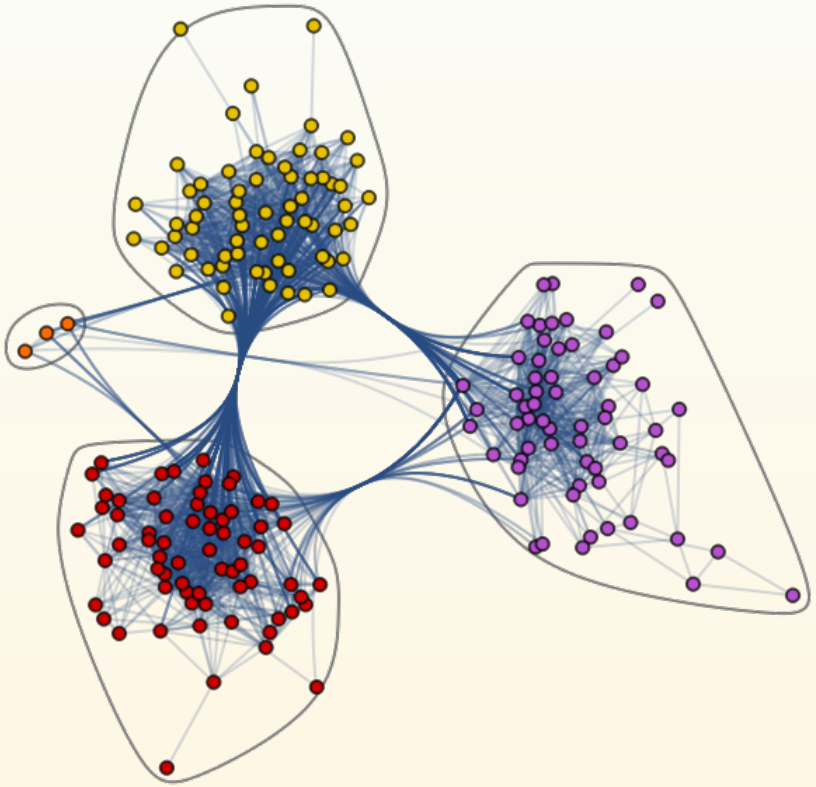

- Attention is a directed ‘weight’ determined by learnable key and query pairs

- It only acts on pair-wise relations
  - Permutation invariant
  - Sequence-length independent

- Transformers (e.g. for translation)
  - Add positional encodings
  - Output word probabilities, one by one
  - Use previous output to generate next word
  - Are on average the most powerful architectures around



# Graph Neural Networks



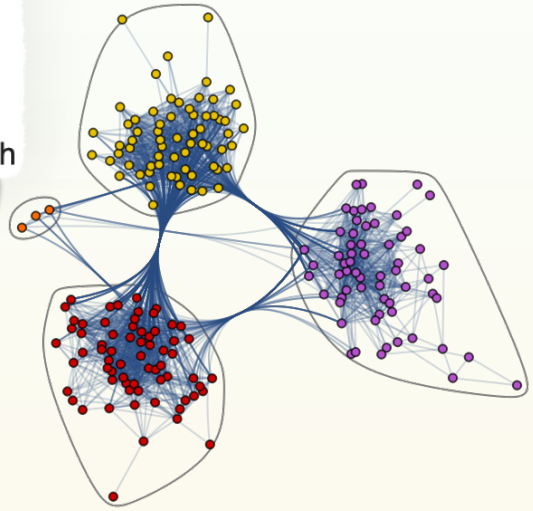
# Graphs

**Graph** [ edit ]

In one restricted but very common sense of the term,[1][2] a **graph** is an ordered pair $G = (V, E)$ comprising:

- $V$, a set of **vertices** (also called **nodes** or **points**);
- $E \subseteq \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$ , a set of **edges** (also called **links** or **lines**), which

https://en.wikipedia.org/wiki/Graph_theory, 27.7.23



- Do you know some data that can be represented by one?

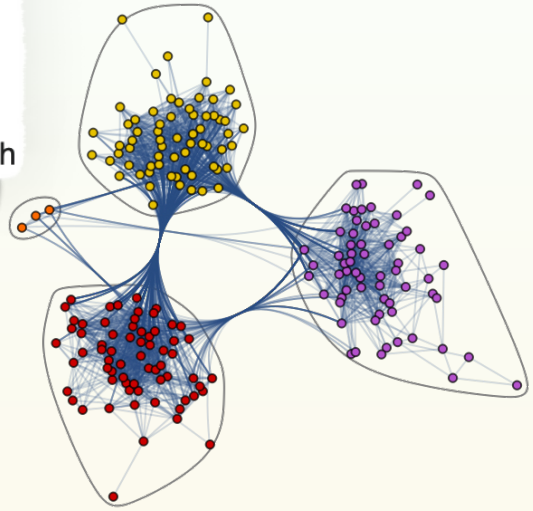
# Graphs

**Graph** [ edit ]

In one restricted but very common sense of the term,[1][2] a **graph** is an ordered pair $G = (V, E)$ comprising:

- $V$, a set of **vertices** (also called **nodes** or **points**);
- $E \subseteq \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$ , a set of **edges** (also called **links** or **lines**), which

https://en.wikipedia.org/wiki/Graph_theory, 27.7.23



- Do you know some data that can be represented by one?

- Facebook: G(accounts, friend relations)
  - Undirected
- Twitter: G(accounts, who's following who)
  - Directed

# Graphs

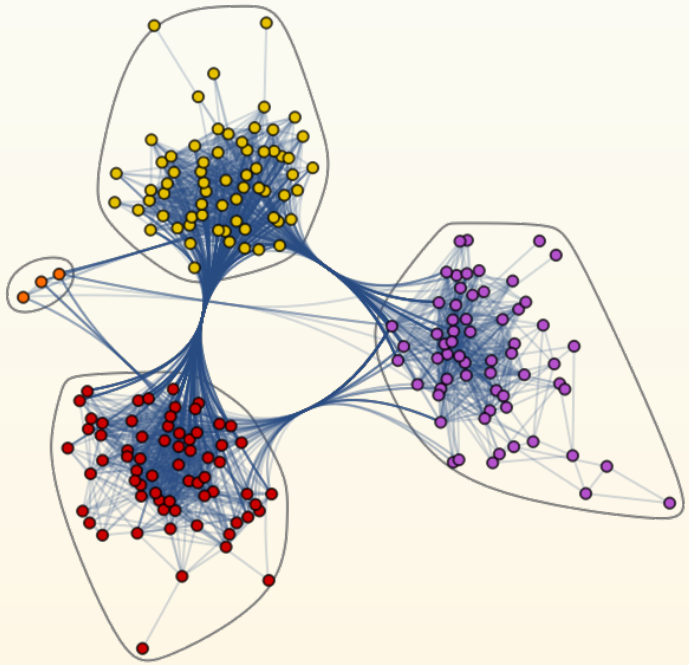
**Graph** [ edit ]

In one restricted but very common sense of the term,[1][2] a **graph** is an ordered pair $G = (V, E)$ comprising:

- $V$, a set of **vertices** (also called **nodes** or **points**);
- $E \subseteq \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$ , a set of **edges** (also called **links** or **lines**), which

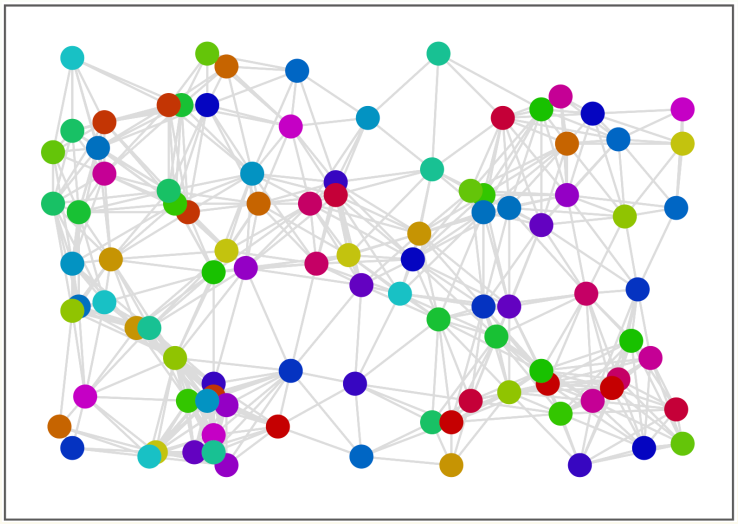
https://en.wikipedia.org/wiki/Graph_theory, 27.7.23

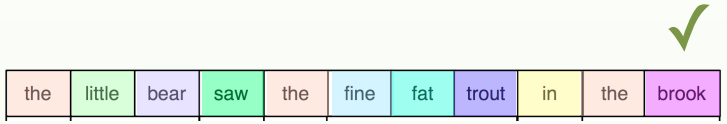
- Vertices can have properties
  - Colour
  - Age, favourite bar, hair style
  - General feature vector
- Edges can have properties
  - Direction
  - ‘Strength’, ‘Length’
  - General feature vector

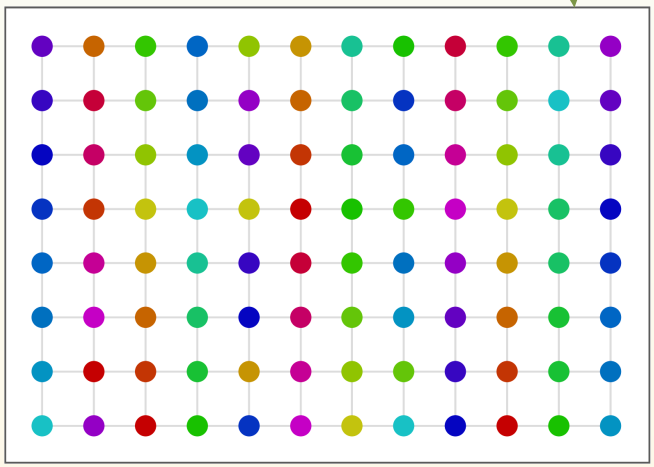


- Graphs are a very generic tool to describe “things” and “relations between them”

# What can't we do with a graph?





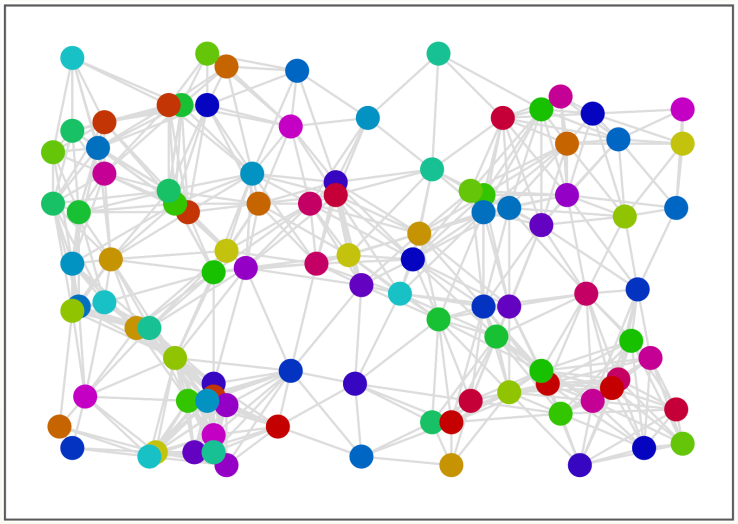



- No straight-forward particular ordering
  → no sequence processing

- No grid
  → no straight-forward neighbour indexing, no CNN

- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



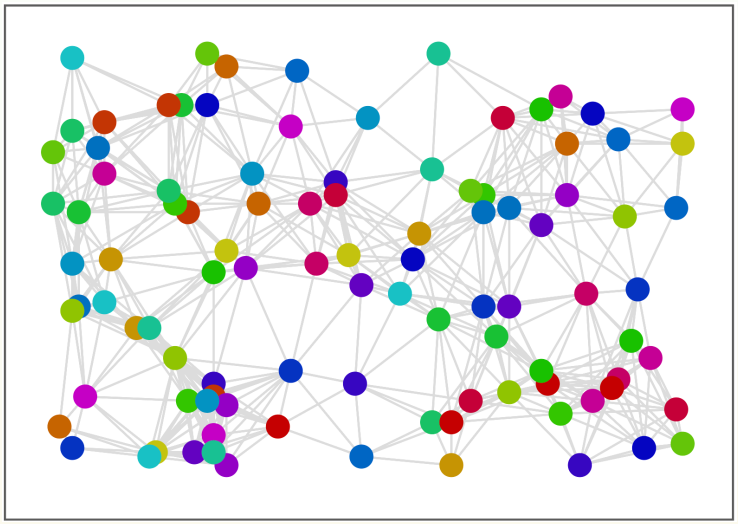
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



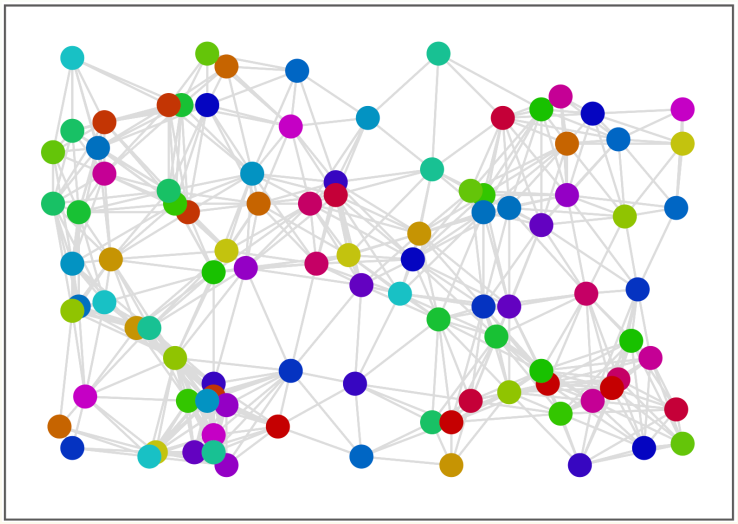
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



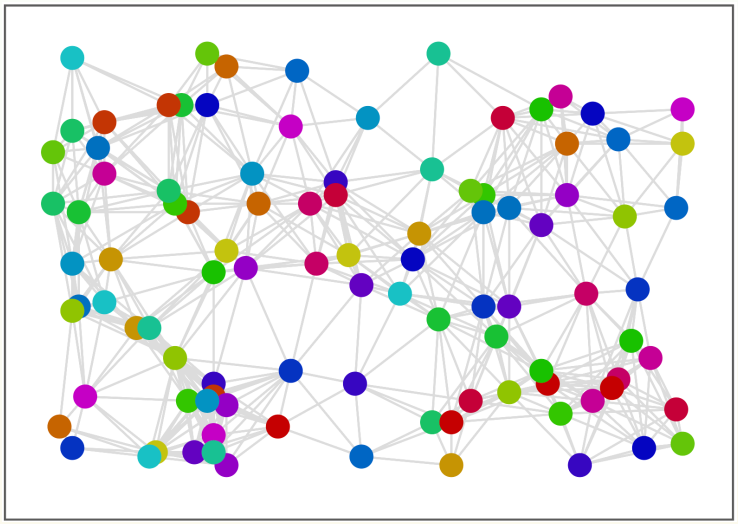
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



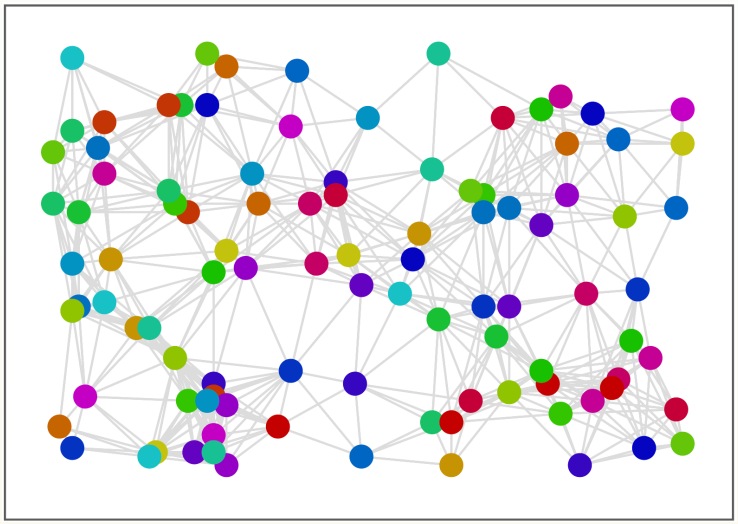
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



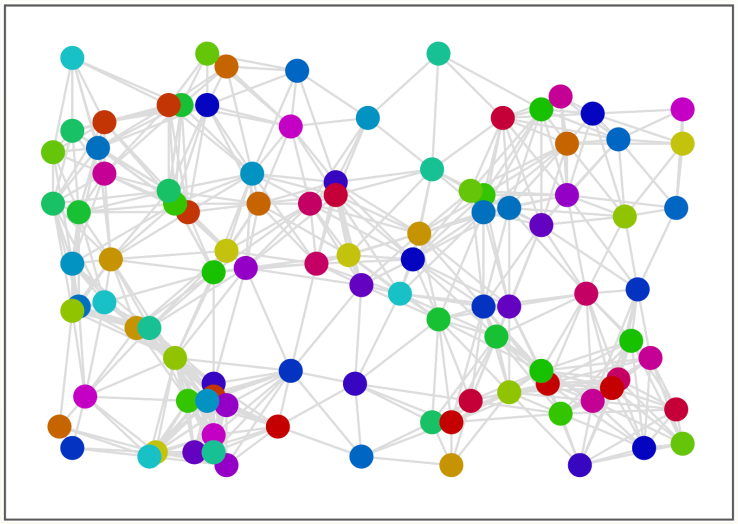
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



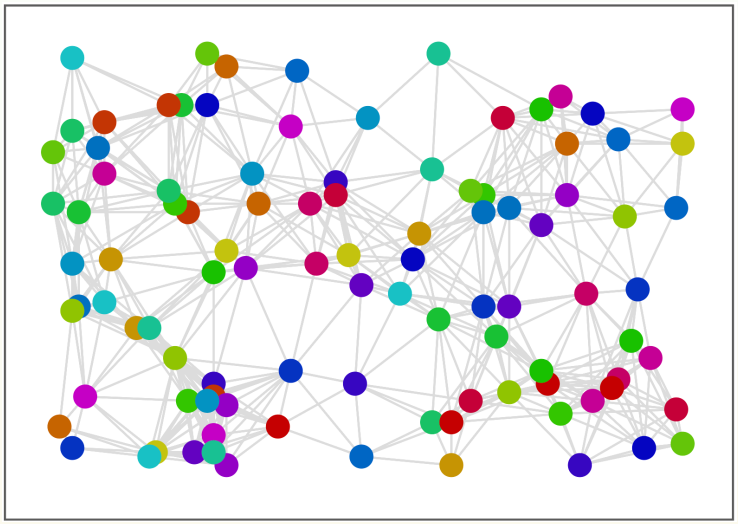
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



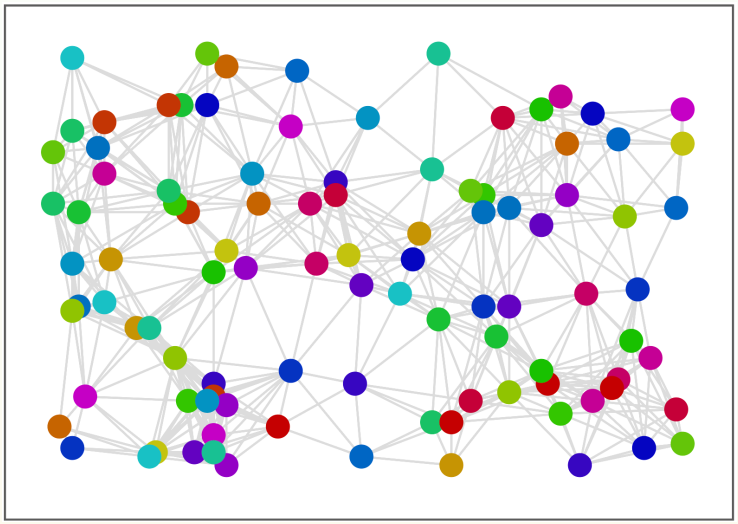
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



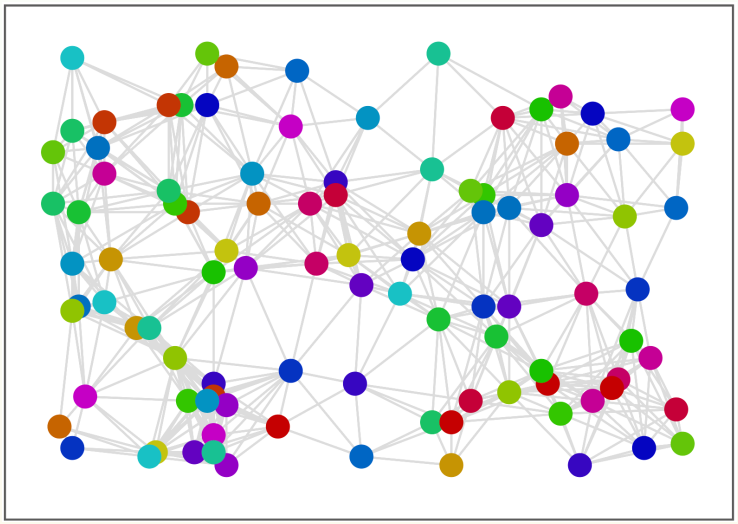
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



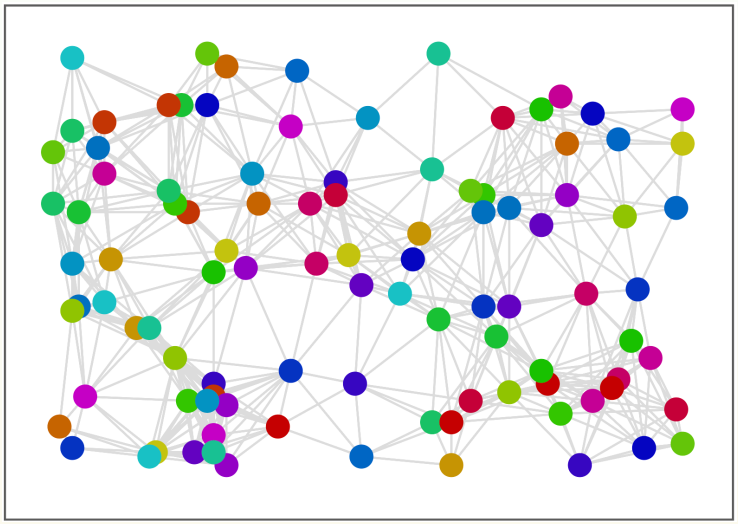
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



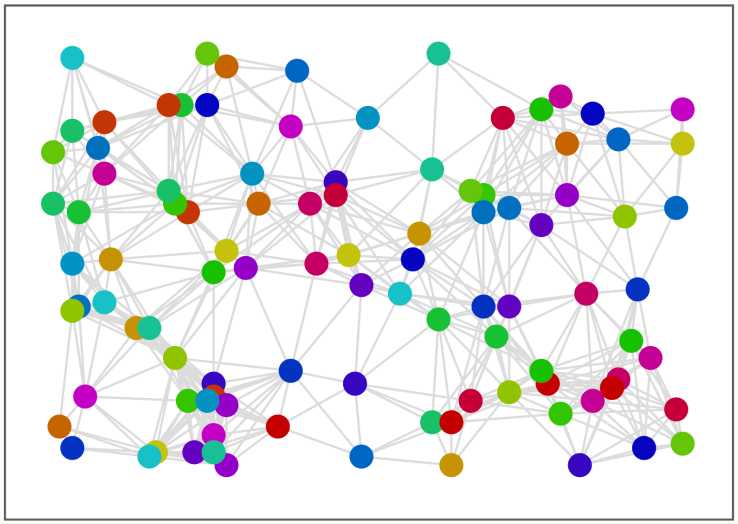
Atoms in folded molecules

Sensors in detectors

Particles in an event

- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

# What can't we do with a graph?



Atoms in folded molecules

Sensors in detectors

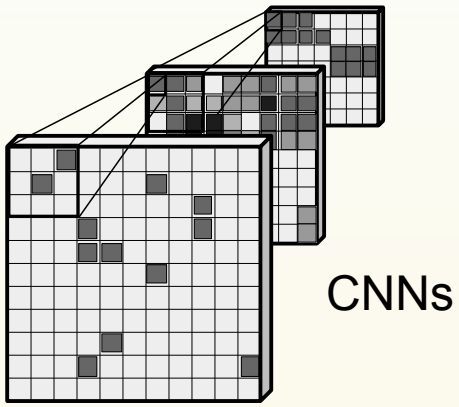
Particles in an event

***Fully connected*** graph

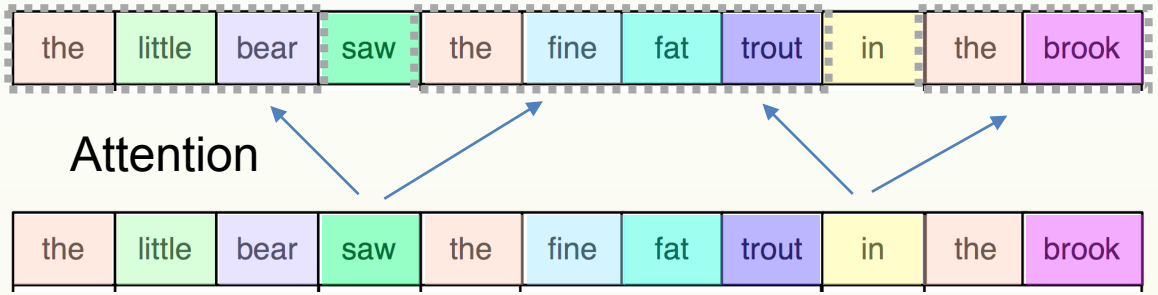
- No straight-forward particular ordering
  → no sequence processing
- No grid
  → no straight-forward neighbour indexing, no CNN
- No a-priori fixed size
  → can't just feed it into an MLP as a whole

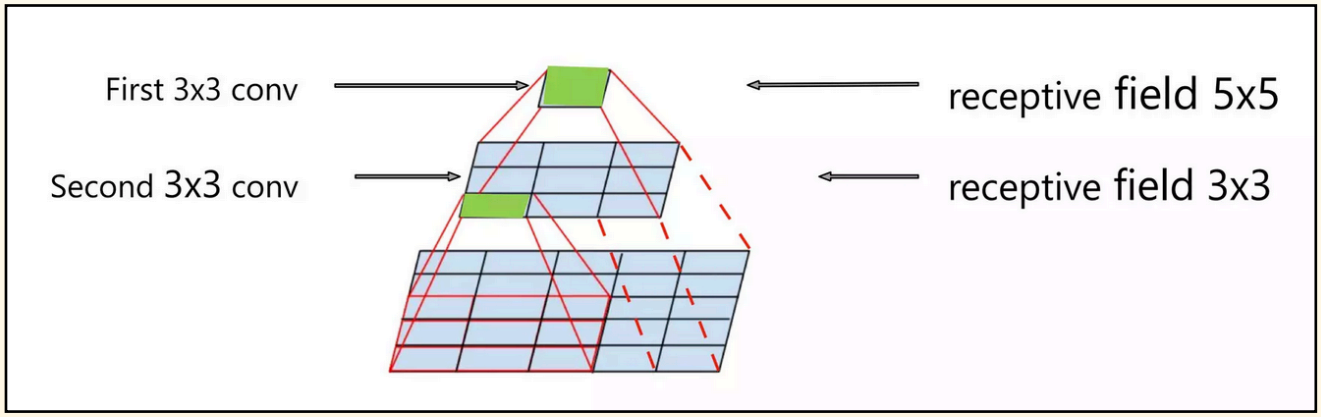
# Graph processing

- For a DNN to learn something, information needs to be exchanged
- Reminders:



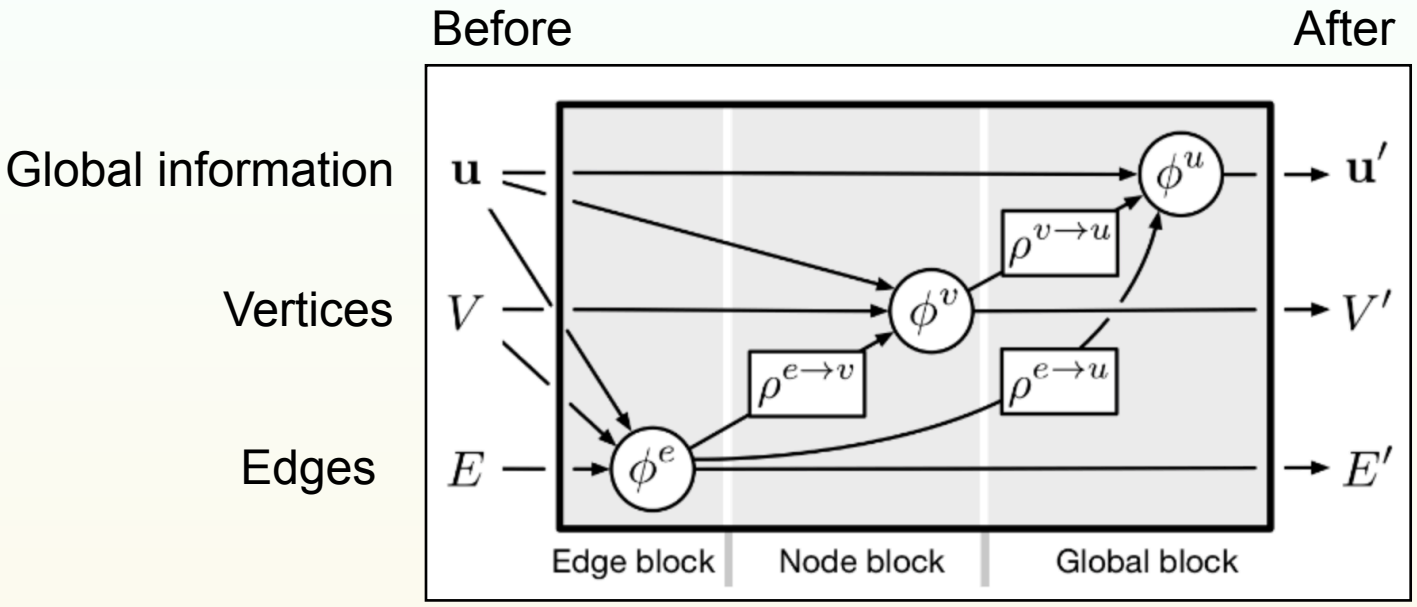
CNNs



# Information exchange in a graph





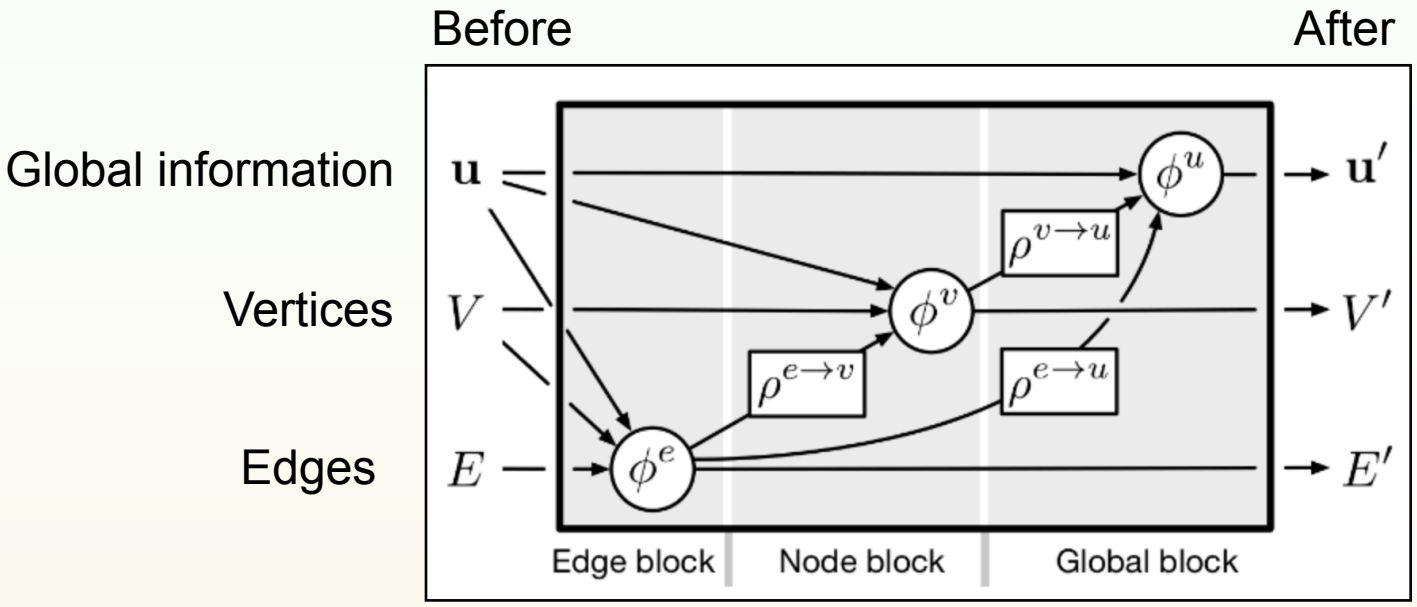

arXiv:1806.01261

- Various ways to exchange information across a graph

- Functions $\rho$ aggregate information from different sized sets
- Aggregation functions must respect order invariance, which do?

# Information exchange in a graph






arXiv:1806.01261

- Various ways to exchange information across a graph

- Functions $\rho$ aggregate information from different sized sets
- Aggregation functions must respect order invariance, which do?

$$\rho = \sum, \prod, \max, \min$$ **that simple.**

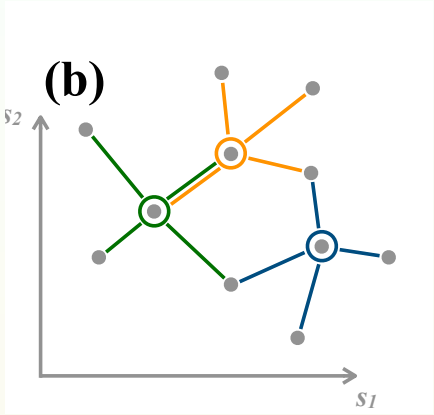
# Message passing GNNs

- First take: information exchange by aggregating messages from all neighbours of a node:

$$y_j = \sum_{i}^{N_k} x_{I(j,i)}$$

$N_k$: Number of neighbours

$I(j,i)$: Index $m$ of vertex that is connected to vertex $j$




- Seems familiar?

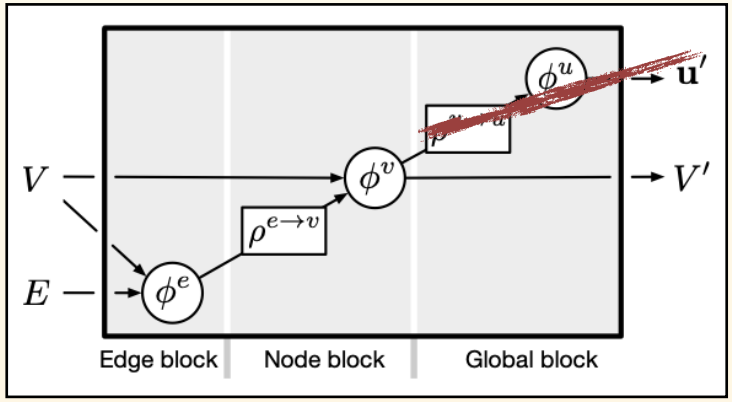
That's a convolution with $\omega_{ij} = 1$

- So far no free parameters, add an MLP layer before:

$$x_{i\beta} = \theta\left(\sum_{\alpha} \omega_{\beta\alpha}\tilde{x}_{i\alpha} + T_\beta\right)$$

$$y_{j\beta} = \sum_{i}^{N_k} x_{I(j,i)\beta}$$

Reminder: could also be $\prod$, max, ...




- This is a simple message passing NN

# A more complex MP NN



- Allow for a different number of neighbours for simple MP:

$$y_{j\beta} = \square_{i \,\epsilon\, N(j)} x_{i\beta}$$

Can stand for $\sum, \prod, \ldots$

Neighbour indices of vertex $j$, e.g. $N(5) = \{23, 43, 2, 4\}$

- Another way of expressing it: $y_{j\beta} = \square_{i} A_{ji} x_{i\beta}$

All vertices

Adjacency matrix, here 0 or 1 (can be sparse)

- In general, the message might also depend on $x_j$

$$y_{j\beta} = \square_{i \,\epsilon\, N(j)} \Phi_\beta(x_j, x_i)$$

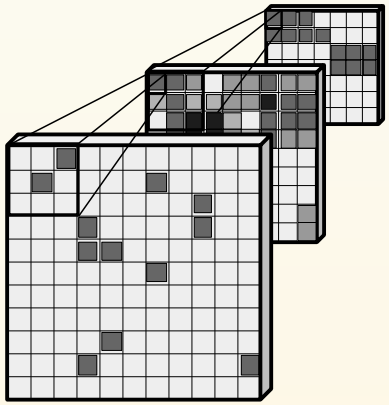
This is a neural network

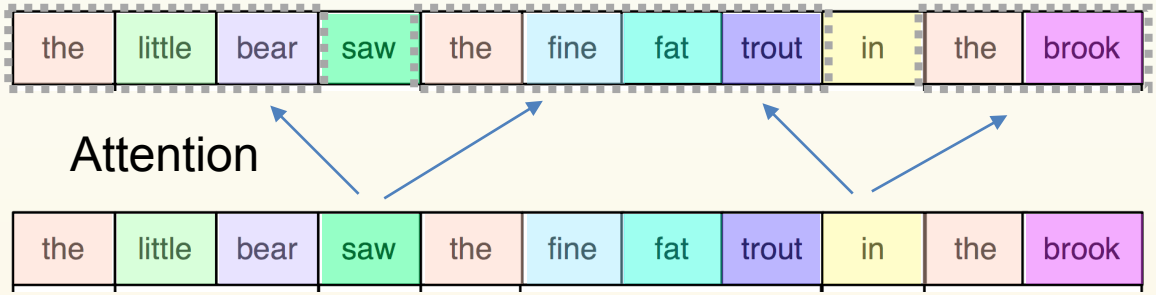
# Generalisation of the formalism

- Information exchange across some sort of connection possibly with a weight, possibly depending on that connection…
- Have we seen this before, qualitatively?

# Generalisation of the formalism

- Information exchange across some sort of connection possibly with a weight, possibly depending on that connection…
- Have we seen this before, qualitatively?





# One formalism to rule them all



Now all vectors

A general MP: $y_j = \square_{i \in N(j)} \Phi(x_j, x_i)$

An attention aggregation: $y_j = \square_{i \in N(j)} a(x_j, x_i) \cdot \Phi(x_i)$

Here, these are all other vertices (fully connected)

A CNN aggregation: $y_j = \square_{i \in N(j)} c_{ij} \cdot \Phi(x_i)$

These depend on the relative position of $i$ to $j$

The graph formalism is very powerful

Very nice, more in-depth overview: arxiv:2301.08210

Message passing

$$y_{j\beta} = \square_{i \in N(j)} x_{i\beta}$$

Permutation invariance

Non-linearity

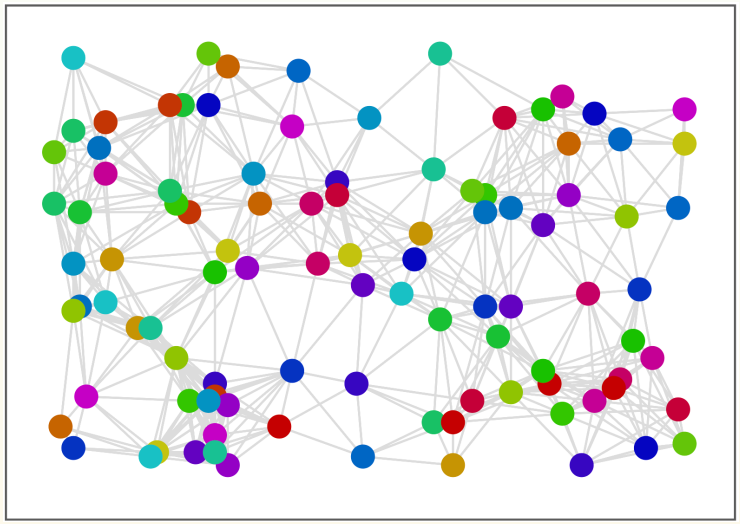
# Time for some questions

Connection to CNNs

Connection to attention

Neighbourhoods

# Our toolbox



- NB: $\square$ is effectively a low-pass, simply stacking many layers can lead to over-smoothing; performance degrades
- But where does the graph actually come from?

Simple MP

$$y_{j\beta} = \square_{i \in N(j)} x_{i\beta}$$

CNN-like

$$y_j = \square_{i \in N(j)} c_{ij} \cdot \Phi(x_i)$$

Why not?

Attention based

$$y_j = \square_{i \in N(j)} a(x_j, x_i) \cdot \Phi(x_i)$$

General MP

$$y_j = \square_{i \in N(j)} \Phi(x_j, x_i)$$

# Representing physics data

- Particles, detector signals, … can be represented by *point clouds*

- A point cloud is a set of points, each with features (position, colour,…)

# Creating the edges

**Graph** [ edit ]

In one restricted but very common sense of the term,[1][2] a **graph** is an ordered pair $G = (V, E)$ comprising:

- $V$, a set of **vertices** (also called **nodes** or **points**);
- $E \subseteq \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$ , a set of **edges** (also called **links** or **lines**), which

- One simple solution: fully connected

Why not?

# Creating the edges

**Graph** [ edit ]

In one restricted but very common sense of the term,[1][2] a **graph** is an ordered pair $G = (V, E)$ comprising:

- $V$, a set of **vertices** (also called **nodes** or **points**);
- $E \subseteq \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$ , a set of **edges** (also called **links** or **lines**), which

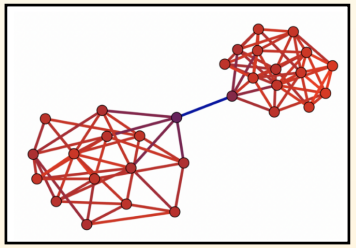
- One simple solution: fully connected

Why not?

- Connections scale with $N^2 \rightarrow$ becomes very resource intense


- Might not capture the best connections to solve the problem (sometimes less is more)

- Connect in physical space - everything within a radius?
  - Can lead to weak connections (or too many)

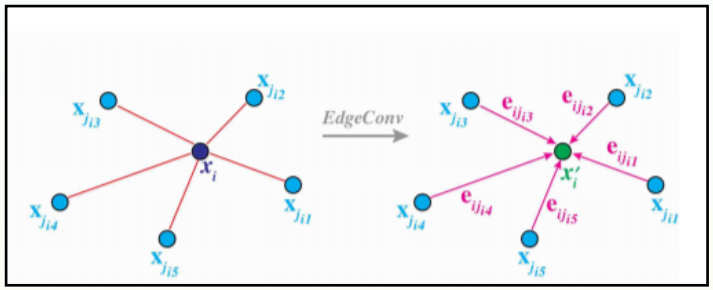


# Learning the edges: DGCNN

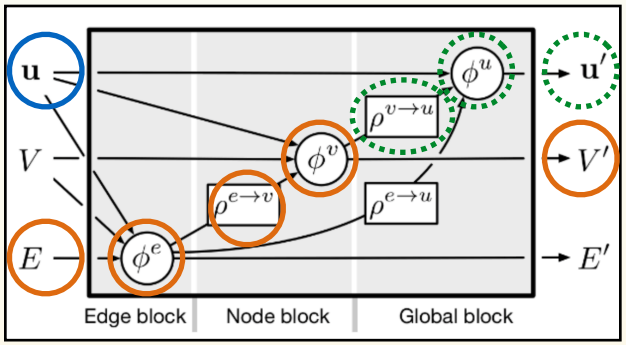
- 3D segmentation of point clouds: EdgeConv
  - Calculate distances **in full feature space**
  - Select **K** neighbours
- $y_j = \max_{i \epsilon N(j)} MLP(x_j, x_j - x_i)$

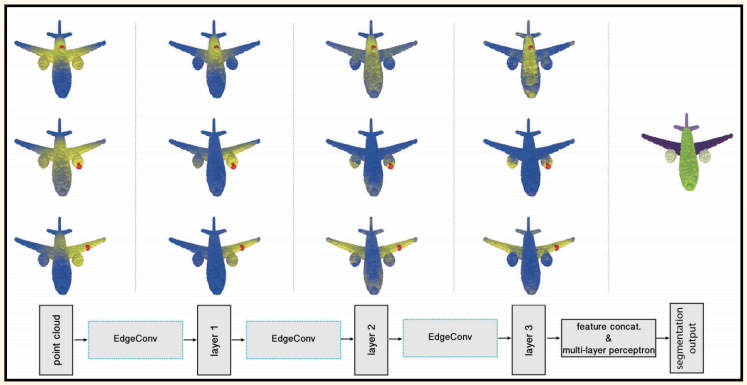
  What type of GNN?
- Proven very powerful for segmentation
- Very resource demanding network architecture

Multiple operations scale:
N x **K x F**



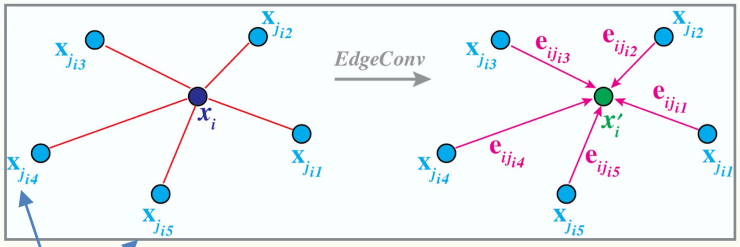

DGCNN: Repeat EdgeConv, then
determine u in penultimate iteration,
include u in last iteration



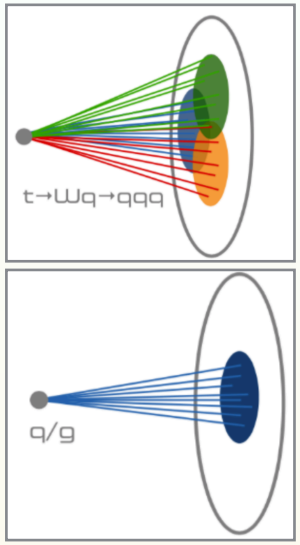



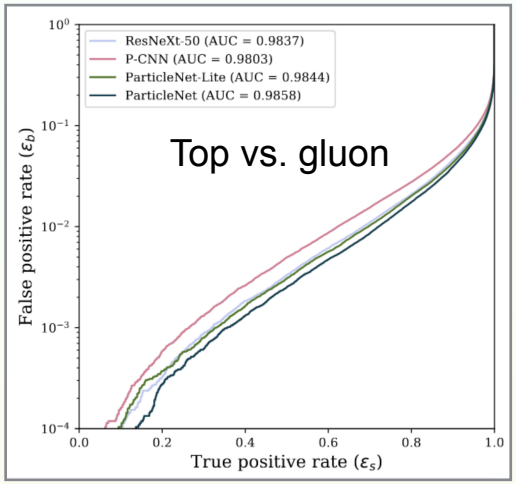

arXiv:1801.07829

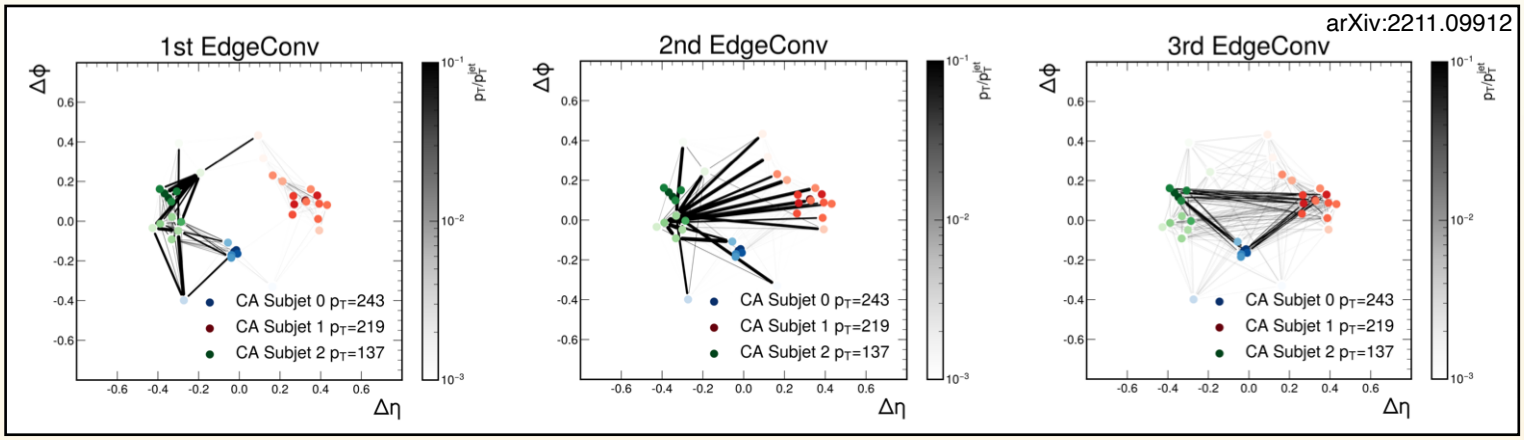
# DGCNN for jets (aka ParticleNet) arXiv:1902.08570




Embeddings of jet constituents

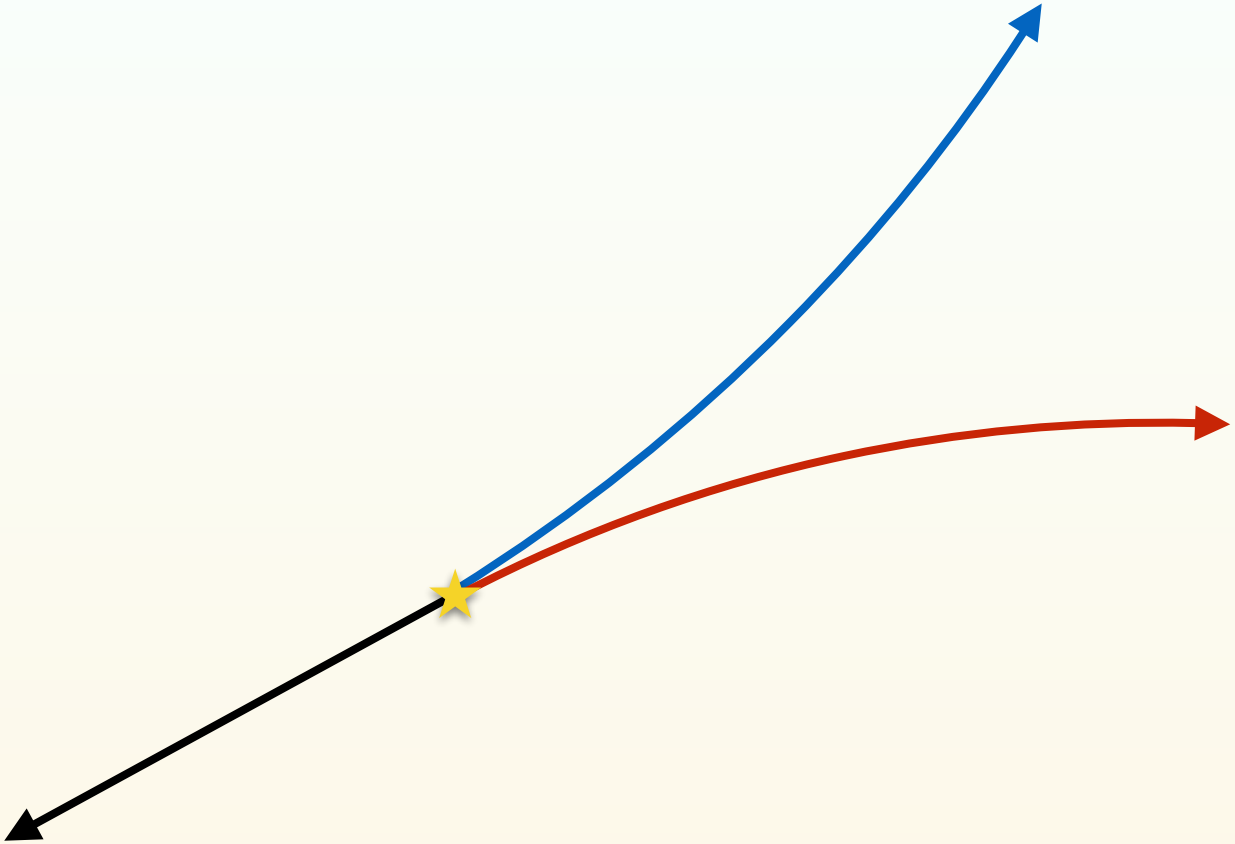


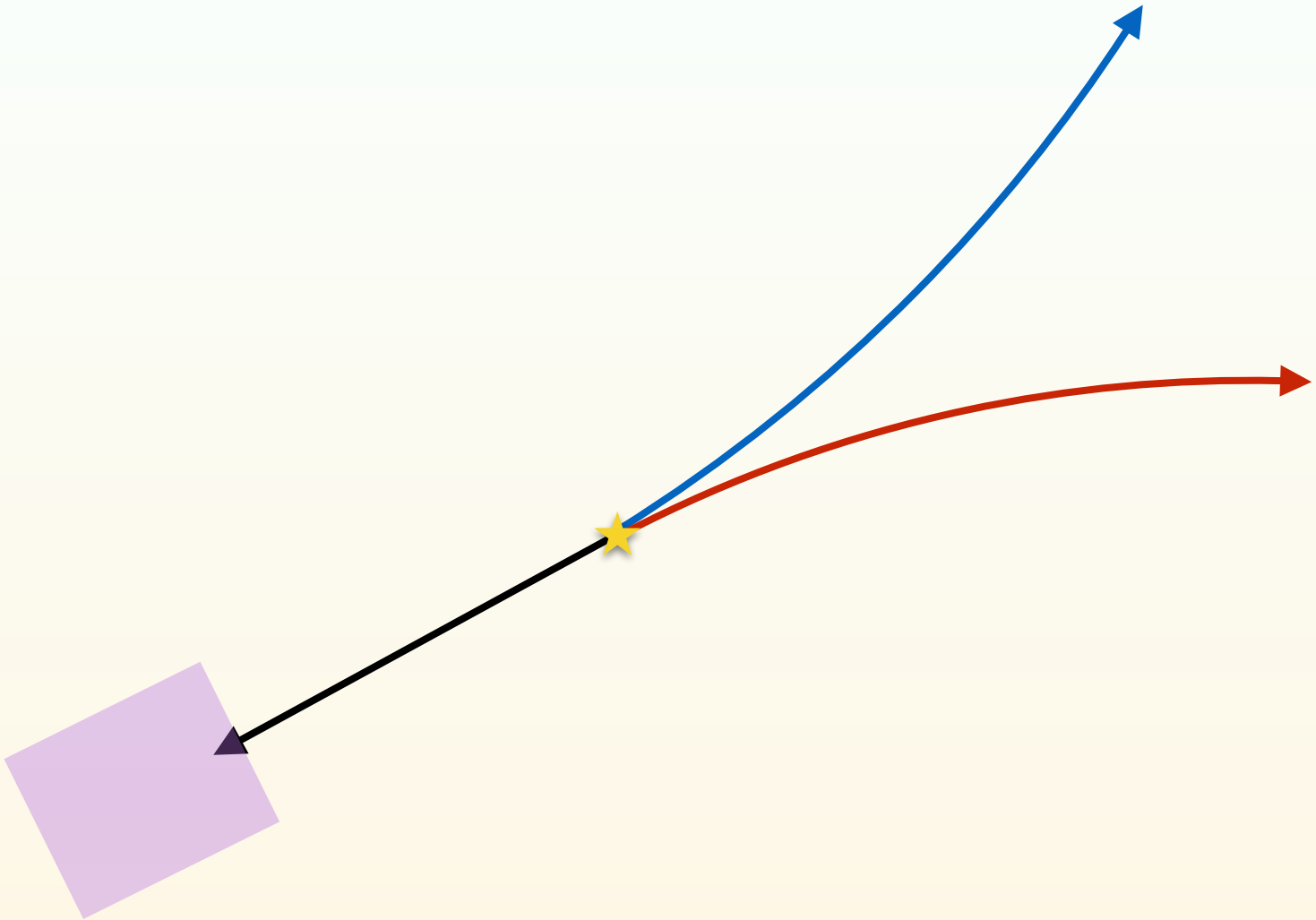

- Each iteration, a different graph is built
- The last iteration captures the sub-jets

# Going deeper: reconstruction



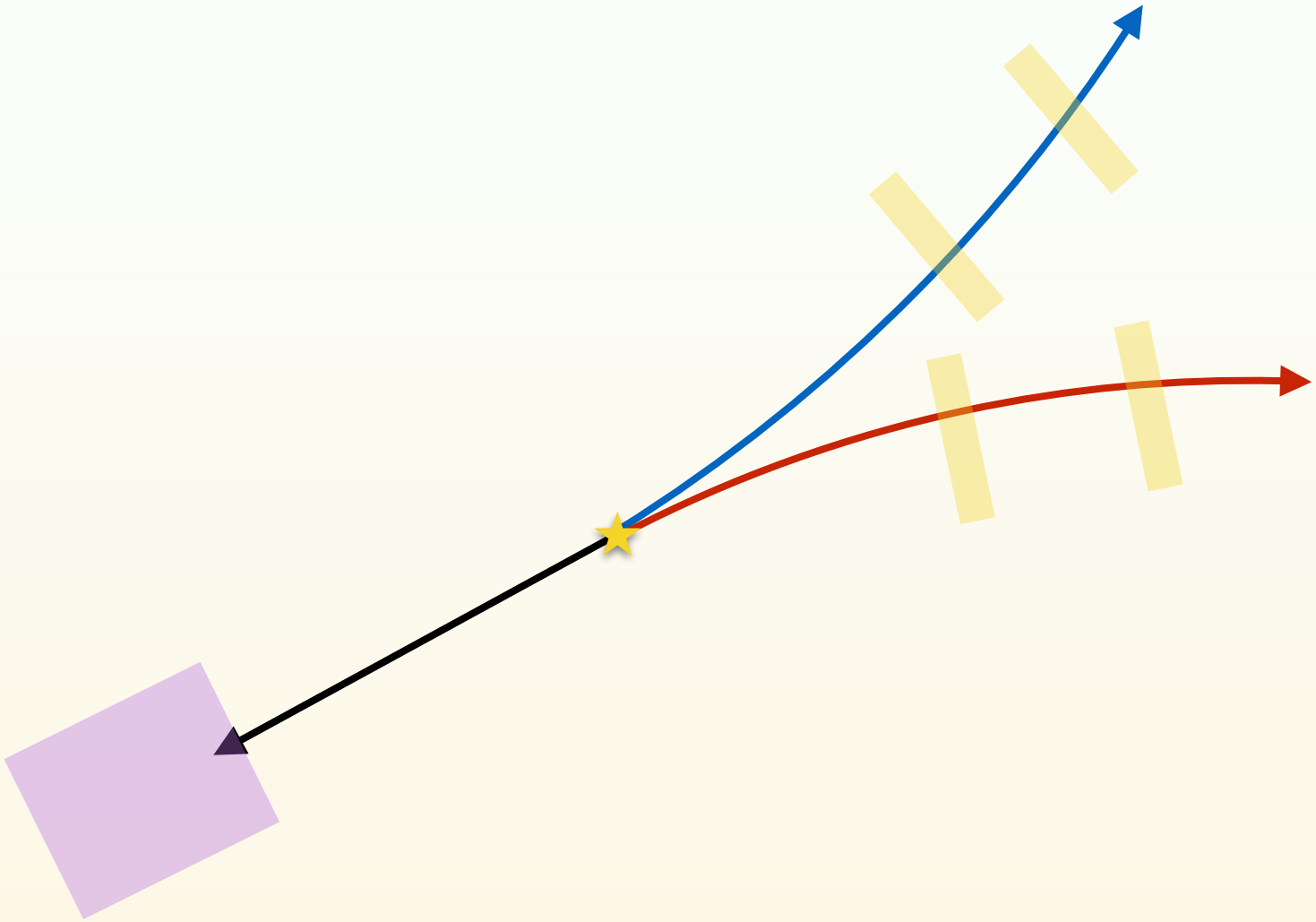
- Basic input for reconstruction of a particle are hits (calibrated, above threshold, …)
- Hits can have very different properties depending on the sub detector
- Unordered, not dense, … → GNNs

# Going deeper: reconstruction



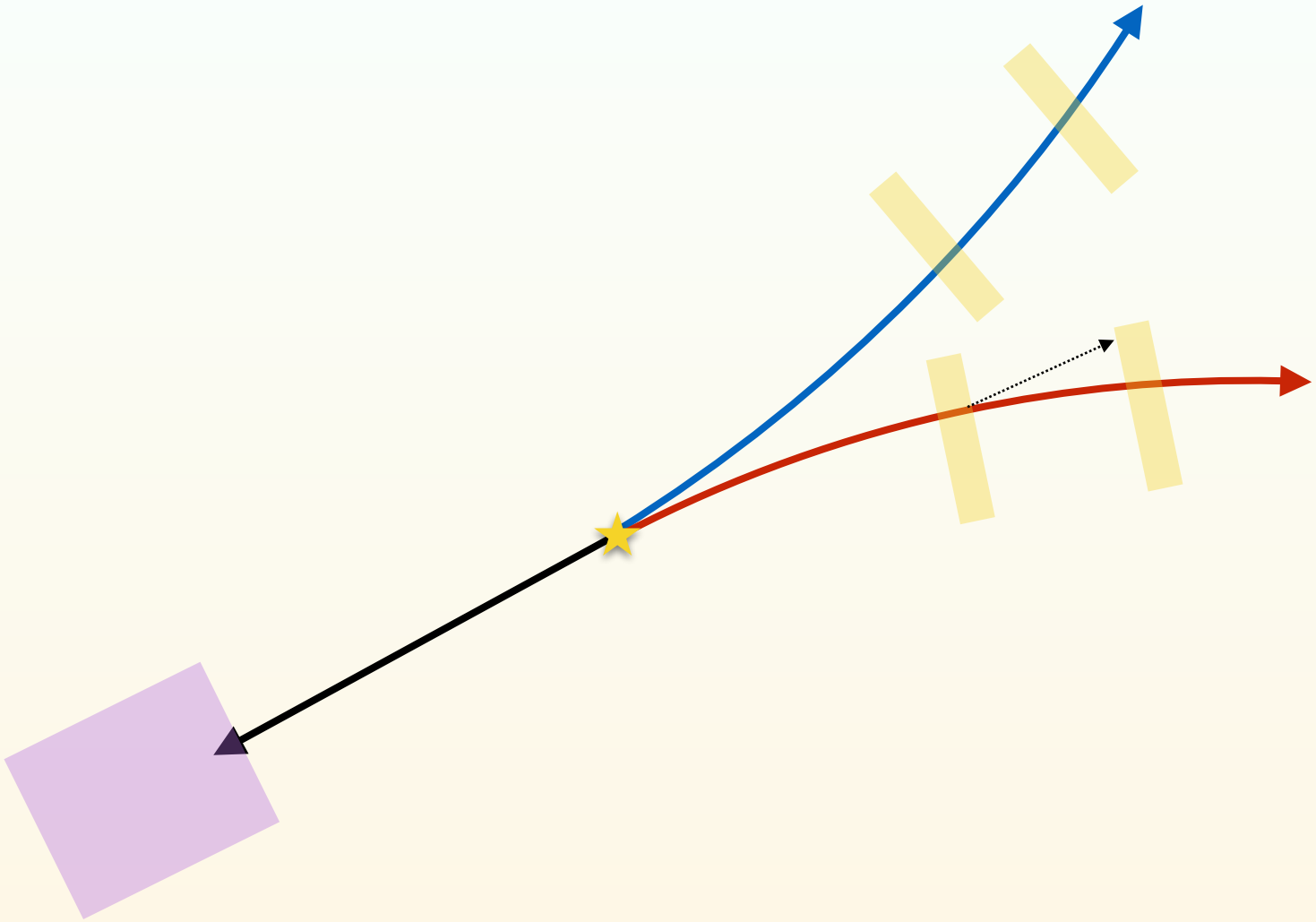
- Basic input for reconstruction of a particle are hits (calibrated, above threshold, …)
- Hits can have very different properties depending on the sub detector
- Unordered, not dense, … → GNNs

# Going deeper: reconstruction



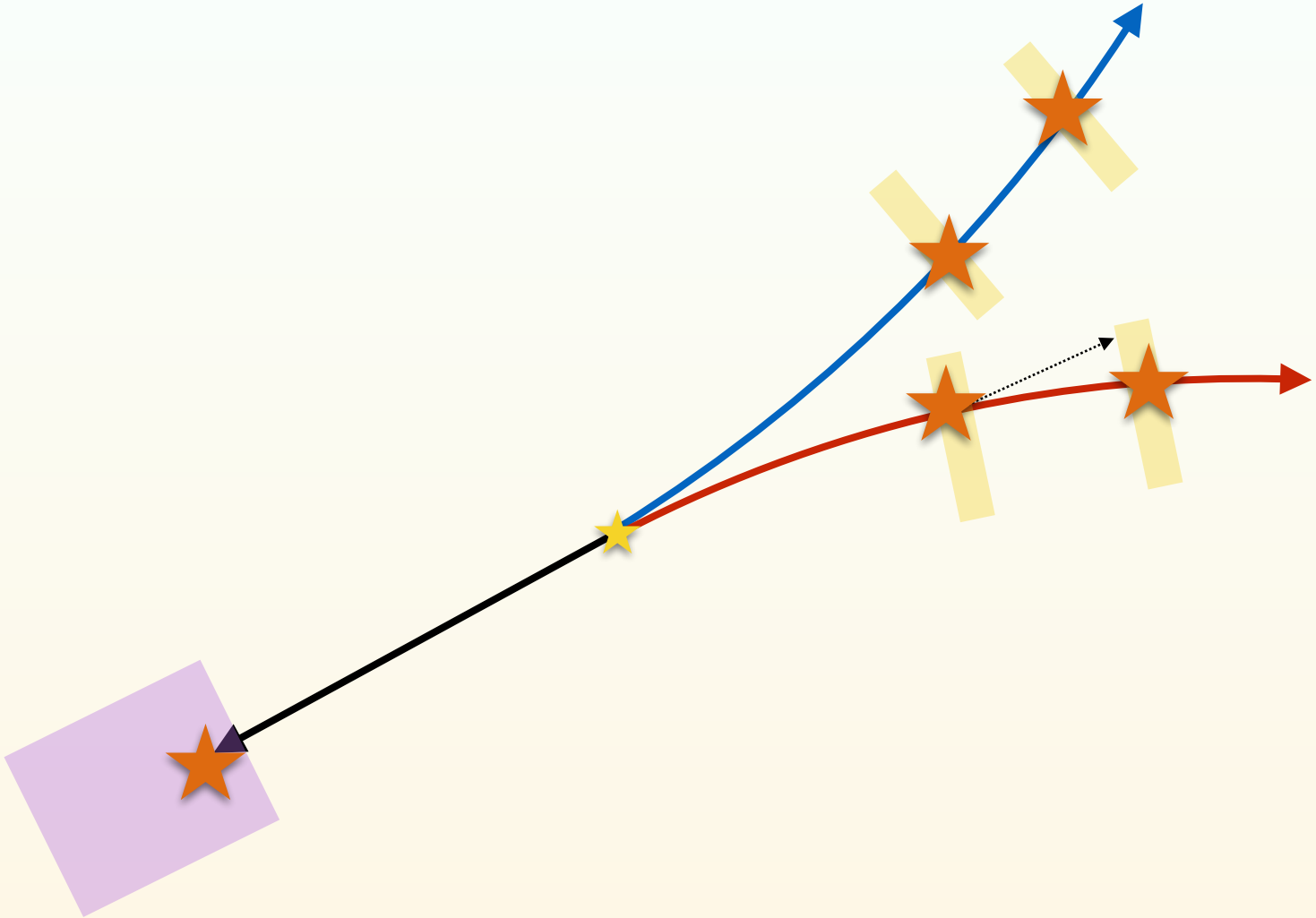
- Basic input for reconstruction of a particle are hits (calibrated, above threshold, …)
- Hits can have very different properties depending on the sub detector
- Unordered, not dense, … → GNNs

# Going deeper: reconstruction



- Basic input for reconstruction of a particle are hits (calibrated, above threshold, …)
- Hits can have very different properties depending on the sub detector
- Unordered, not dense, … → GNNs

# Going deeper: reconstruction



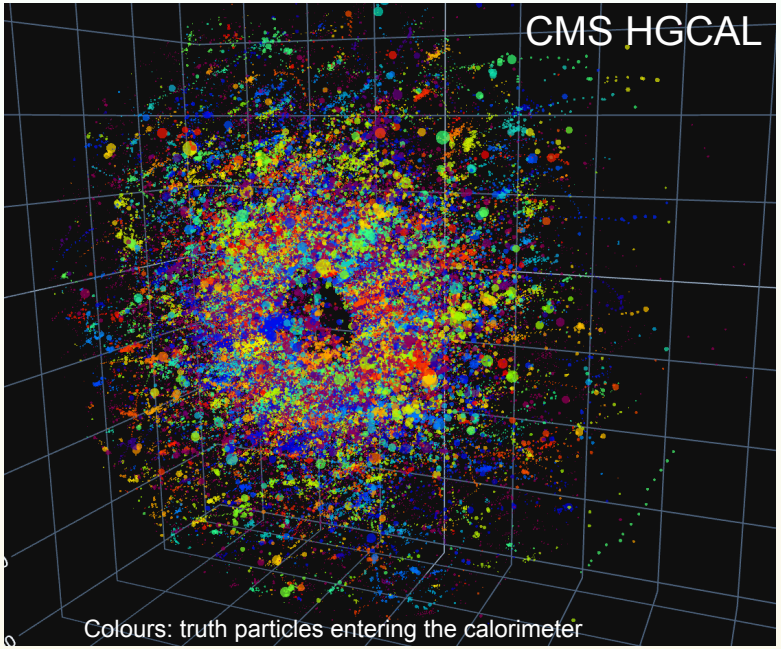
- Basic input for reconstruction of a particle are hits (calibrated, above threshold, …)
- Hits can have very different properties depending on the sub detector
- Unordered, not dense, … → GNNs

# Which GNN for reconstruction

- In a big detector, there is a large number of hits >100k
- There is not graph a-priori
- Is DGCNN a good idea?

$$y_j = \max_{i \epsilon N(j)} MLP(x_j, x_j - x_i)$$

?

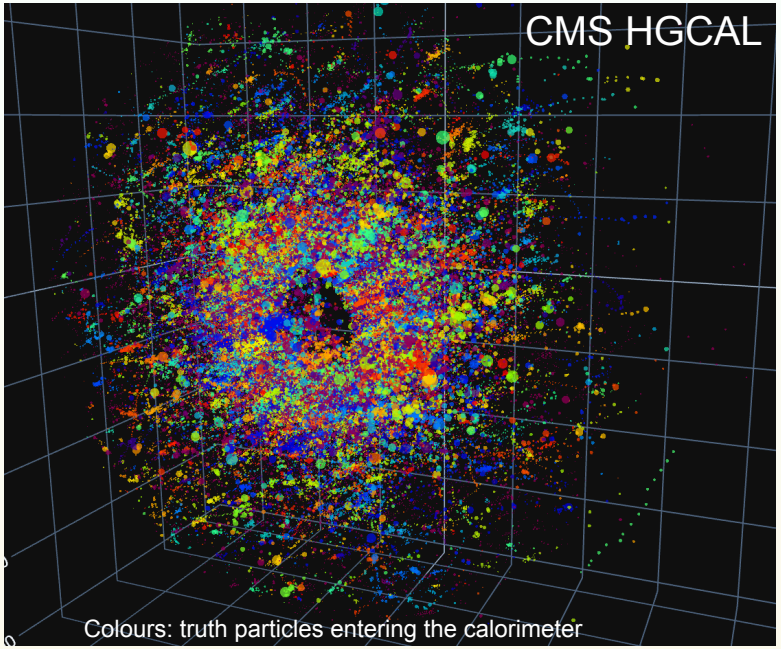


# Which GNN for reconstruction

- In a big detector, there is a large number of hits >100k
- There is not graph a-priori
- Is DGCNN a good idea?

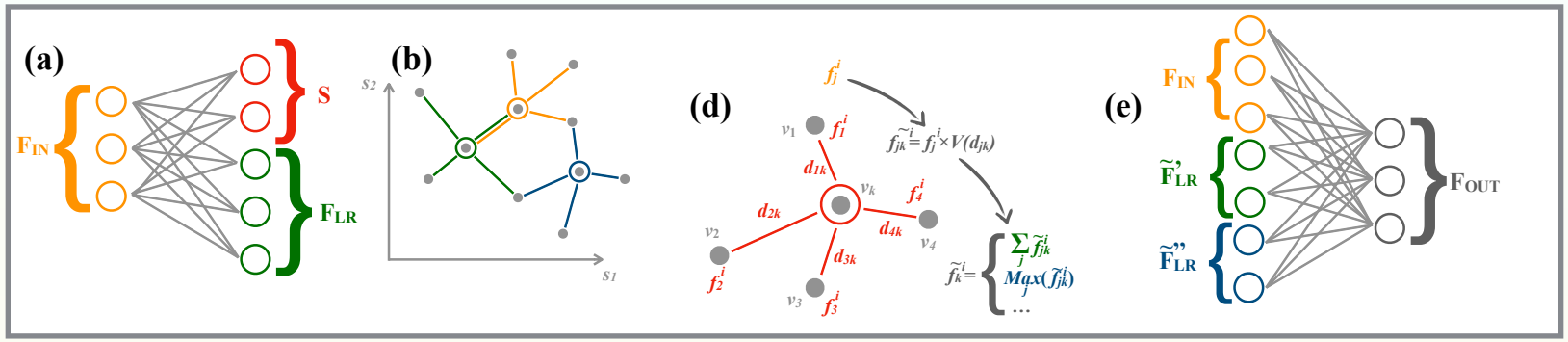
$$y_j = \max_{i \epsilon N(j)} MLP(x_j, x_j - x_i)$$

?




- No: Multiple operations scale with N x K x F
- Find a less resource-intense architecture
- Implement the graph topology building more direct *

# GravNet: a faster EdgeConv?




- Instead of generic message passing, use powerful attention

$$y_j = \square_{i \epsilon N(j)} \; a(x_j, x_i) \cdot \text{MLP}(x_i)$$

Does not depend on $j$

EdgeConv

$$y_j = \max_{i \epsilon N(j)} MLP(x_j, x_j - x_i)$$

Per edge MLP: N x K x F

- Can be rewritten

$$\tilde{x}_i = \text{MLP}(x_i)$$

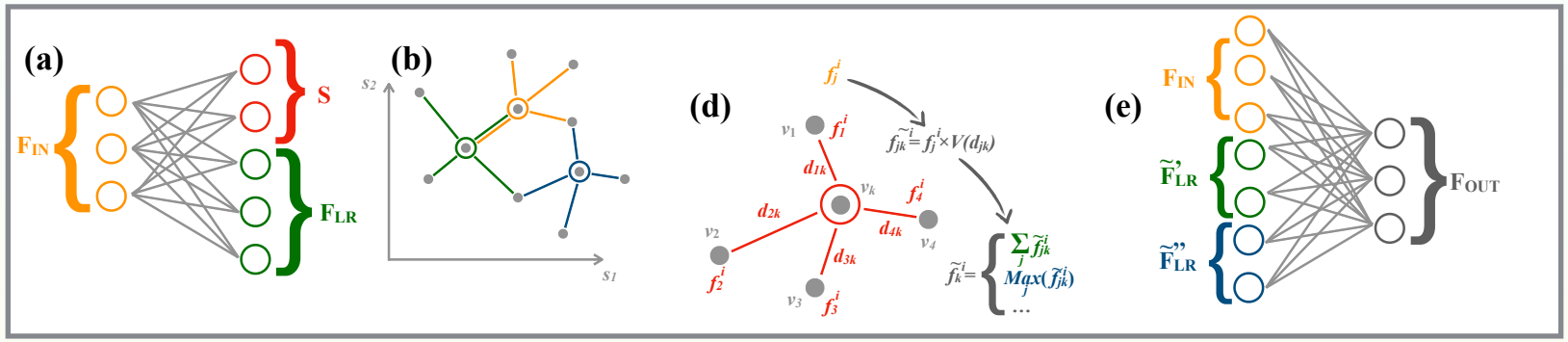
Per-vertex MLP: N x F

$$y_j = \square_{i \epsilon N(j)} \; a(x_j, x_i) \cdot \tilde{x}_i$$

now ETP

arXiv:1902.07987

# Building the graph topology




- Attention scales the information exchange: higher value → more exchange

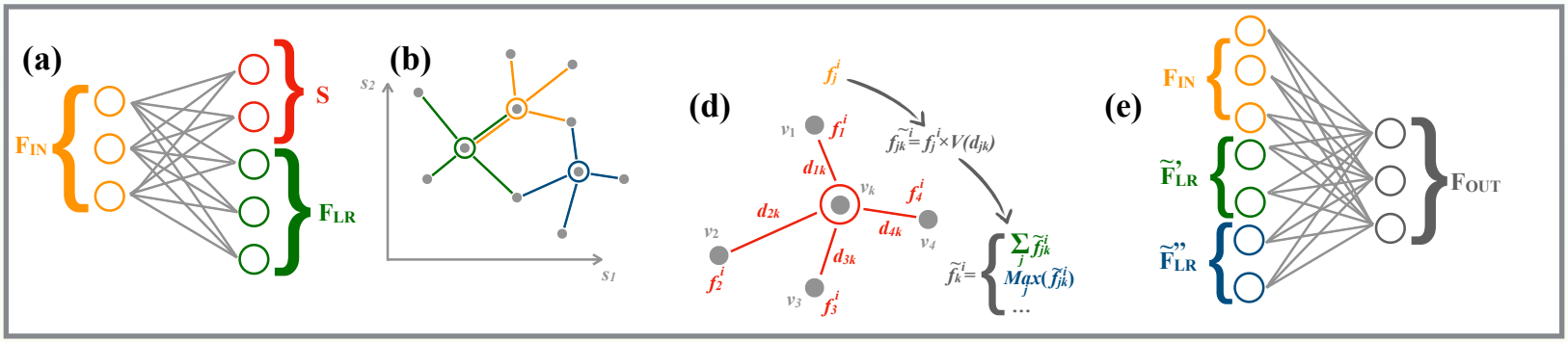
$$y_j = \square_{i \epsilon N(j)} \; a(x_j, x_i) \cdot \tilde{x}_i$$

- Learn an embedding space for each point
- Chose $a(x_i, x_j)$ such that close-by means a lot of information exchange:

$$a(x_i, x_j) = \exp(-d_{ij}^2/10)$$

- The embedding space $S$ will be a representation of the attention between points

# Building the graph topology




- Close-by: a lot of attention → 'best' K neighbours are the nearest ones!

$$y_j = \square_{i \epsilon N(j)} \; a(||x_j^S - x_i^S||) \cdot \tilde{x}_i$$

Nearest neighbours

now ETP
arXiv:1902.07987

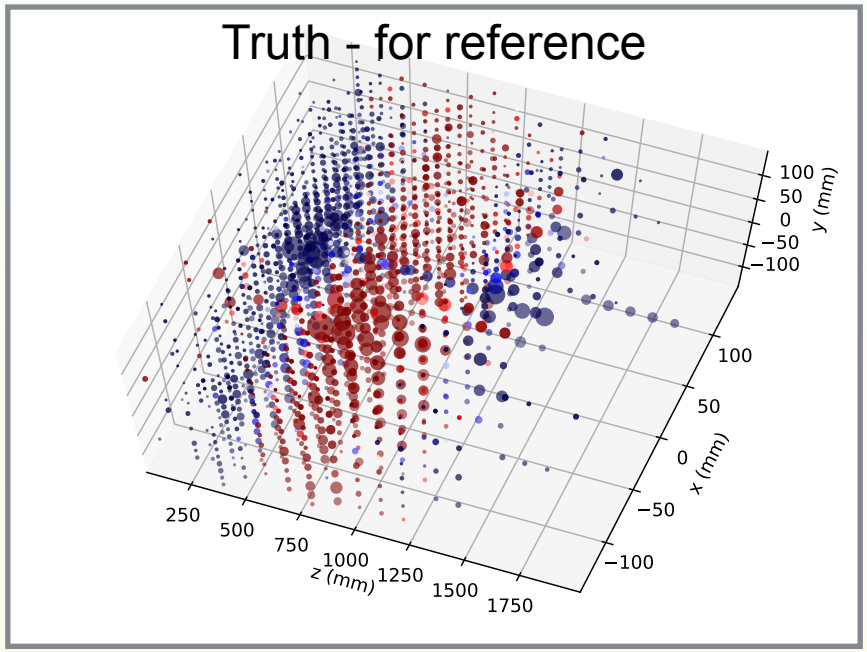
- We have build a graph topology that makes sure information is passed optimally between vertices

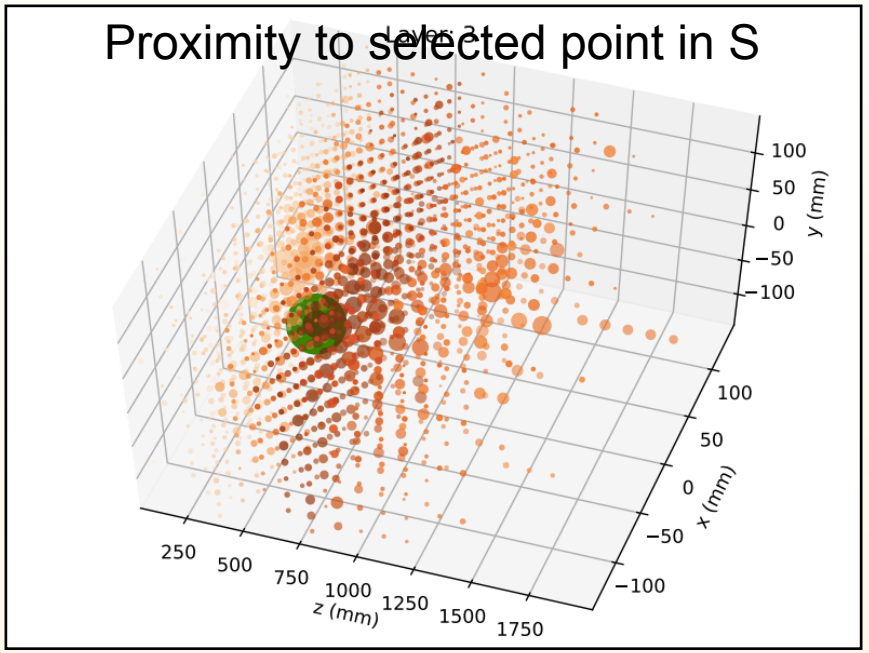
- S can have a few dimensions → safe a factor of ~10 in resources

DGCNN: dim(F) = 64

GravNet: dim(S) ~ 6

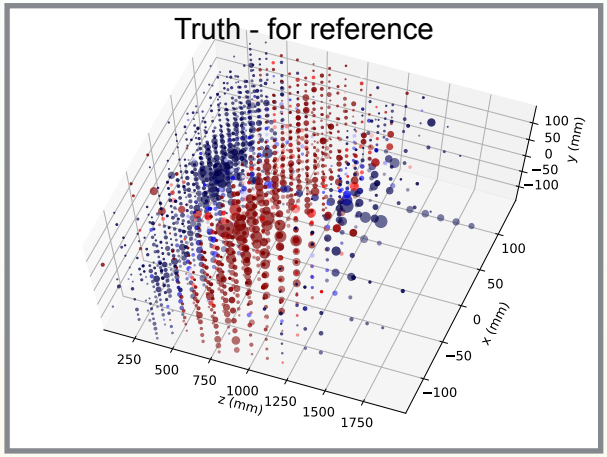
# Interpretability and examples

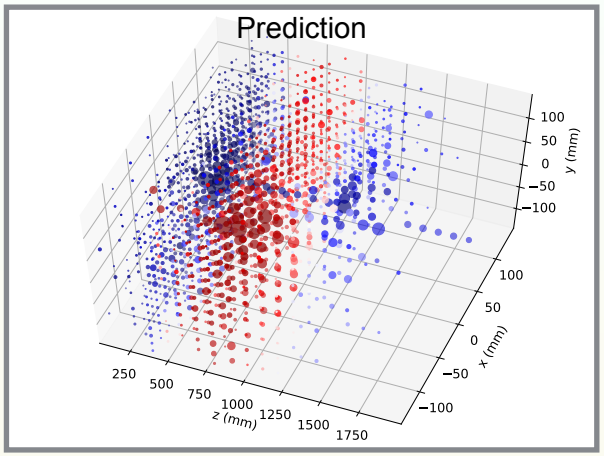


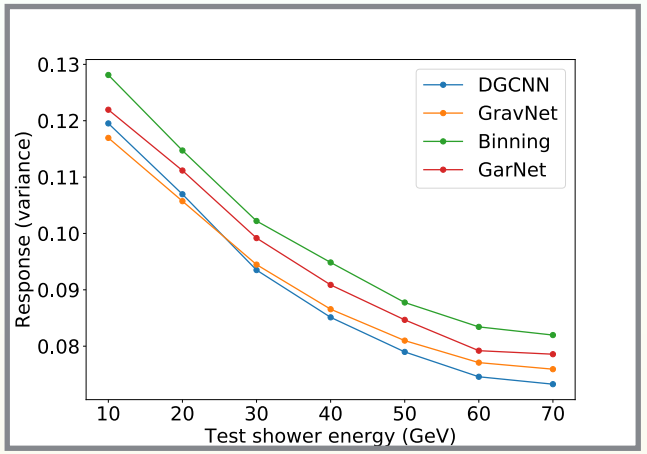




- Adds a handle for interpretation to the network

# Performance




- Compare layers embedded in very similar architectures for toy calorimeter
- The graph network architectures outperform the CNN approach
- Similar performance but lower resource requirements of GravNet versus DGCNN

- Also confirmed higher performance for separation of hits from different particles than DGCNN [2]

[2] ATL-PHYS-PUB-2021-002

# How to reconstruct multiple objects (very brief)

- Two particles leave hits

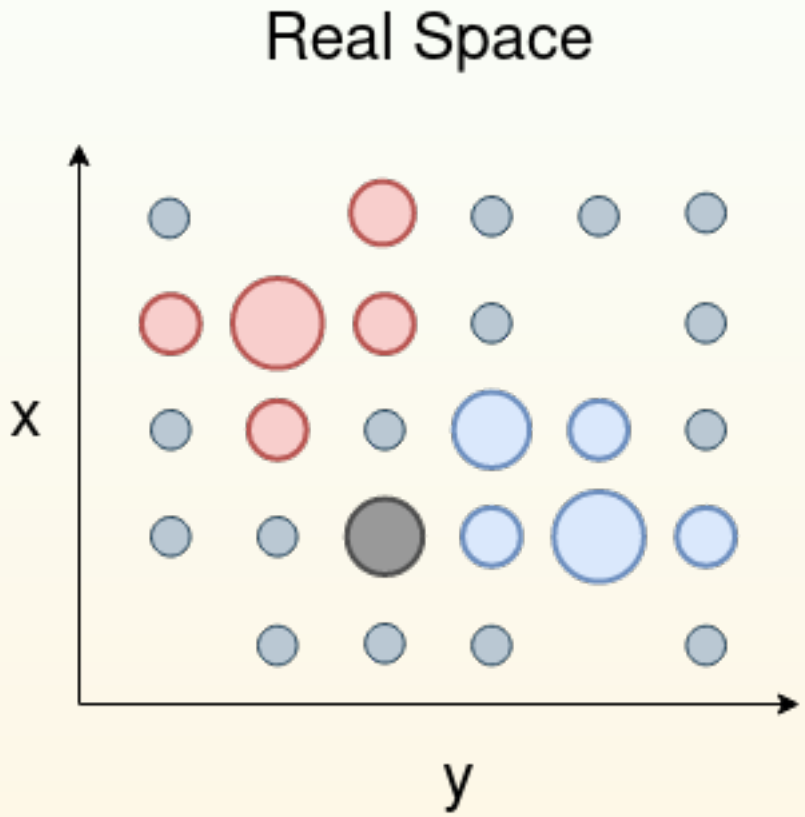
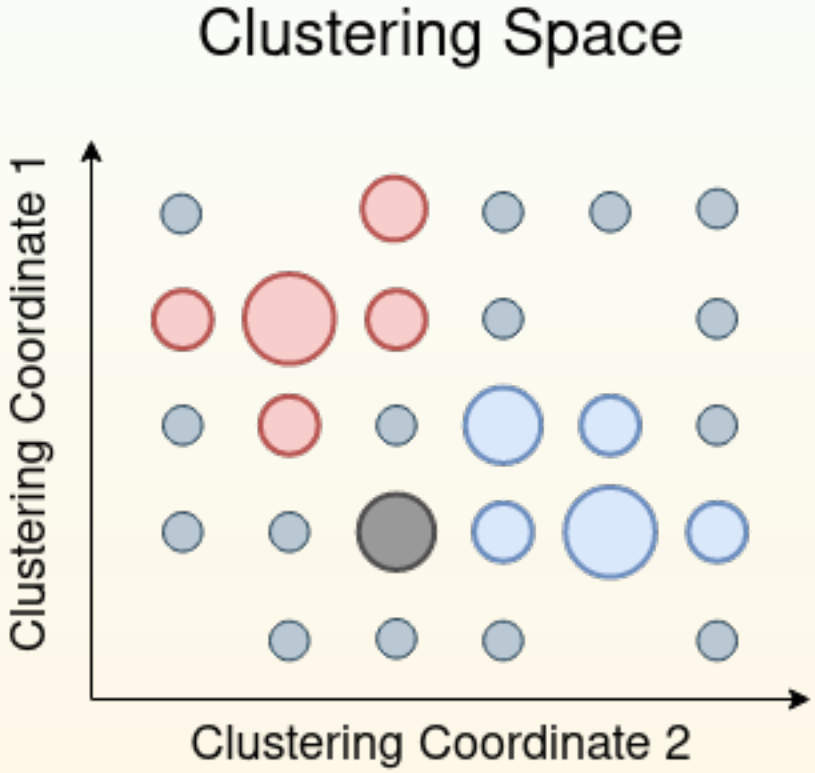



Image credit: Lea Reuter, Isabel Haide

# How to reconstruct multiple objects (very brief)

- Create a clustering space (with a GNN), transform input features $x_c = \text{GravNet}(x_{in})$




Full disclosure

Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph, and image data

Jan Kieseler[1]
(jan.kieseler@cern.ch)
CERN, Experimental Physics Department, Geneva, Switzerland

now ETP

arxiv:2002.03605

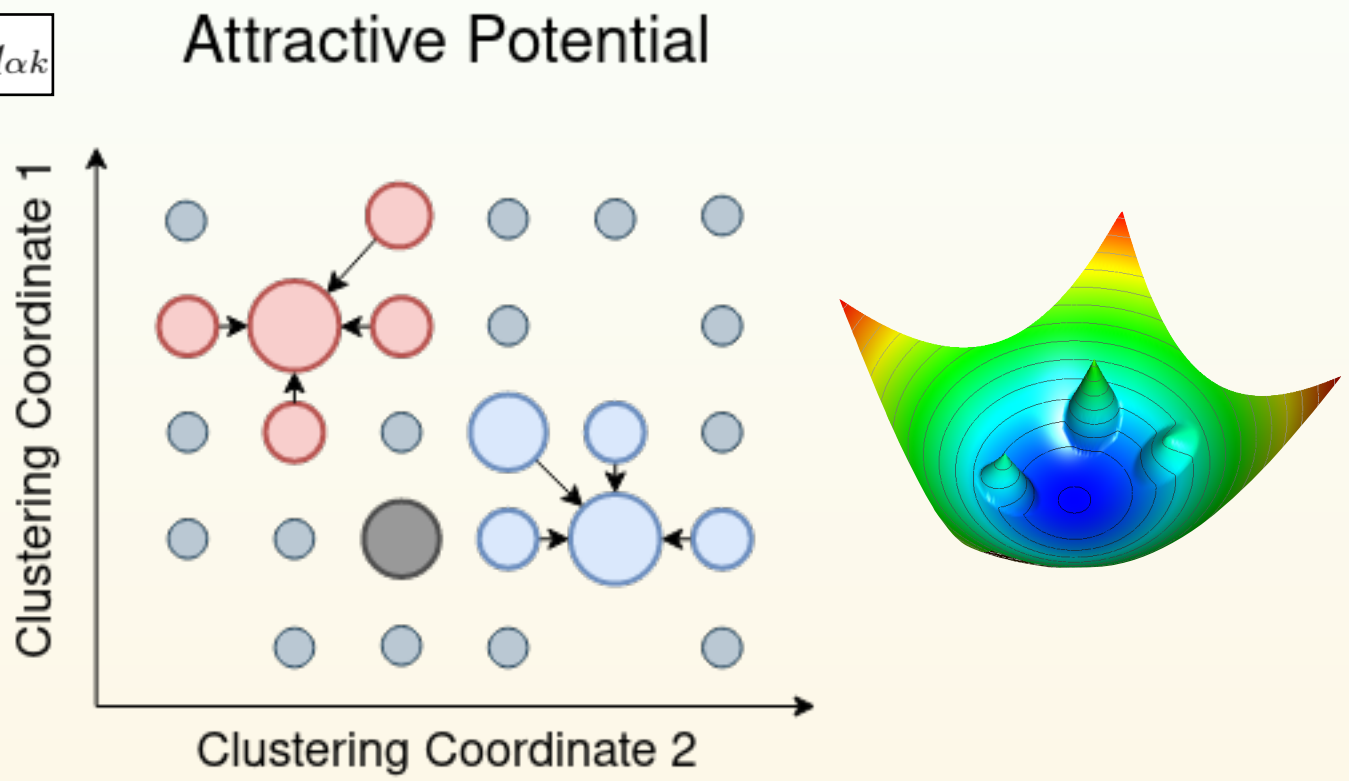
ETP

Image credit: Lea Reuter, Isabel Haide

# Make the network learn to reconstruct objects

- Write a loss function (Object Condensation) that attracts points from same object

$$\breve{V}_k(x) = ||x - x_\alpha||^2 q_{\alpha k}$$




now ETP
arxiv:2002.03605

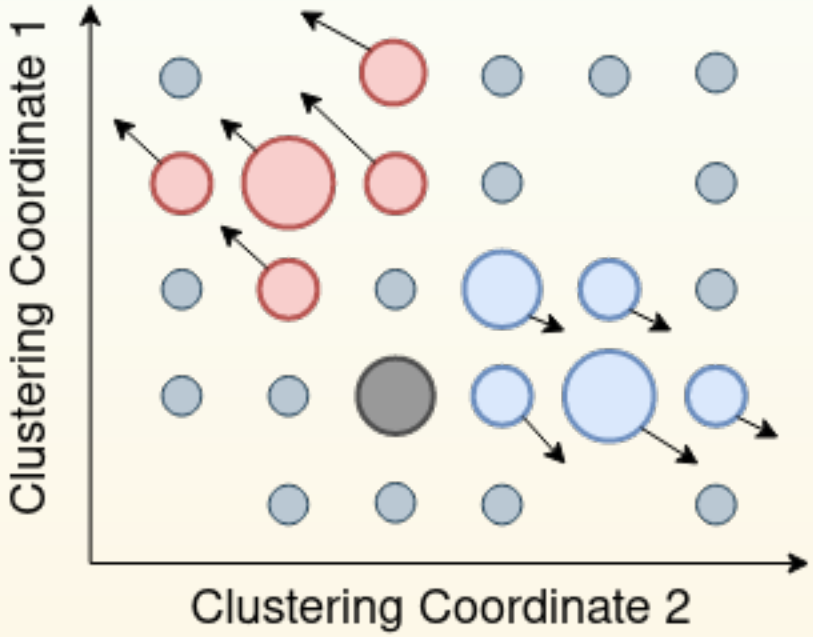
ETP
Image credit: Lea Reuter, Isabel Haide

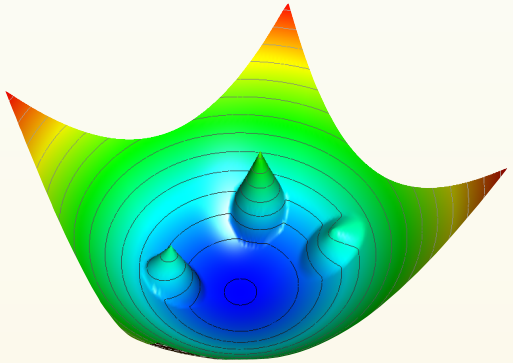
# Separate different objects

- Write a loss function (Object Condensation) that attracts points from same object

$$\hat{V}_k(x) = \max(0, 1 - ||x - x_\alpha||)q_{\alpha k}$$

Repulsive Potential



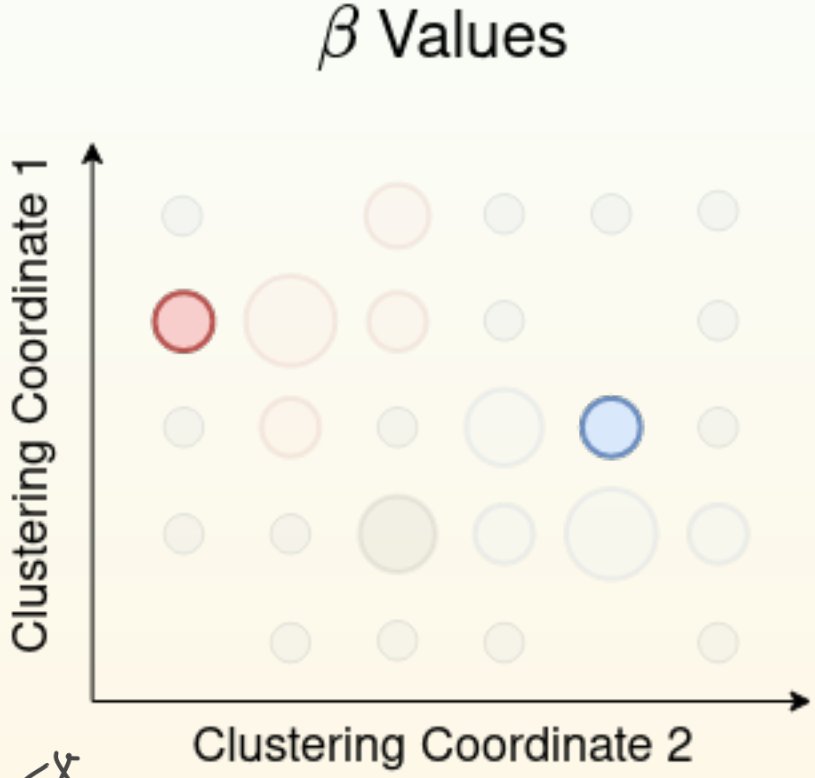



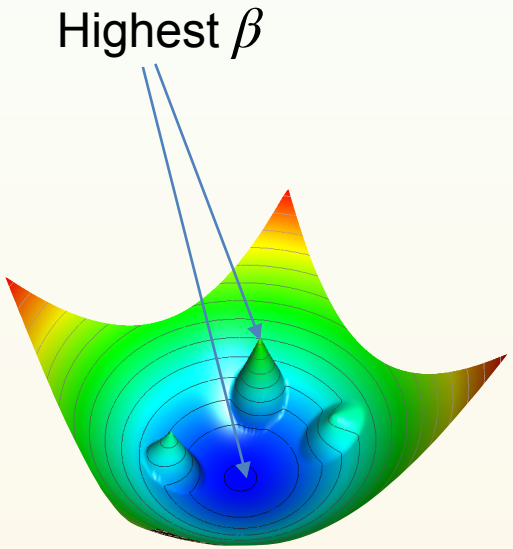
now ETP
arxiv:2002.03605

ETP
Image credit: Lea Reuter, Isabel Haide

# Learn a point representing the object

- Details are in backup
- We can also assign properties to these representative points



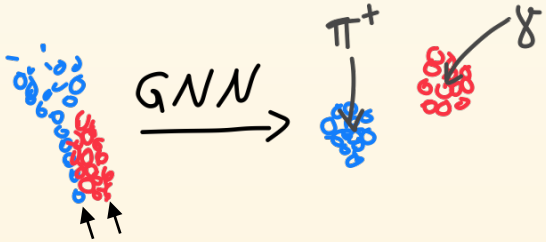
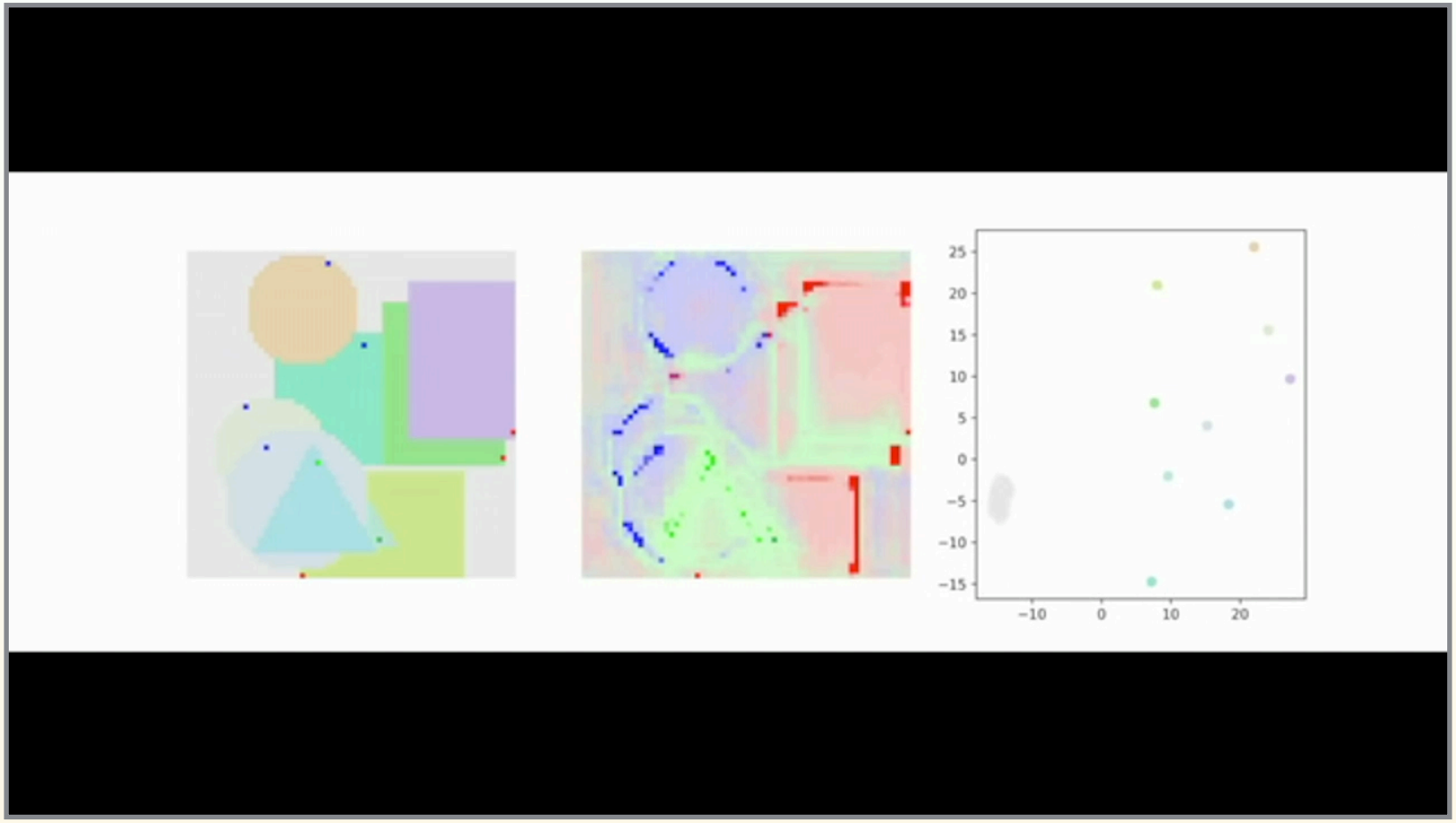



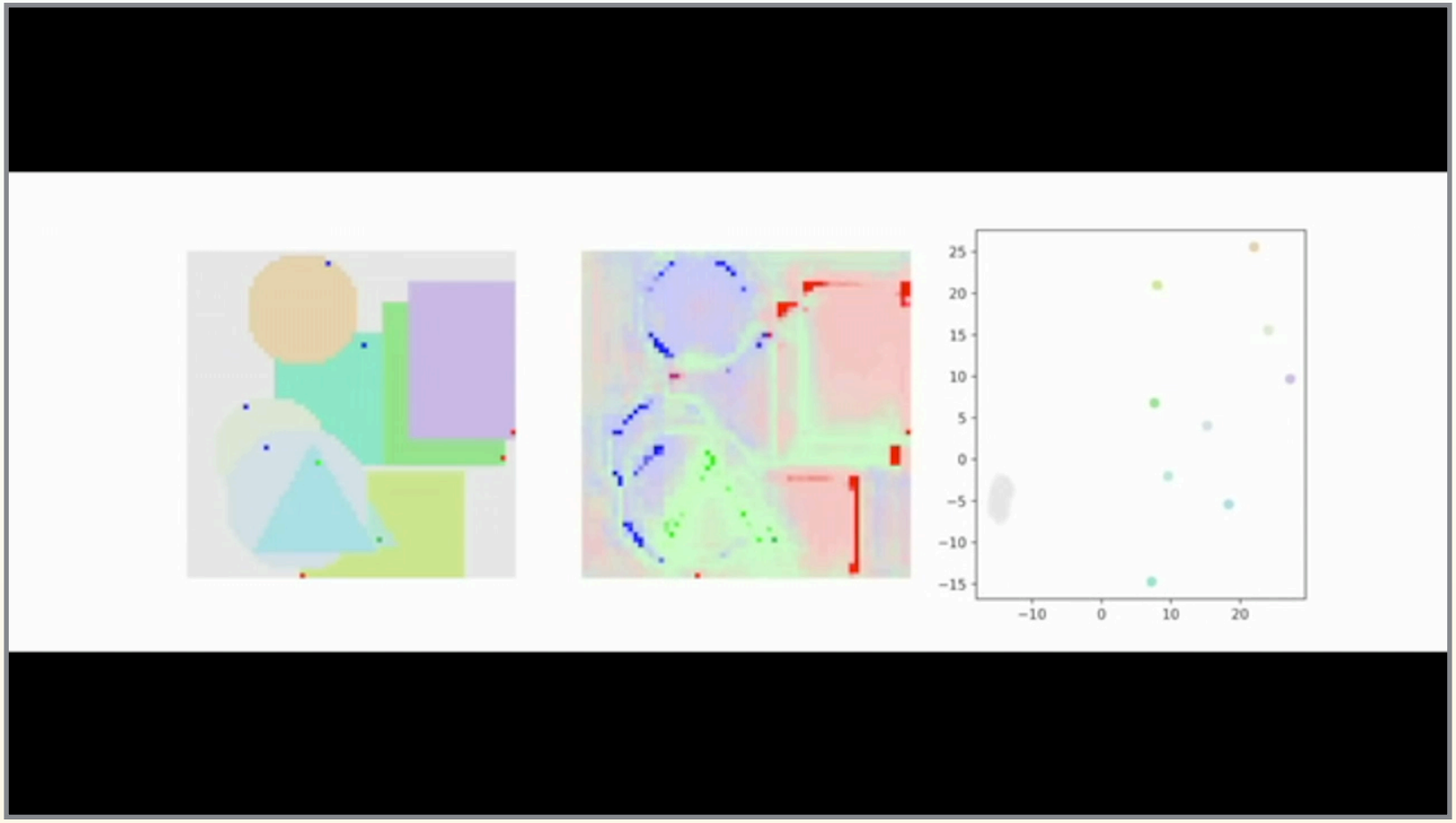

Image credit: Lea Reuter, Isabel Haide

# Example on an image



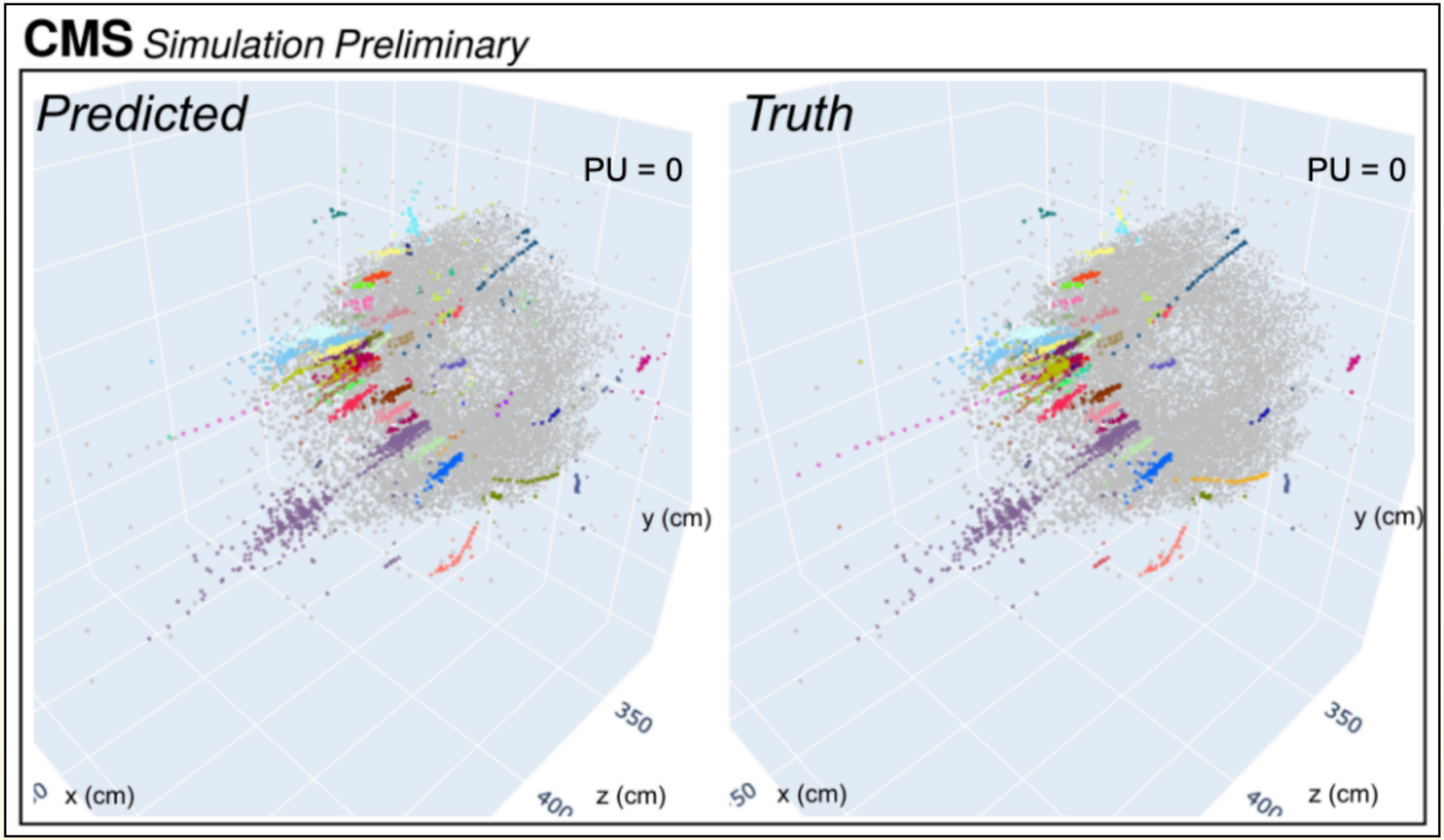

- Points are separated and assigned a property
- Now, objects can be picked up (clustered) easily

# Example on an image




- Points are separated and assigned a property
- Now, objects can be picked up (clustered) easily

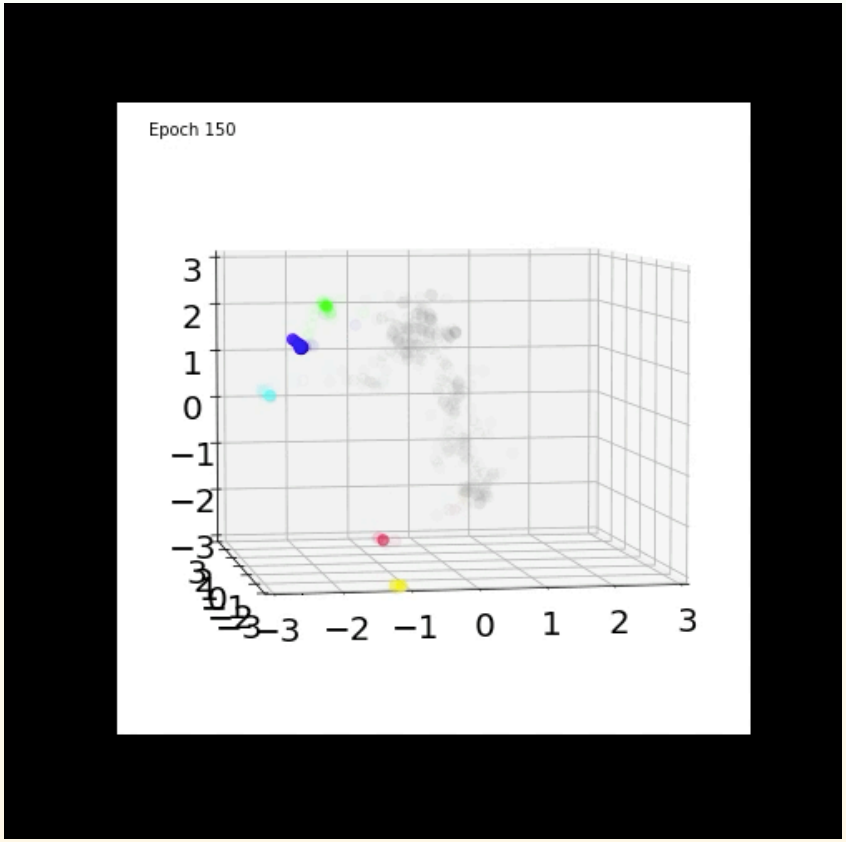
# In action in CMS



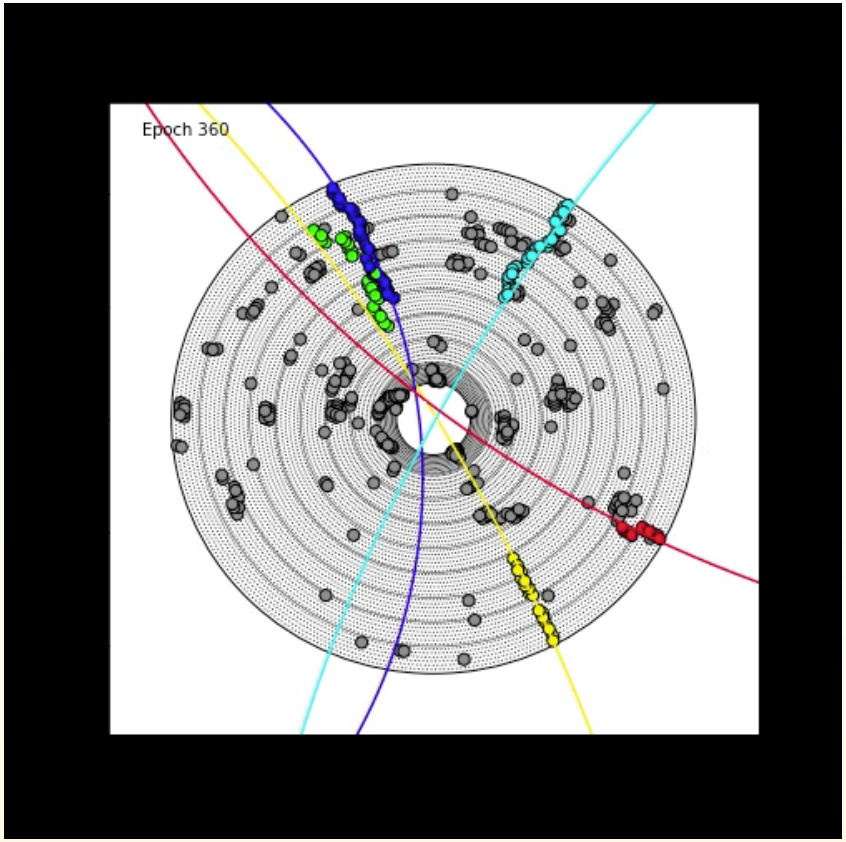

- Excellent shower separation

# In action at Belle2

Unfortunately not in sync




Clustering space

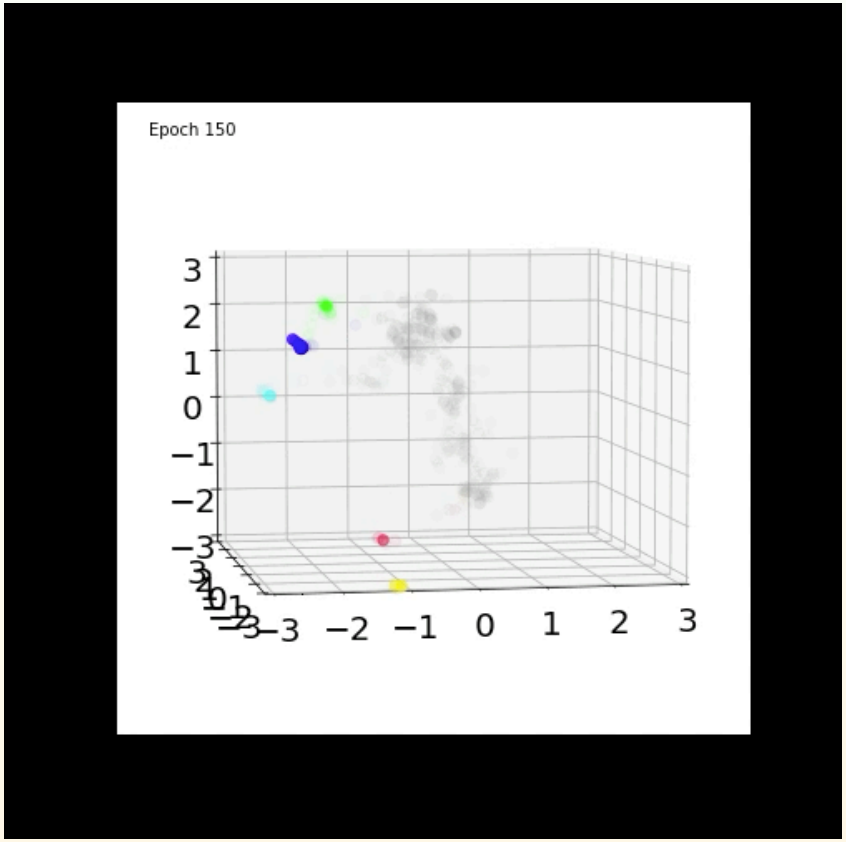



Detector hits and reconstructed tracks

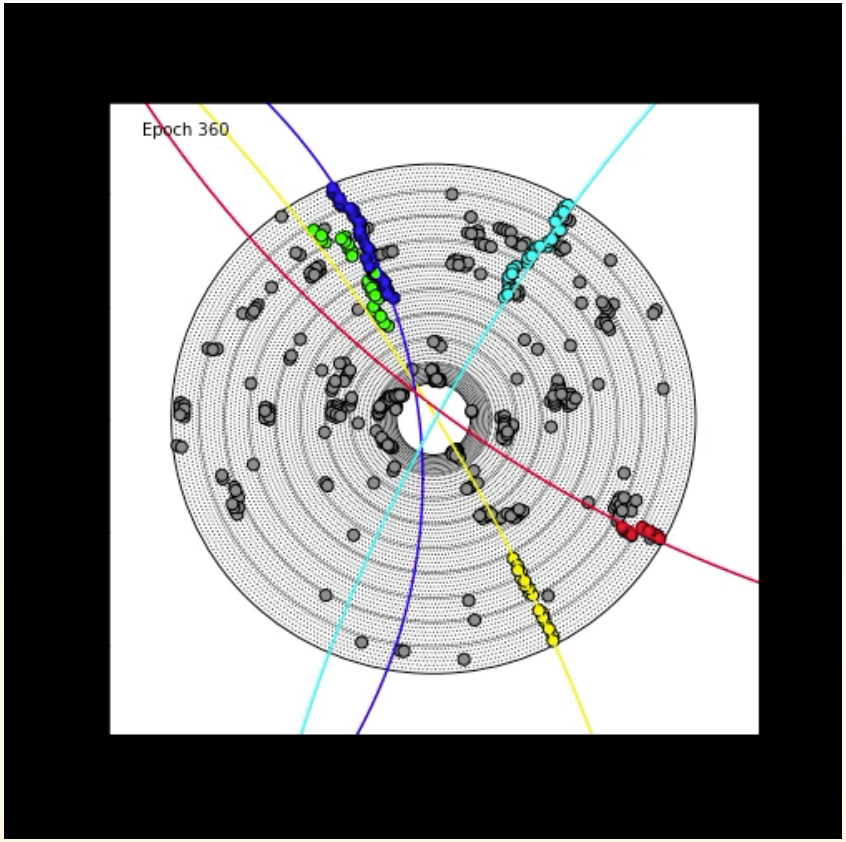
Image credit: Lea Reuter

# In action at Belle2

Unfortunately not in sync




Clustering space



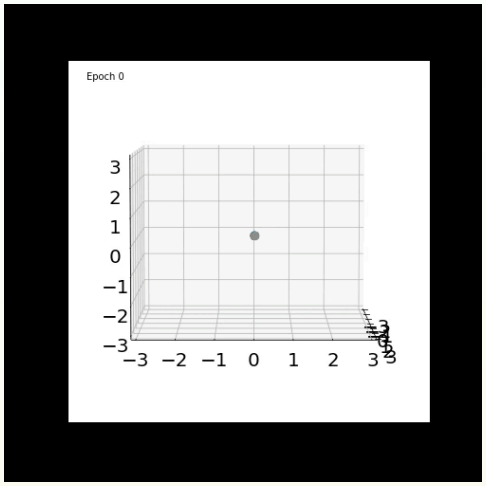

Detector hits and reconstructed tracks

Image credit: Lea Reuter

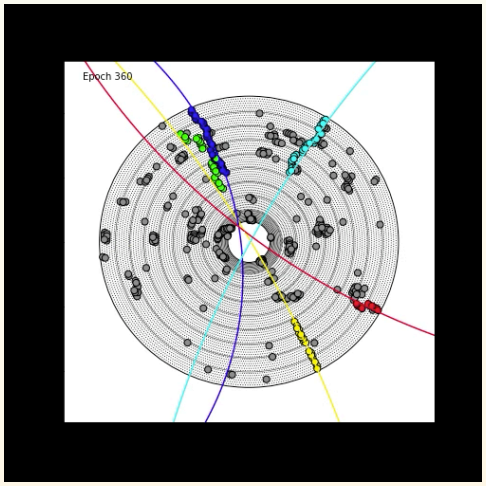
# Summary

- Physics is exciting! Machine learning too!
  Together they are even more exciting

- Machine learning can be used at all levels of data analysis
  - Final classifier
  - Improve high-level objects / variables
  - Reconstruct from processed detector signals
  - Directly analyse the raw signals




- We went through a large toolbox of NN architectures to do this
  - MLPs
  - CNNs
  - Attention / Transformers
  - GNNs



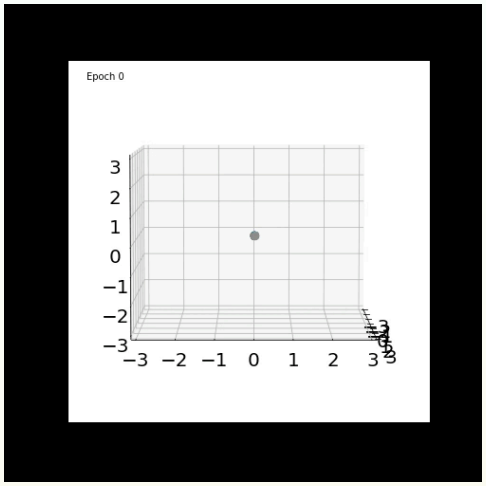

- We discussed handles to interpret the models

- Always keep in mind: understanding the structure of the data (aka “the physics”) and smart solutions matter most

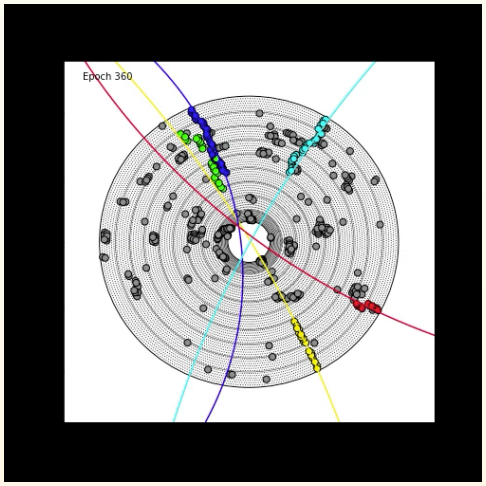
# Summary

- Physics is exciting! Machine learning too!
  Together they are even more exciting

- Machine learning can be used at all levels of data analysis
  - Final classifier
  - Improve high-level objects / variables
  - Reconstruct from processed detector signals
  - Directly analyse the raw signals

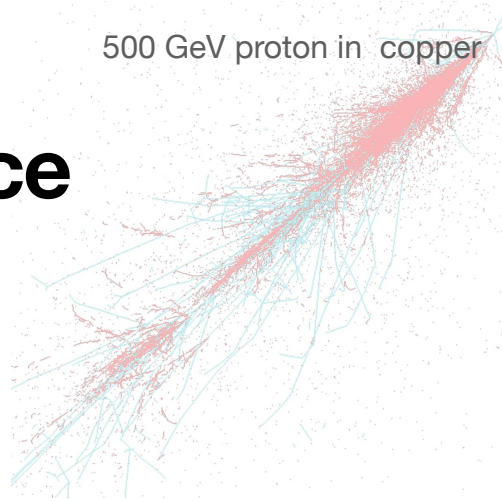



- We went through a large toolbox of NN architectures to do this
  - MLPs
  - CNNs
  - Attention / Transformers
  - GNNs




- We discussed handles to interpret the models

- Always keep in mind: understanding the structure of the data (aka “the physics”) and smart solutions matter most

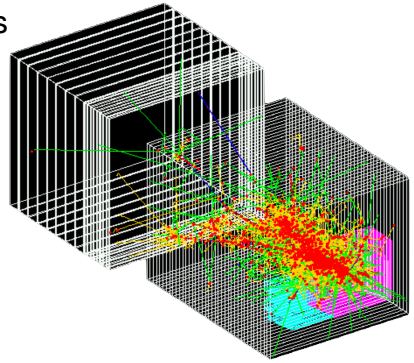


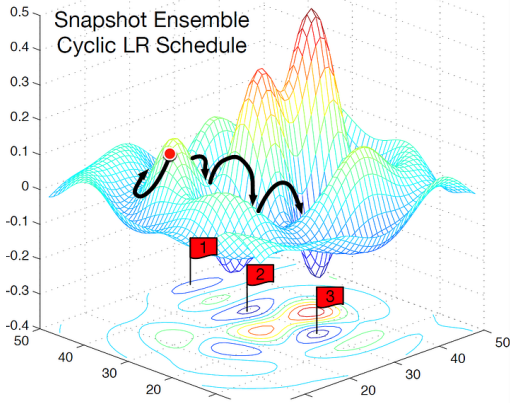

# Blockpraktikum: ETP Data Science

Institut für Experimentelle Teilchenphysik

## Neural networks for calorimetry

- What to expect:
  - Dive into calorimeter physics
  - Hands-on simulation of your own calorimeter designs with state-of-the art software
  - Hands-on course on building blocks for neural networks and their implementations
  - Application of advanced neural networks to particle energy reconstruction in calorimeters






- October 9th — 14th
- Please register well in advance:

  https://plus.campus.kit.edu/signmeup/procedures/954

Prof. Dr. Ferber, Prof. Dr. Klute, Dr. Kieseler

# Feedback

https://cern.ch/mmda

Please help improve the lectures

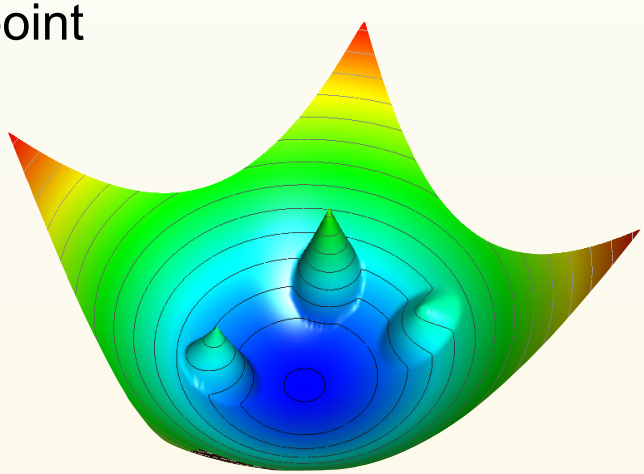
# Backup

# Object condensation in short

$$\breve{V}_k(x) = ||x - x_\alpha||^2 q_{\alpha k}, \text{ and}$$

$$\hat{V}_k(x) = \max(0, 1 - ||x - x_\alpha||) q_{\alpha k}.$$

- Maximum β/charge vertices *are* center points *
- Encourage network to select one representative point per object k and make it have a high $\beta$ value



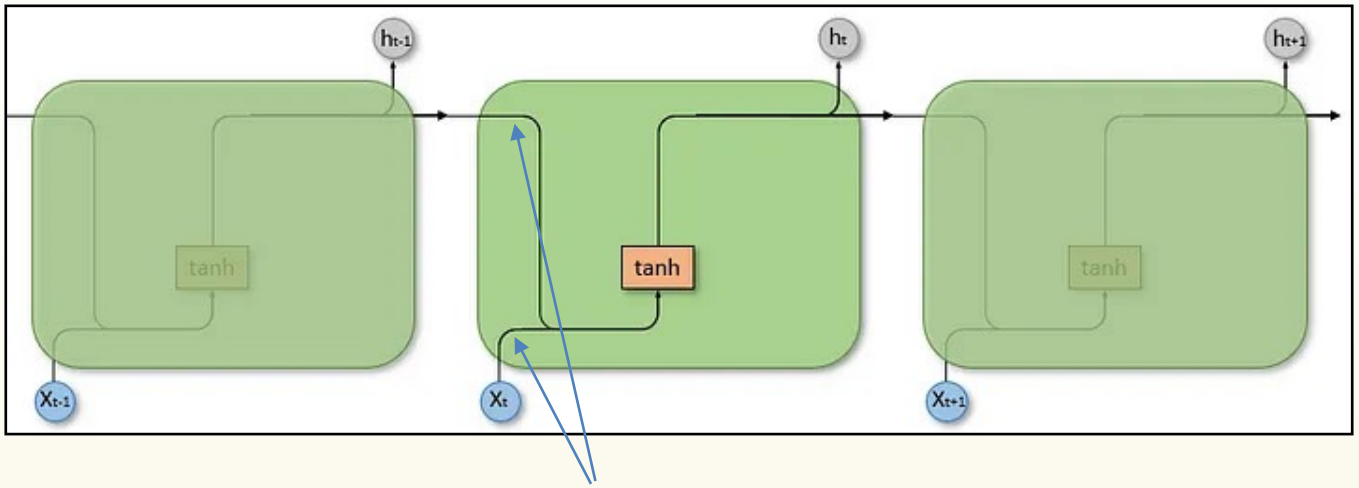
$$L_\beta = \frac{1}{K} \sum_k (1 - \beta_{\alpha k}) + s_B \frac{1}{N_B} \sum_i^N n_i \beta_i,$$

- Also weight object property loss with β

$$L_p = \frac{1}{\sum_{i=0}^N (1 - n_i) \operatorname{arctanh}^2 \beta_i} \sum_{i=0}^N L(t_i, p_i)(1 - n_i) \operatorname{arctanh}^2 \beta_i$$

- *Condensation points will carry all object properties*
- *Very natural approach for dynamic graph NN*

# Recurrent neural network in a (very small) nutshell




Apply learnable weights

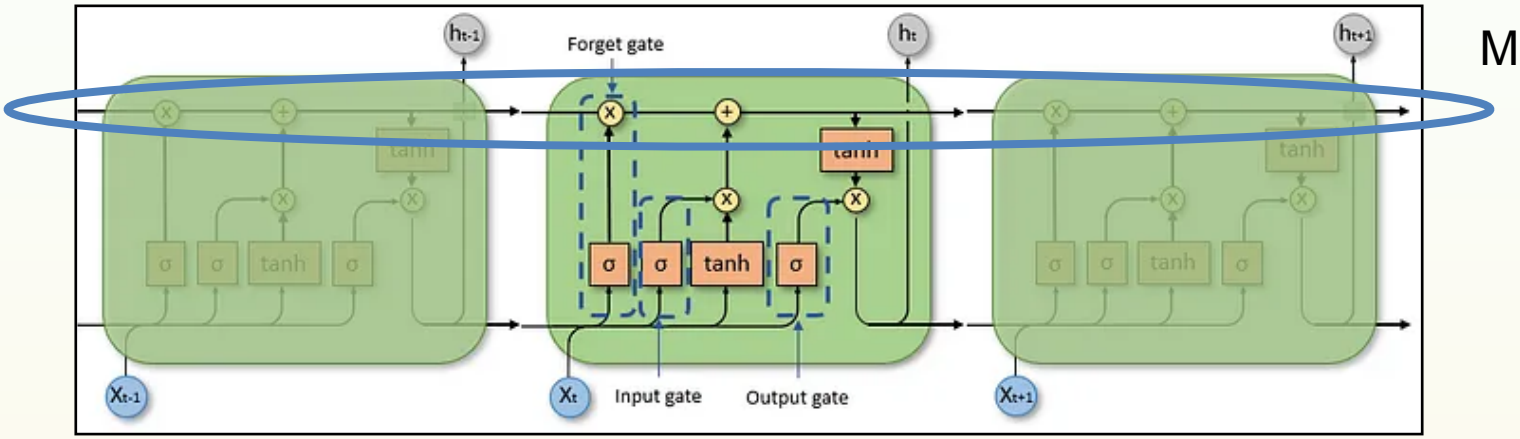
$$h_t = \tanh(\omega_x x_t + \omega_h h_{t-1} + T)$$

Re-use output of previous iteration: sequence

- Can iterate over the sequence
- Each iteration has one input and one output
- Outputs $h_t$ are affected by inputs $x_t$ but also by $x_{t-m}$
  - But vanishes quickly for larger m: short term memory

# Long-short-term memory: just the concept

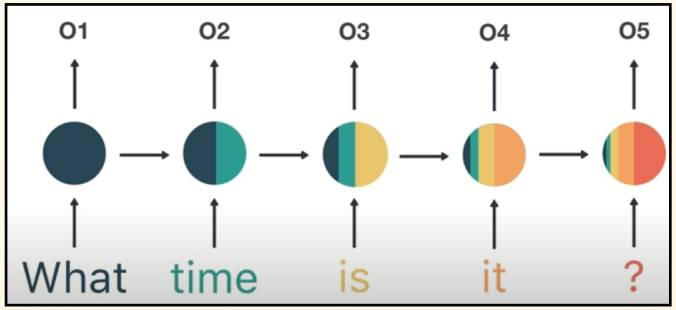
https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4



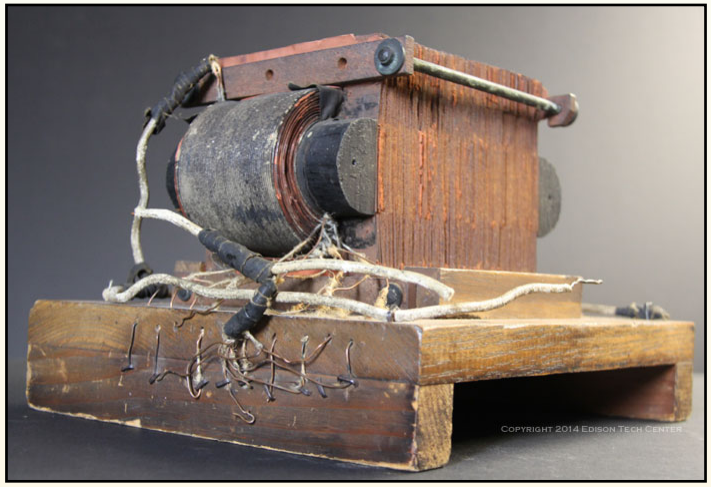

- Add another hidden state: the *cell* state (long term memory)
  - Update (forget and add information) in a learnable way

- Quite complex, won't go into detail here

- Main point $h_t = h_t(x_t, h_{t-1}, m_t)$
  **Output is a function of previous inputs and current input**



# The other approach to attention



# Evolution on the example of image to text

Cat lying on couch

Person wearing hat

- Image: CNN ✓
- What is the structure of text?

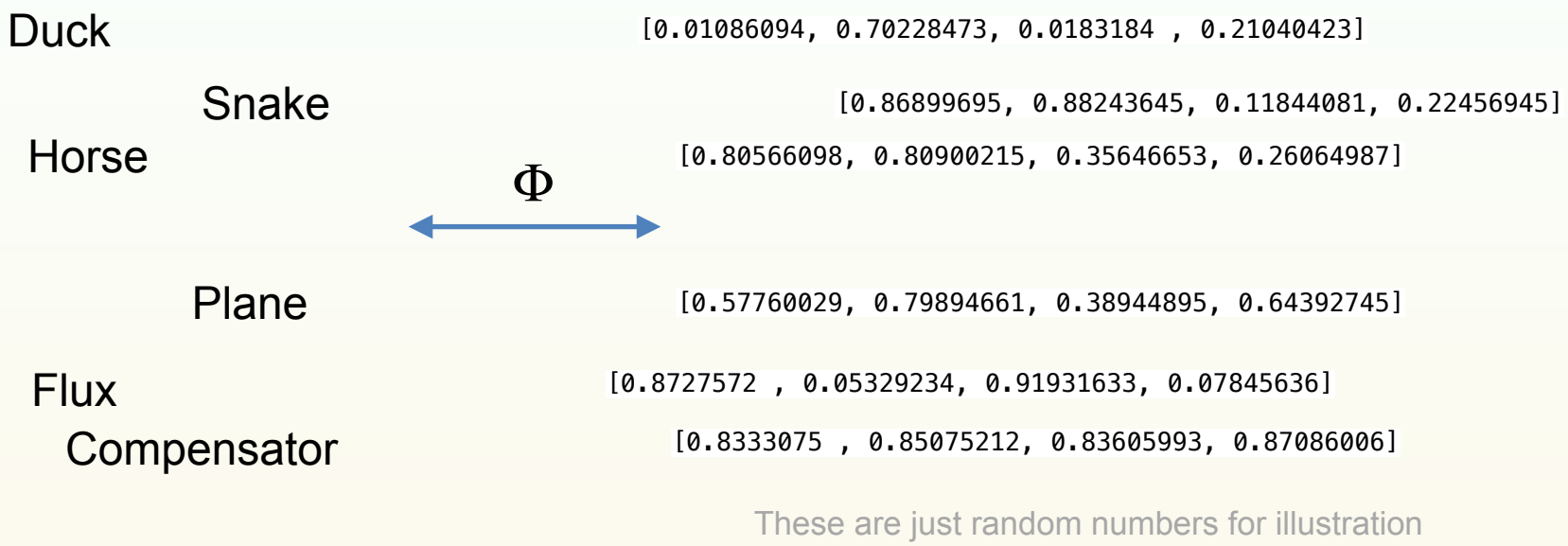
# Evolution on the example of image to text

Cat lying on couch

Person wearing hat

- Image: CNN ✓
- What is the structure of text?
  - Order matters: *sequence*
  - Word relations matter: *context*
  - Length is a-priori undefined

# Embeddings

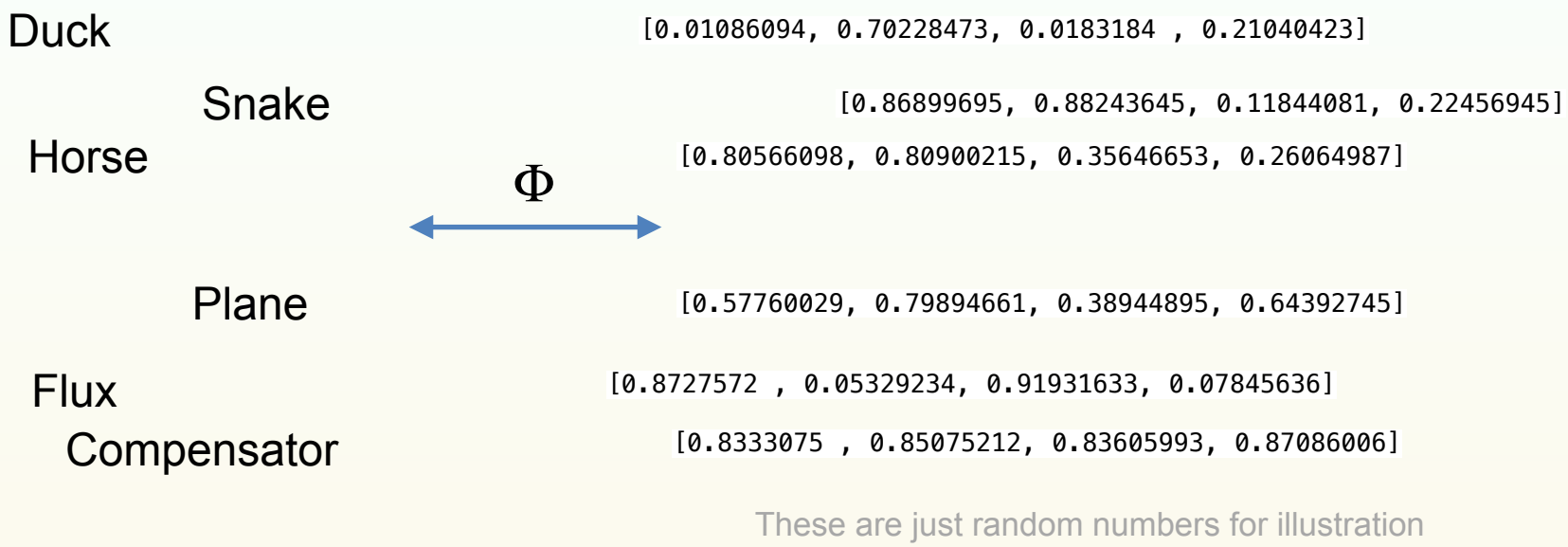



- Translate each human-readable word into an *embedding* vector

Used a lot in ML

- The easiest way to do that in a way optimal to the task?

# Embeddings




- Translate each human-readable word into an *embedding* vector

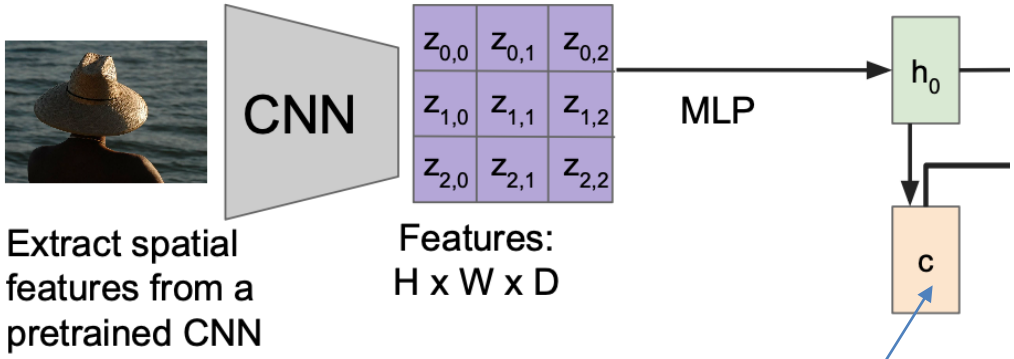
Used a lot in ML

- The easiest way to do that in a way optimal to the task?
  - Train a DNN to do it. (together with the 'main' model)
  - Words are now simple vectors ✓

# A recurrent neural network approach

- We can generate a sequence (of words) using this mechanism

**Encoder**: $h_0 = f_W(\mathbf{z})$
where **z** is spatial CNN features
$f_W(.)$ is an MLP

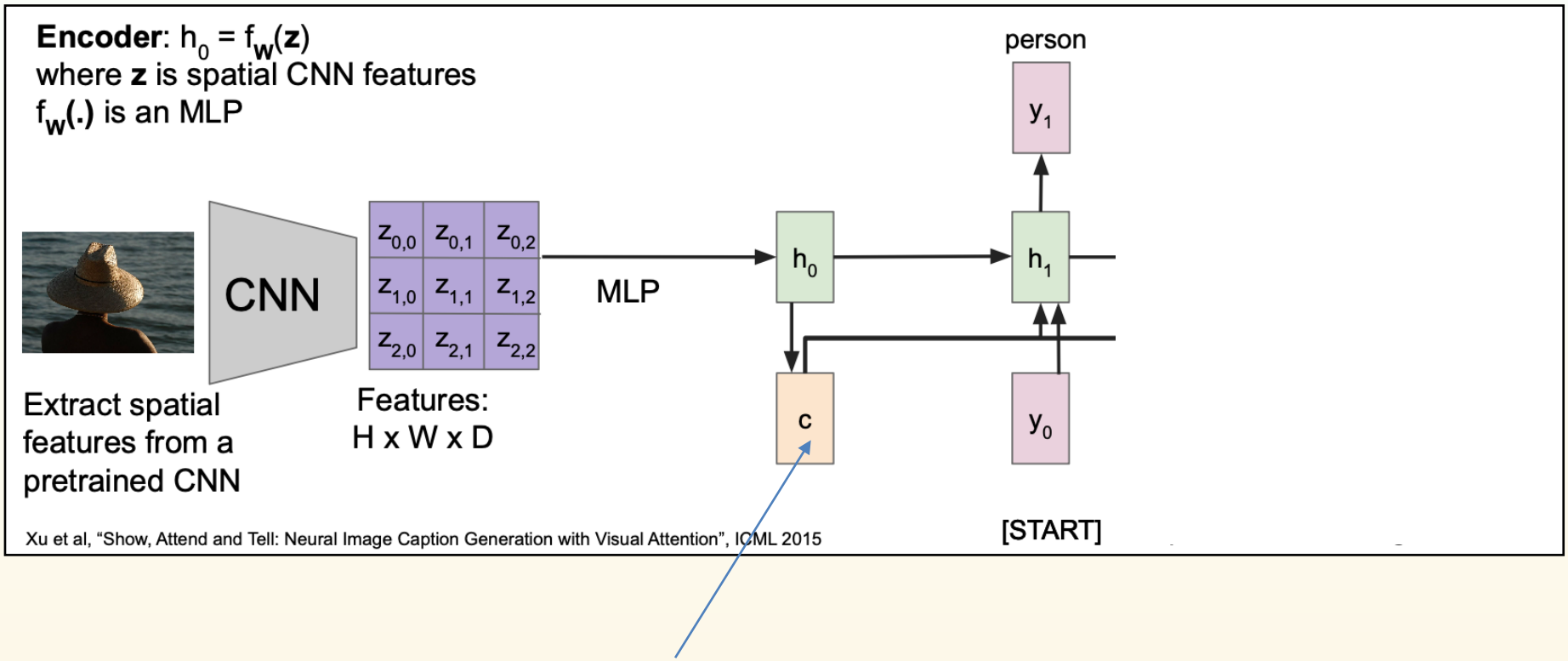



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Context vector, has to contain the full information of the image

# A recurrent neural network approach

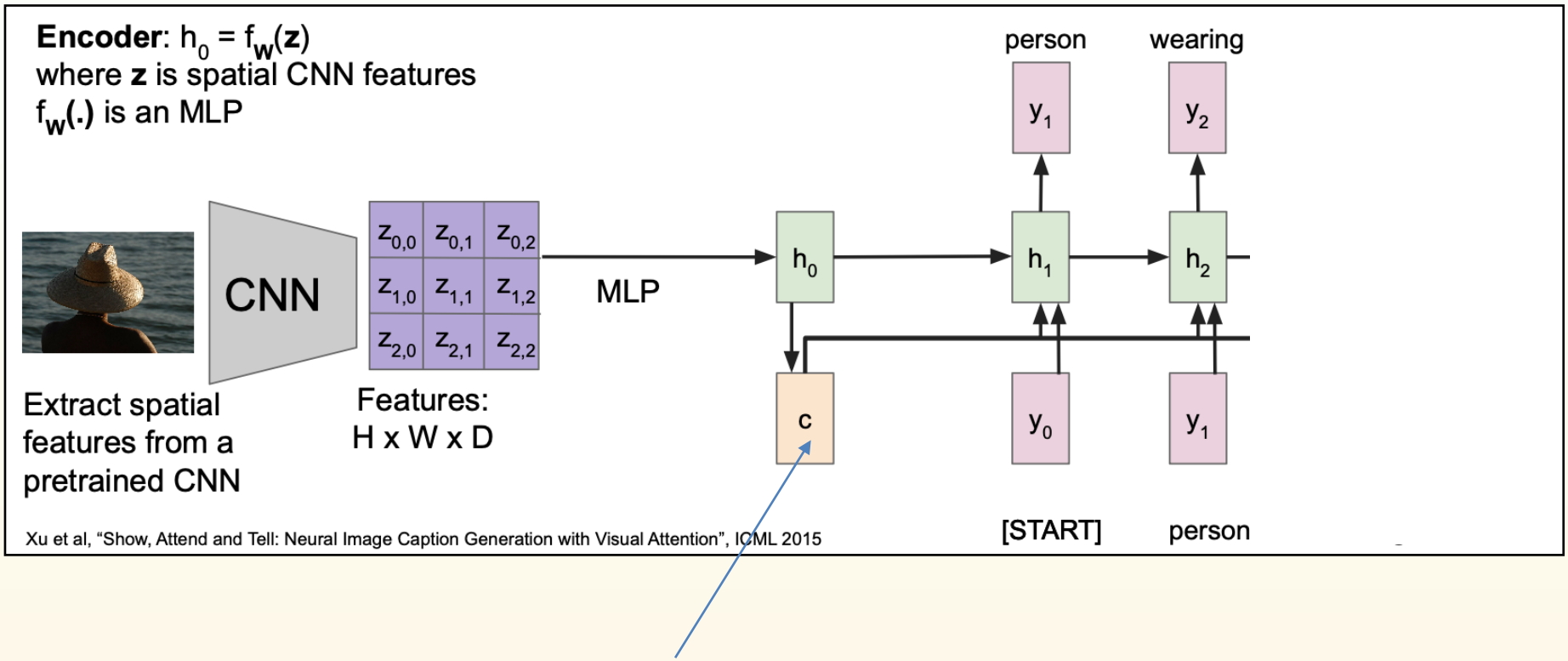
- We can generate a sequence (of words) using this mechanism




Context vector, has to contain the full information of the image

# A recurrent neural network approach

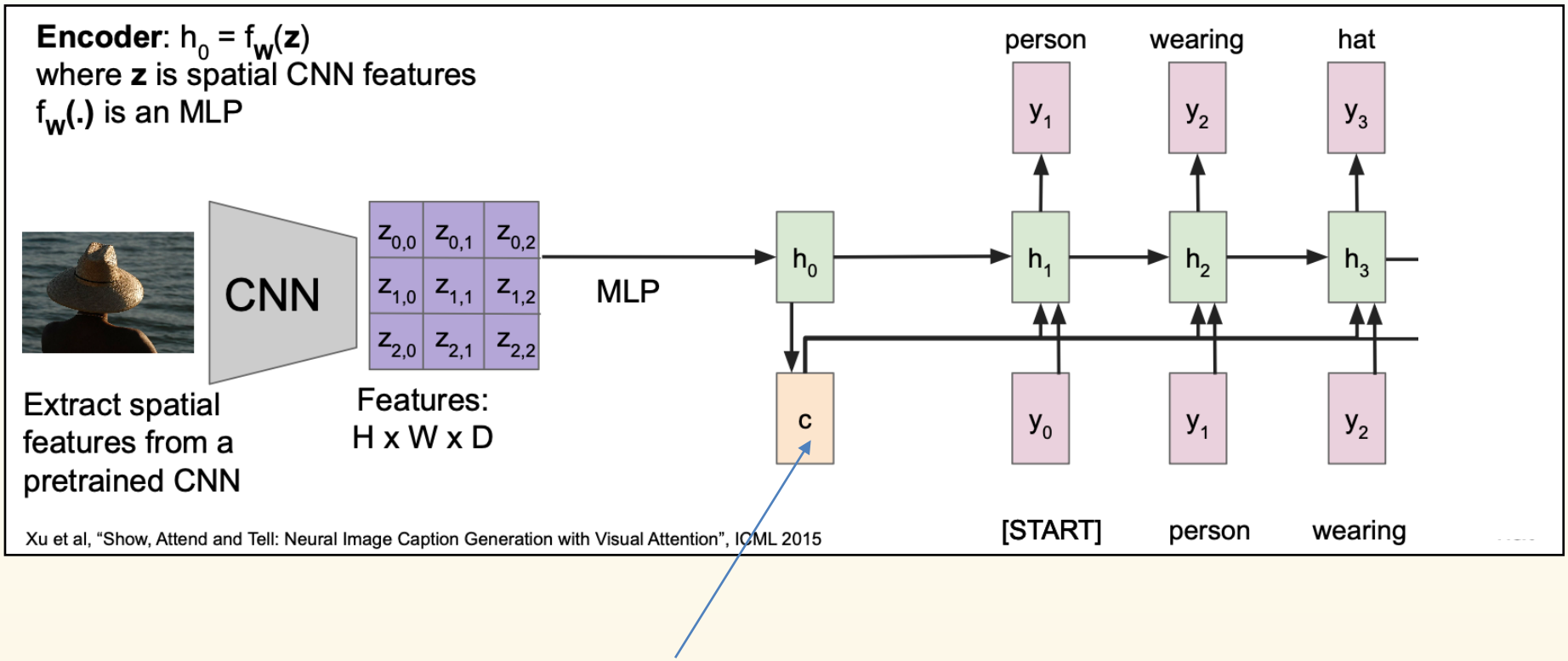
- We can generate a sequence (of words) using this mechanism




Context vector, has to contain the full information of the image

# A recurrent neural network approach

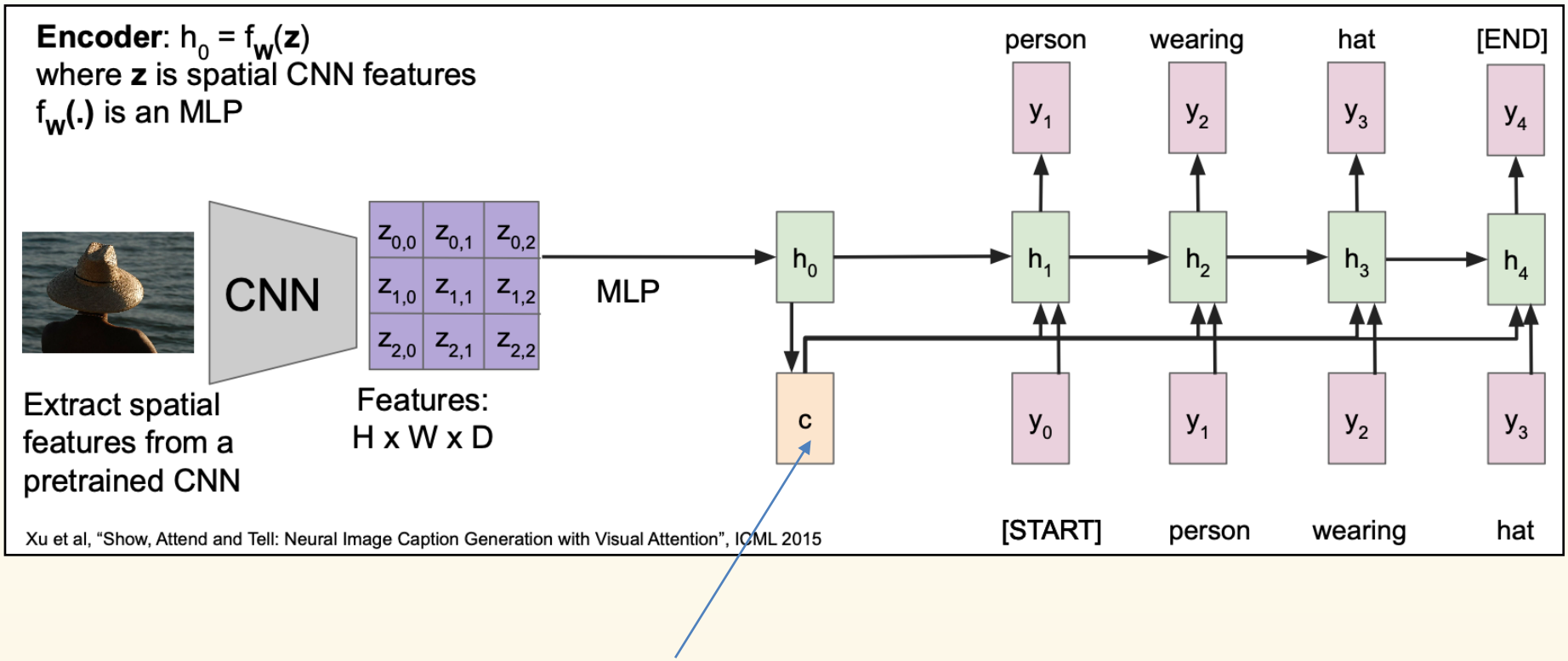
- We can generate a sequence (of words) using this mechanism




Context vector, has to contain the full information of the image

# A recurrent neural network approach

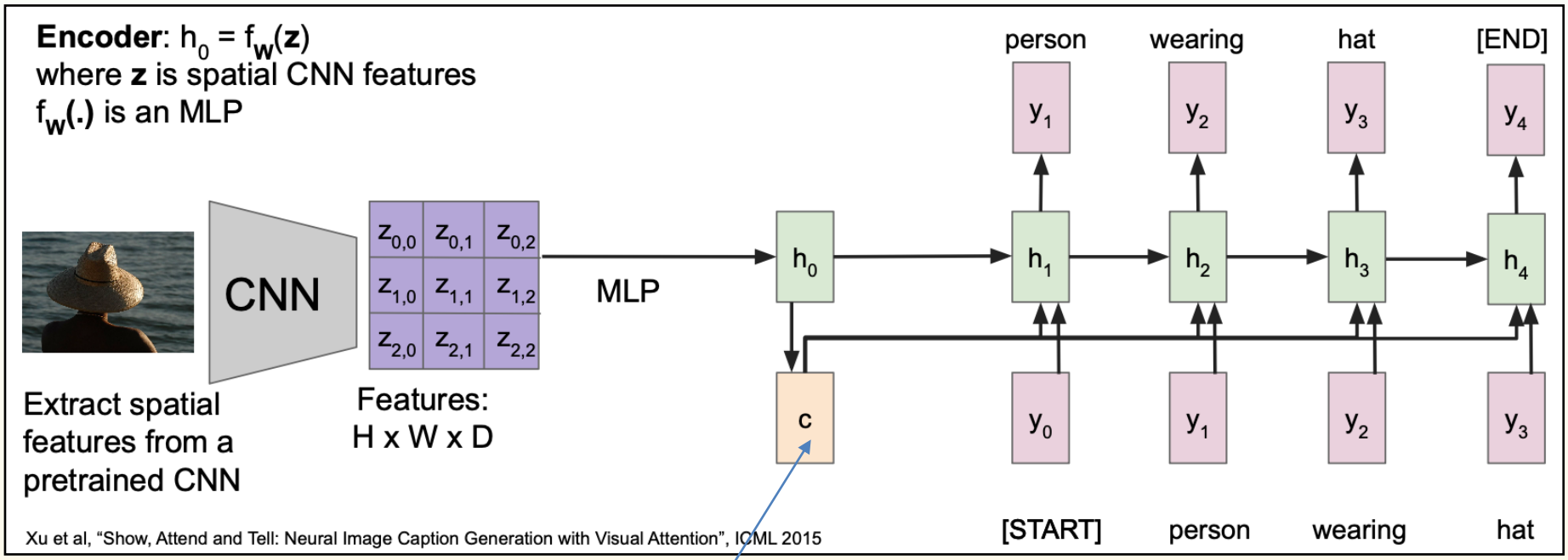
- We can generate a sequence (of words) using this mechanism




Context vector, has to contain the full information of the image

# A recurrent neural network approach

- We can generate a sequence (of words) using this mechanism

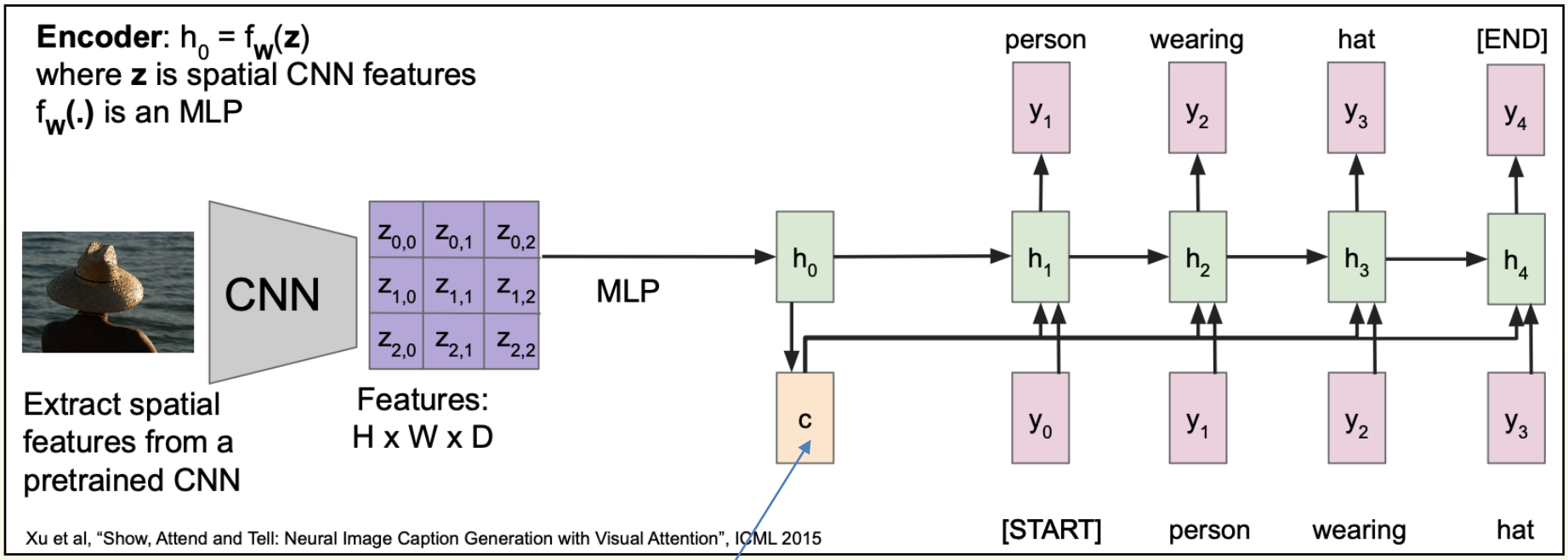



What could be the problem?

Context vector, has to contain the full information of the image

# A recurrent neural network approach

- We can generate a sequence (of words) using this mechanism

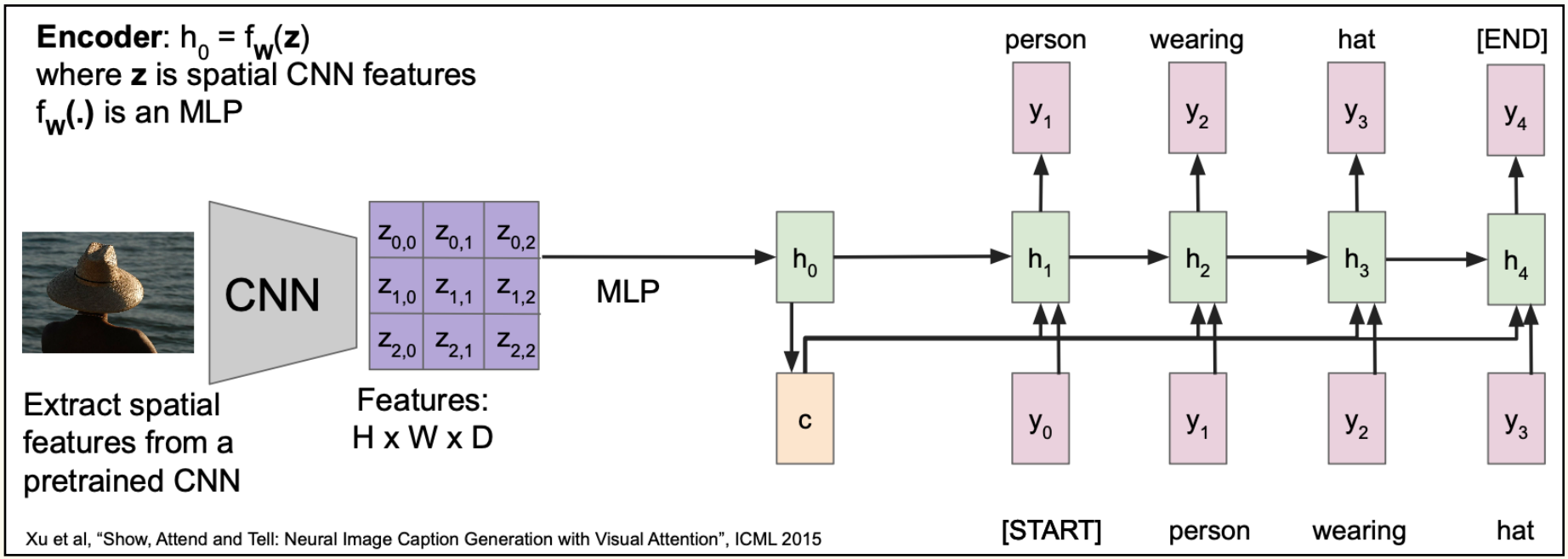



What could be the problem?

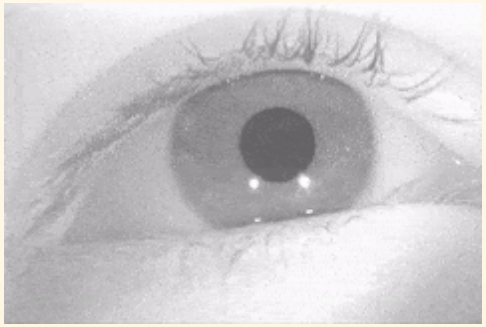
c is a bottleneck

Context vector, has to contain the full information of the image

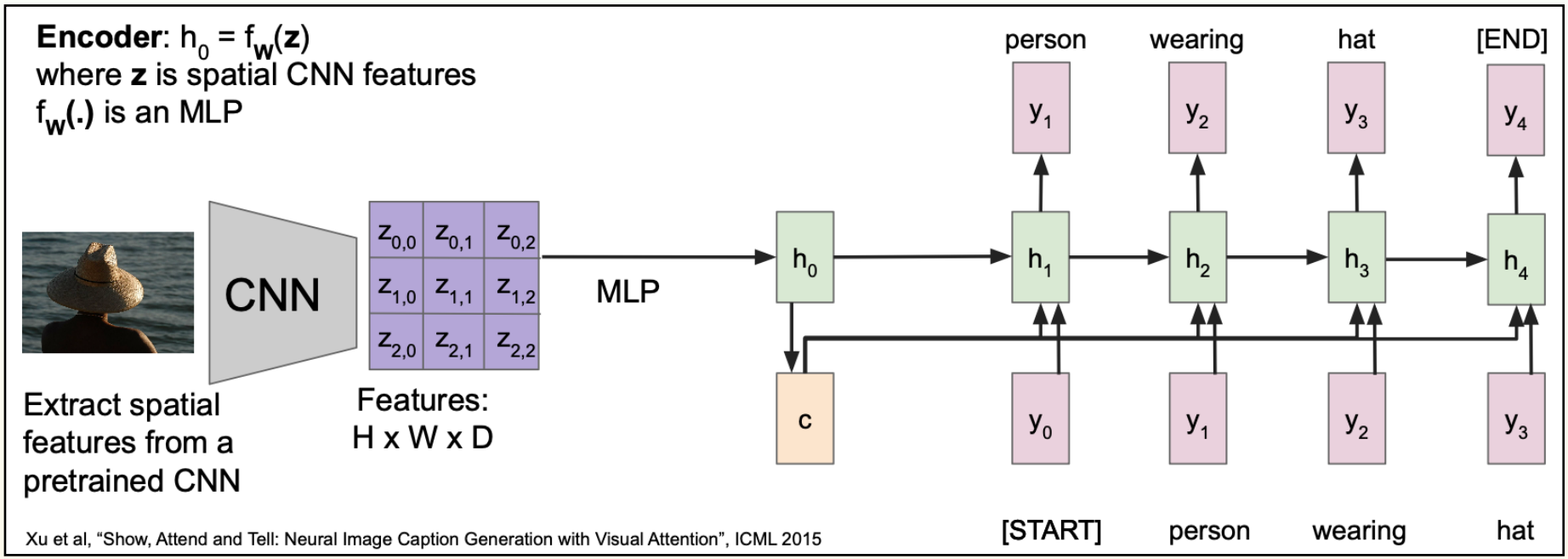
# Adding attention to the RNN



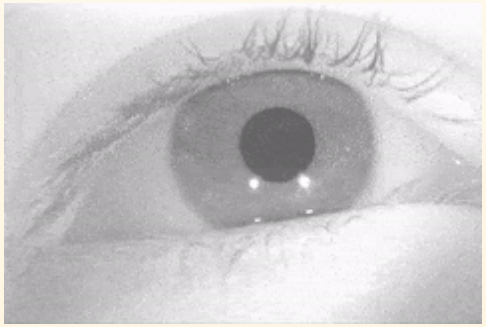

- Why not look at different regions of the image?
- Change the context!



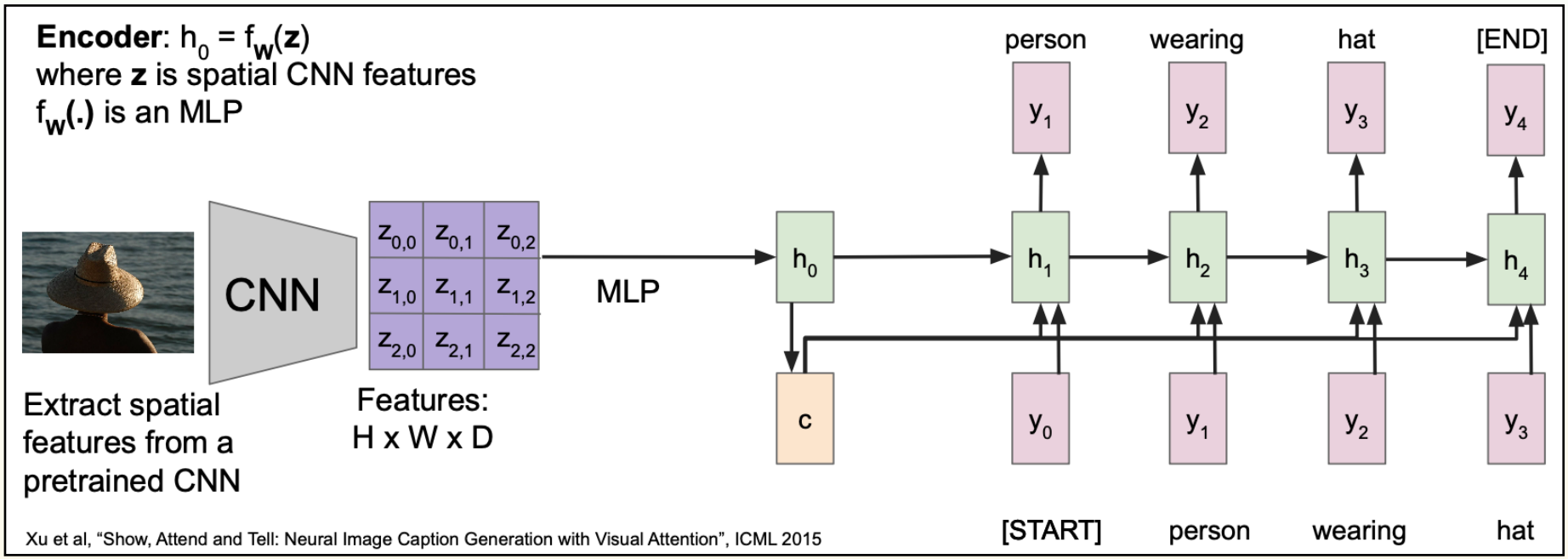
# Adding attention to the RNN




- Why not look at different regions of the image?
- Change the context!

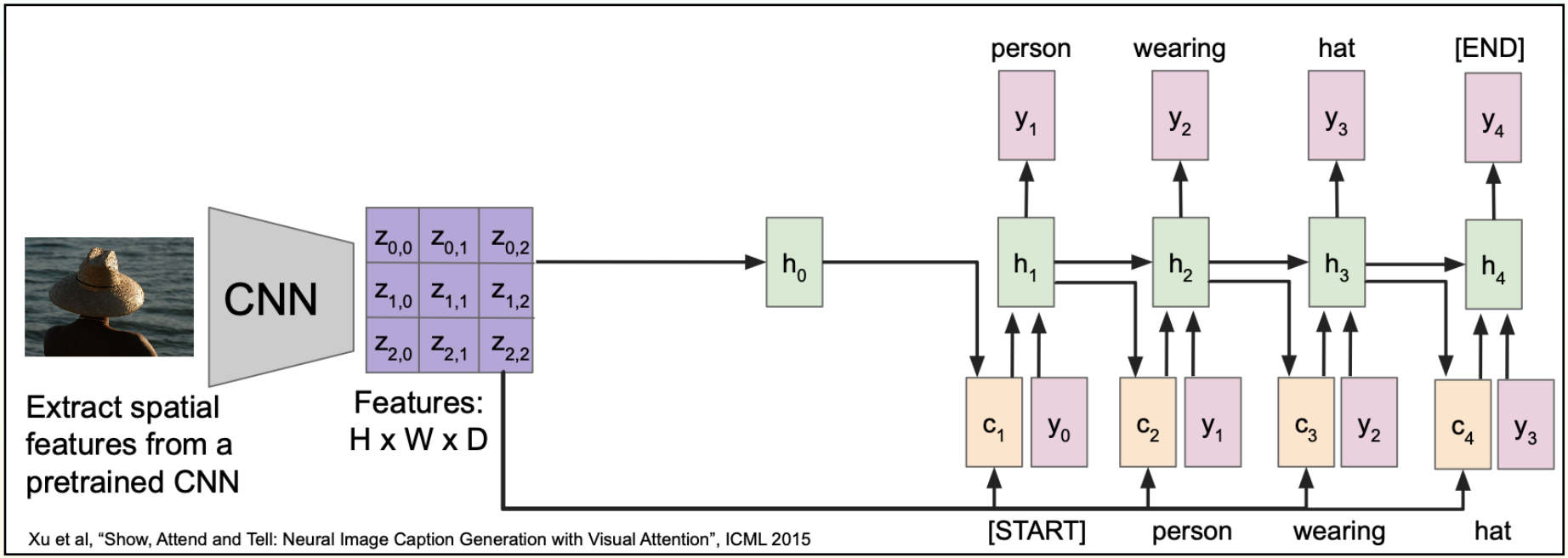


# Adding attention to the RNN



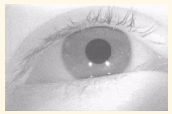

- New context
  - Can 'mask' regions we are not interested in: a filter with values between 0 and 1
  - Needs to know what we looked at before (so $h_{t-1}$ has to be taken into account)

# Adding attention to the RNN




Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

$$c_t = F\left( \begin{bmatrix} a_{1,0,0} & a_{1,0,1} & a_{1,0,2} \\ a_{1,1,0} & a_{1,1,1} & a_{1,1,2} \\ a_{1,2,0} & a_{1,2,1} & a_{1,2,2} \end{bmatrix} \cdot \begin{bmatrix} z_{0,0} & z_{0,1} & z_{0,2} \\ z_{1,0} & z_{1,1} & z_{1,2} \\ z_{2,0} & z_{2,1} & z_{2,2} \end{bmatrix} \right) = F\left( \Phi(h_{t-1}, z) \cdot \begin{bmatrix} z_{0,0} & z_{0,1} & z_{0,2} \\ z_{1,0} & z_{1,1} & z_{1,2} \\ z_{2,0} & z_{2,1} & z_{2,2} \end{bmatrix} \right)$$



Element wise

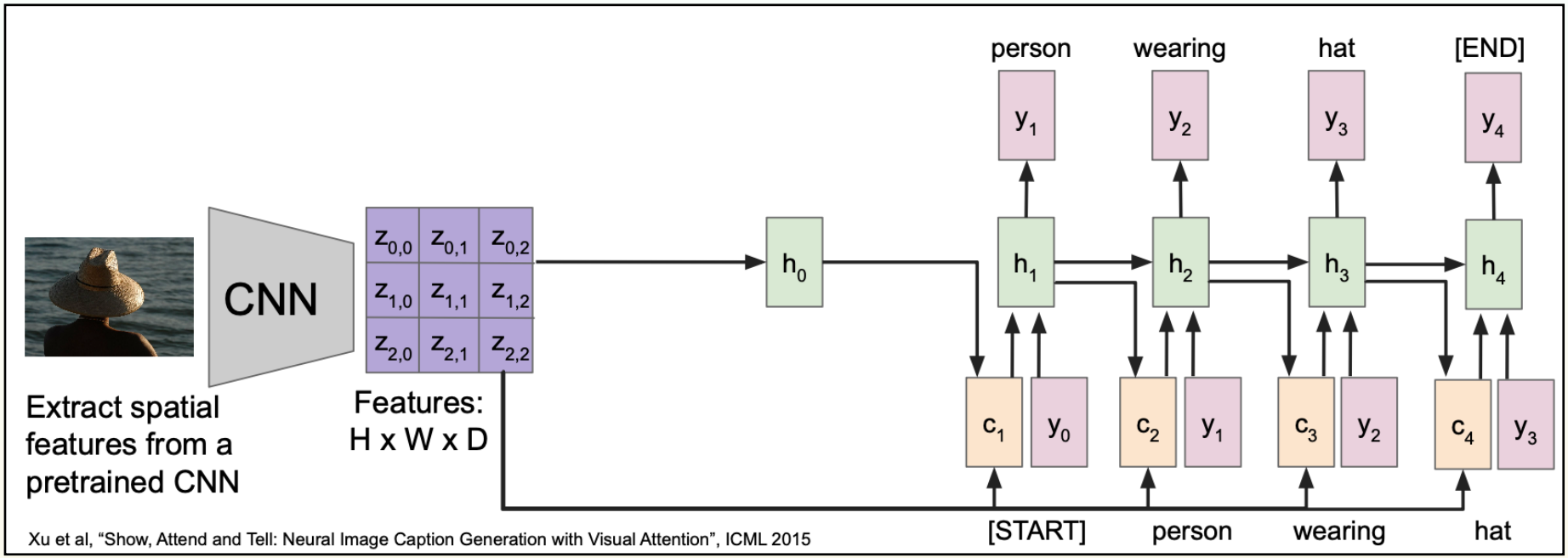
Some sort of reduction (e.g. sum or CNN)

Function of previous step **and** the (hidden) image features:

$0 \leq a_{t,i,j} \leq 1$

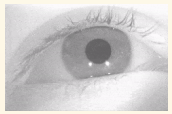
There are different exact implementations of it (in particular $\Phi$), but the concepts are similar [covered in a few slides]

# Adding attention to the RNN




Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

$$c_t = F\left( \begin{bmatrix} a_{1,0,0} & a_{1,0,1} & a_{1,0,2} \\ a_{1,1,0} & a_{1,1,1} & a_{1,1,2} \\ a_{1,2,0} & a_{1,2,1} & a_{1,2,2} \end{bmatrix} \cdot \begin{bmatrix} z_{0,0} & z_{0,1} & z_{0,2} \\ z_{1,0} & z_{1,1} & z_{1,2} \\ z_{2,0} & z_{2,1} & z_{2,2} \end{bmatrix} \right) = F\left( \Phi(h_{t-1}, z) \cdot \begin{bmatrix} z_{0,0} & z_{0,1} & z_{0,2} \\ z_{1,0} & z_{1,1} & z_{1,2} \\ z_{2,0} & z_{2,1} & z_{2,2} \end{bmatrix} \right)$$



Element wise

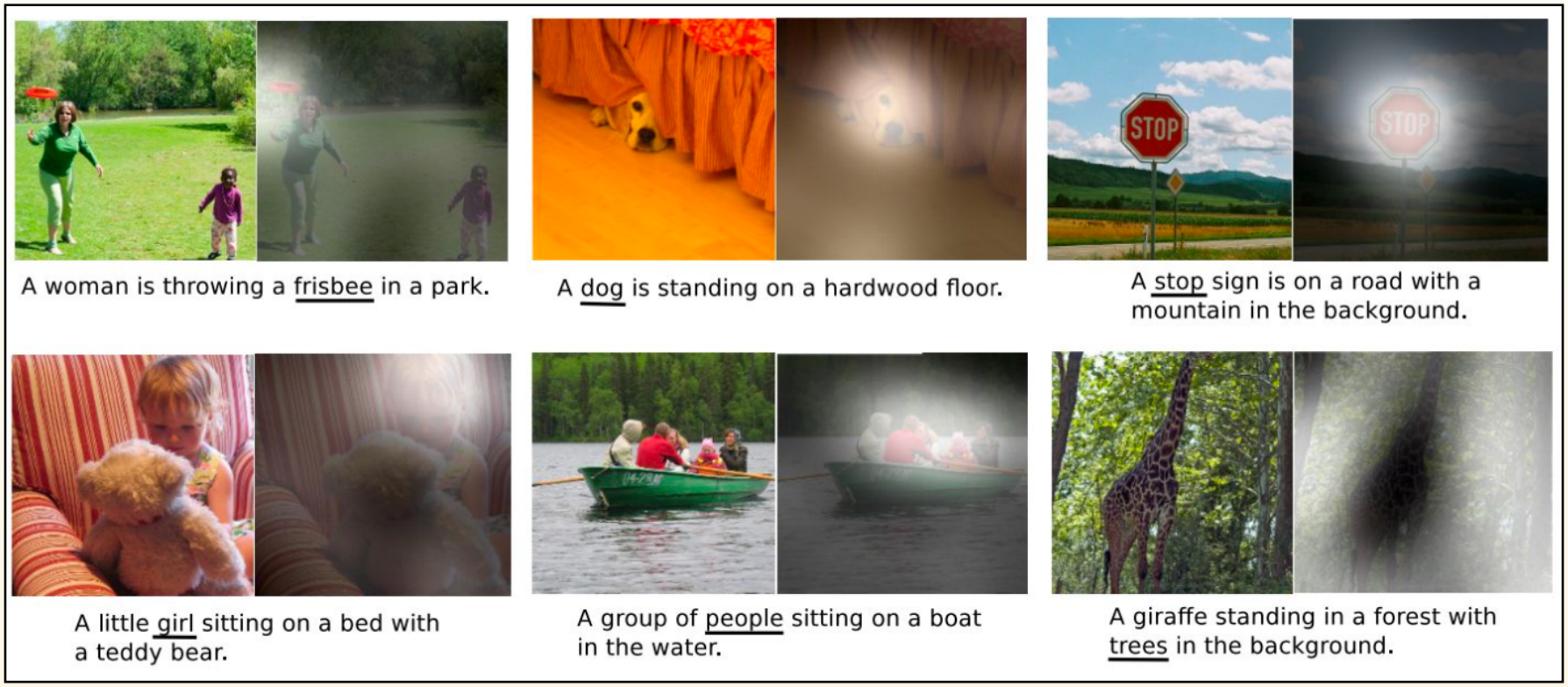
Some sort of reduction (e.g. sum or CNN)

Function of previous step **and** the (hidden) image features:

$0 \leq a_{t,i,j} \leq 1$

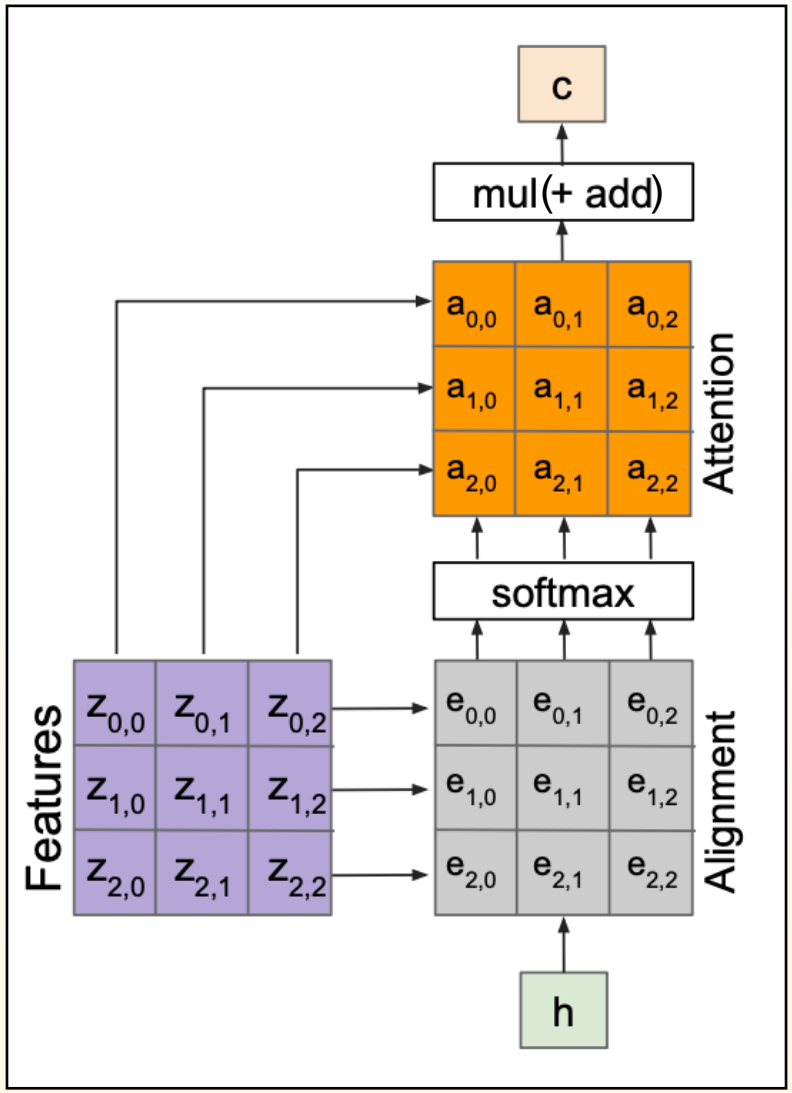
There are different exact implementations of it (in particular $\Phi$), but the concepts are similar [covered in a few slides]

# Attention in action



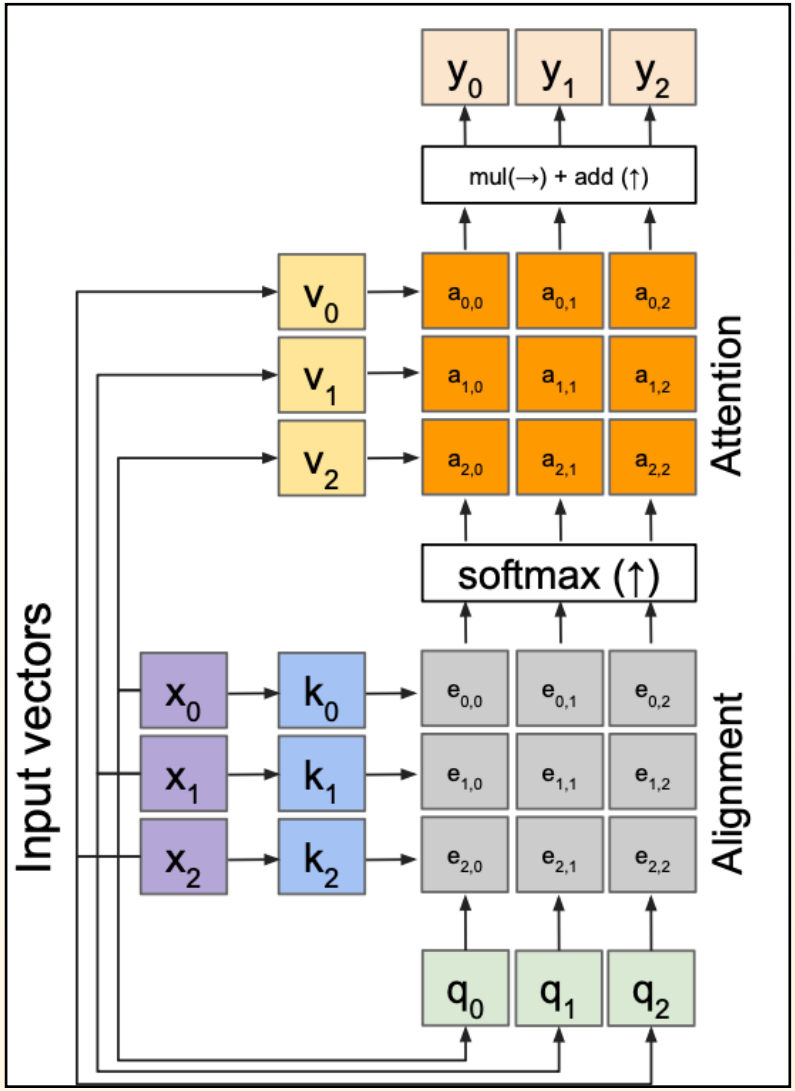
# A closer look at the attention part (removal candidate)

Our previous attention layer



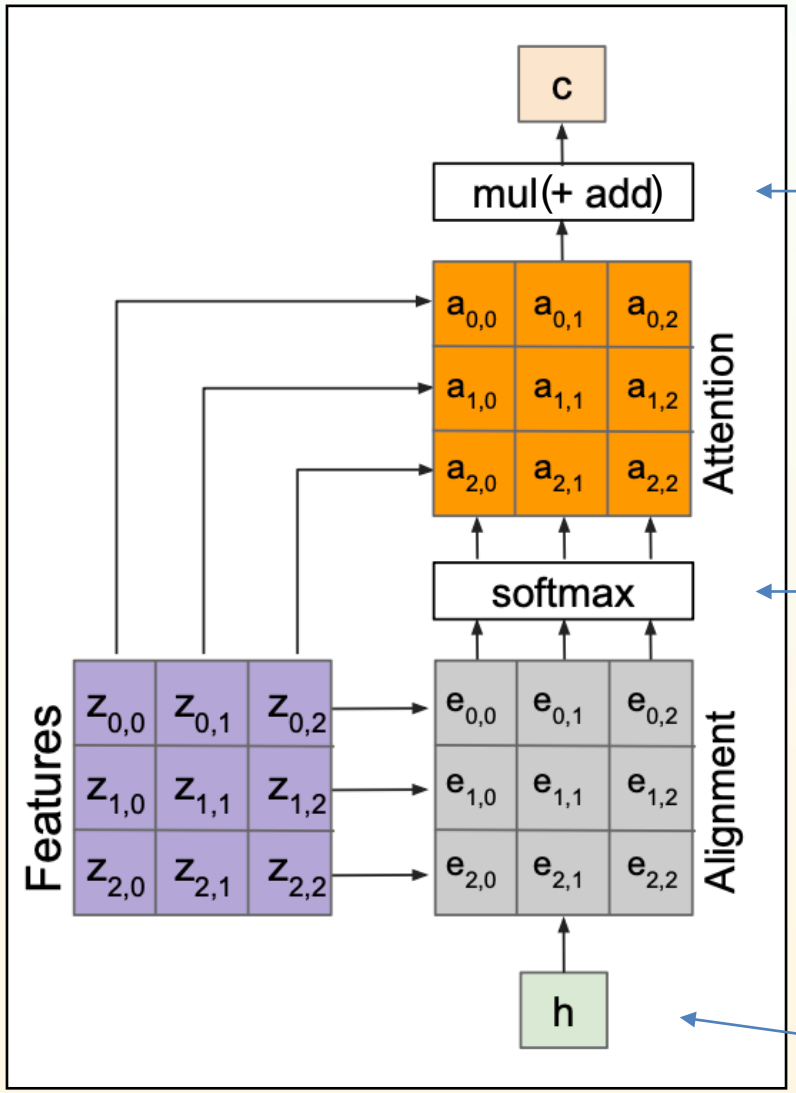

Next slides

A self-attention layer



# A closer look at the attention part (bottom to top)

For simplicity: one feature → one c
(extends transparently)



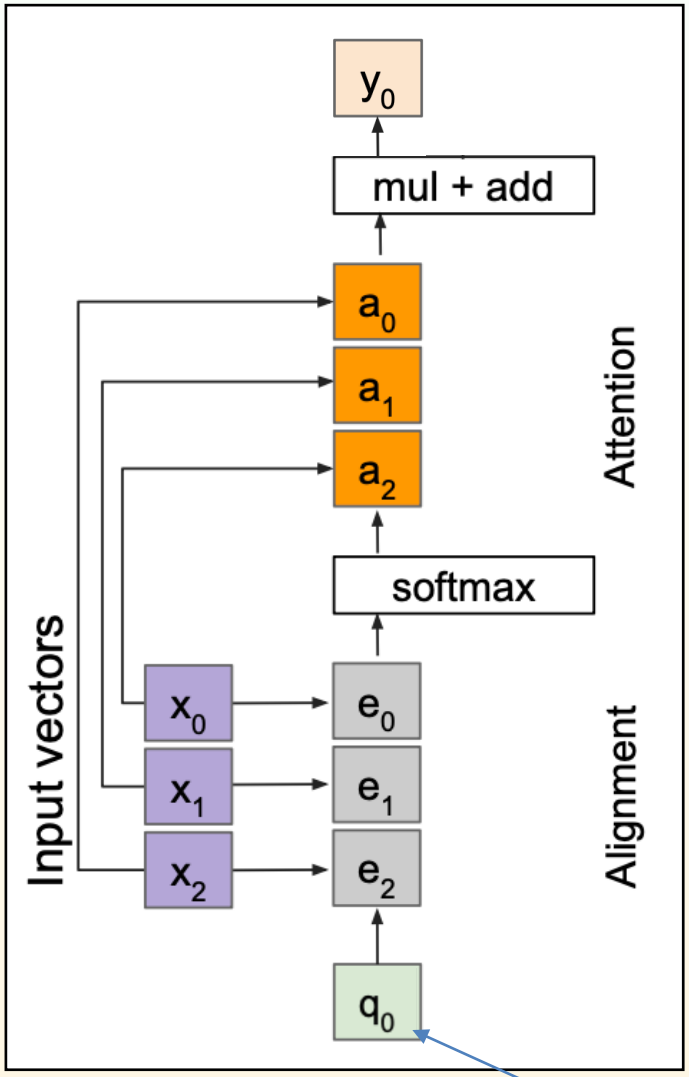

Multiply with features, element-wise and reduce (e.g. sum, could also be done by other means), no ij indices
**Permutation invariant w.r.t. ij**

Simply restrict $e_{ij} \rightarrow a_{ij} \in [0, 1]$
**order equivariant**

A function bringing together features $z$ and $h$
**Element-wise** in $z$
$e_{ij} \in \mathbb{R}$ (**order equivariant**)

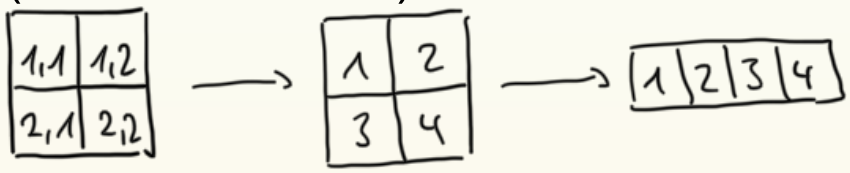
From previous iteration, no ij indices

# A closer look at the attention part




- This is the same attention layer, just having replaced indices $ij$ by one running index $l$

- This is simply an index unrolling (remember CNN)



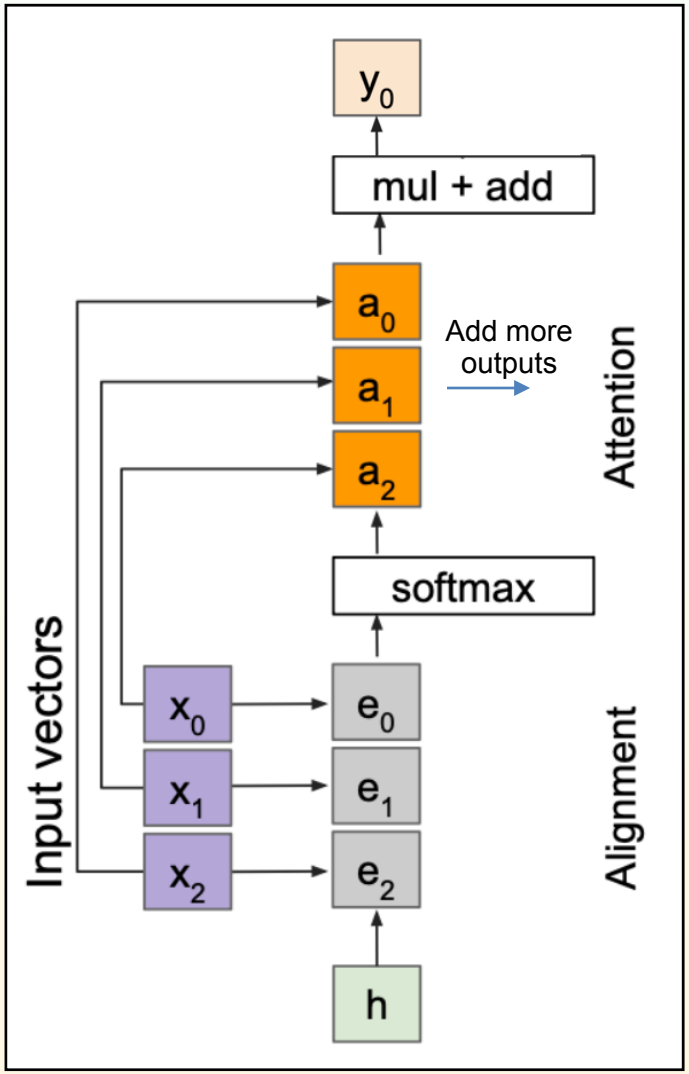

$$y_0 = \sum_i^N a_i(q_0, x_i)\, x_i$$

CNN

$$y_j = \theta\left(\sum_i^{N_k} \omega_i\, x_{I(j,i)} - T\right)$$

- Difference to convolution: $a_i$ is fully determined by h and $x_i$ : $y_j$ is still **invariant w.r.t. permutations in x**

# Multiple outputs

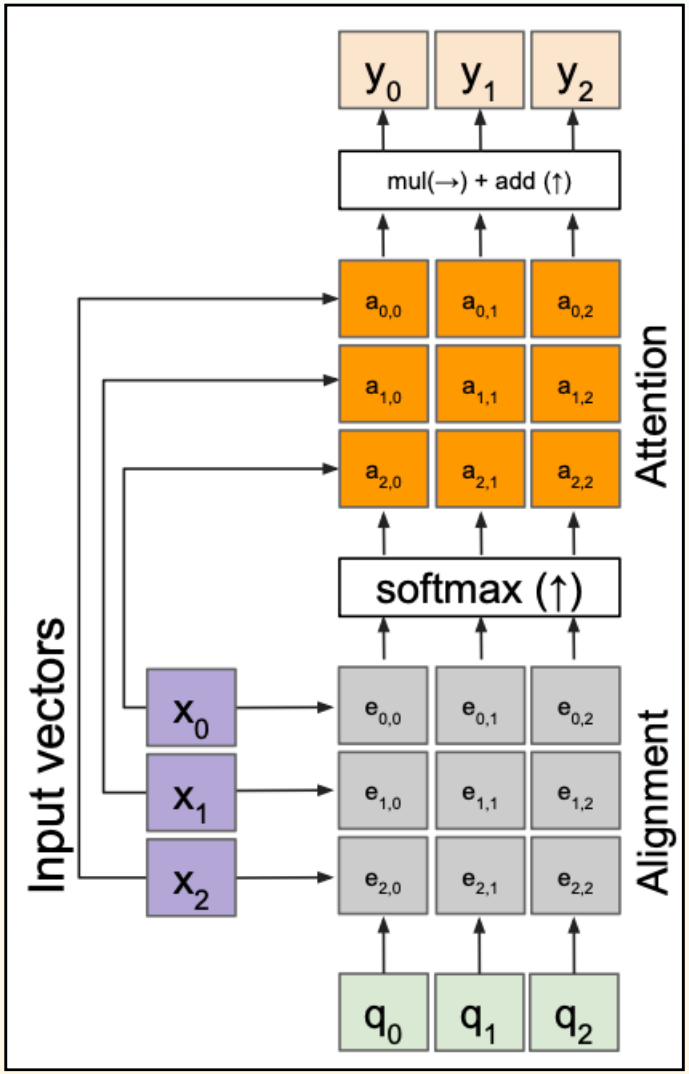



- Multiple outputs, otherwise nothing changed

$$y_j = \sum_i^N a_{ij}(q_j, x_i)\, x_i$$

- Example:
  if $a$ = identity, then $y_i = x_i$

# Multiple outputs

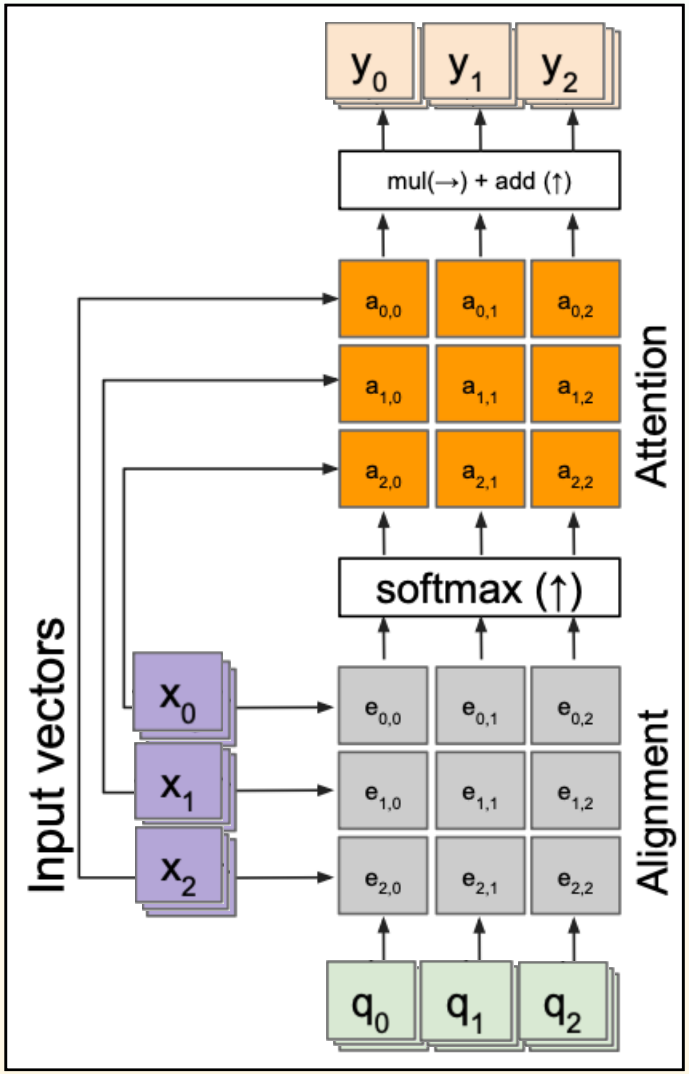



- Multiple outputs, otherwise nothing changed

$$y_j = \sum_i^N a_{ij}(q_j, x_i)\, x_i$$

- Example:
  if $a$ = identity, then $y_i = x_i$

# Adding more dimensions



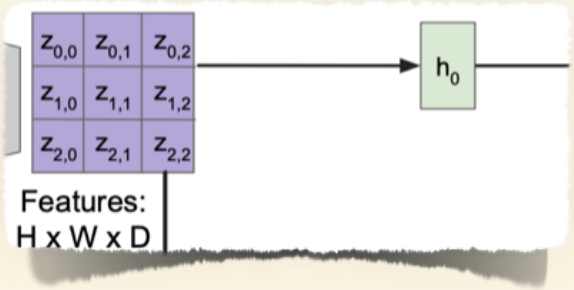

- Multiple dimensions per input

$$y_{jl} = \sum_{i}^{N} a_{ij}(q_j, x_i)\, x_{il}$$

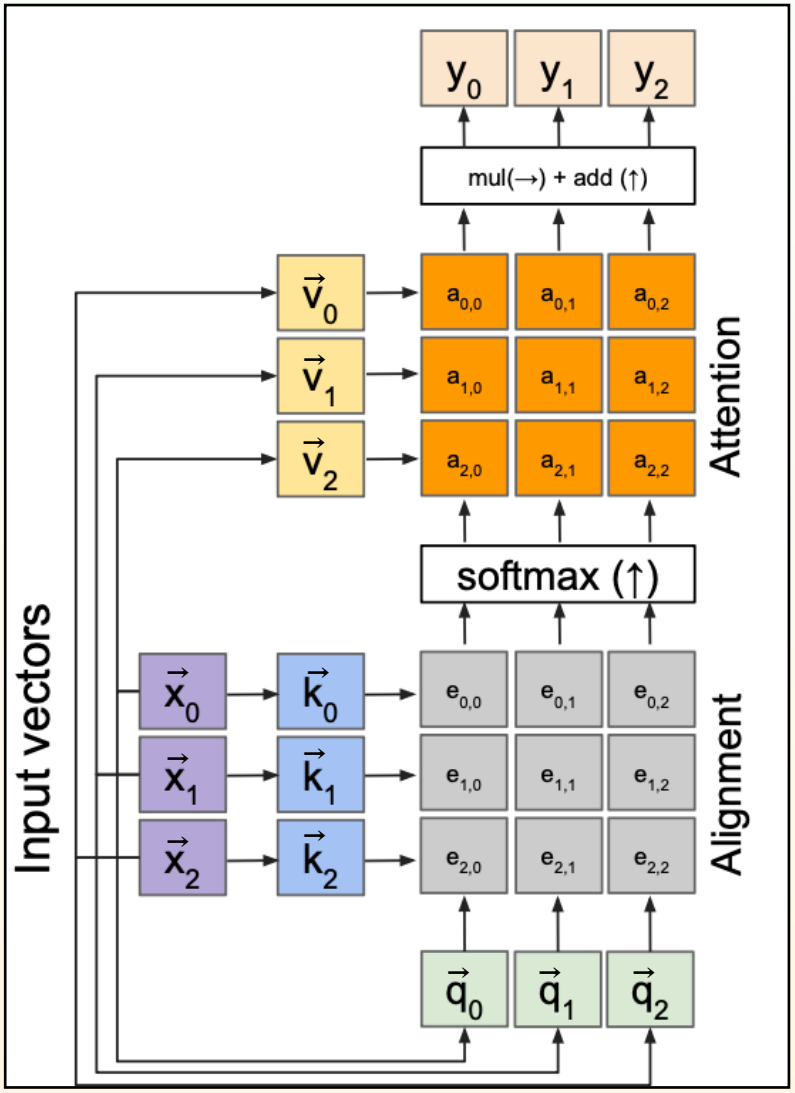
$$a_{ij}(q_j, x_i) = \text{softmax}_j\,(\vec{q_j} \cdot \vec{x_i} / \sqrt{d_q})$$

Why that?

- Where do $q$ come from? Can be external but remember:

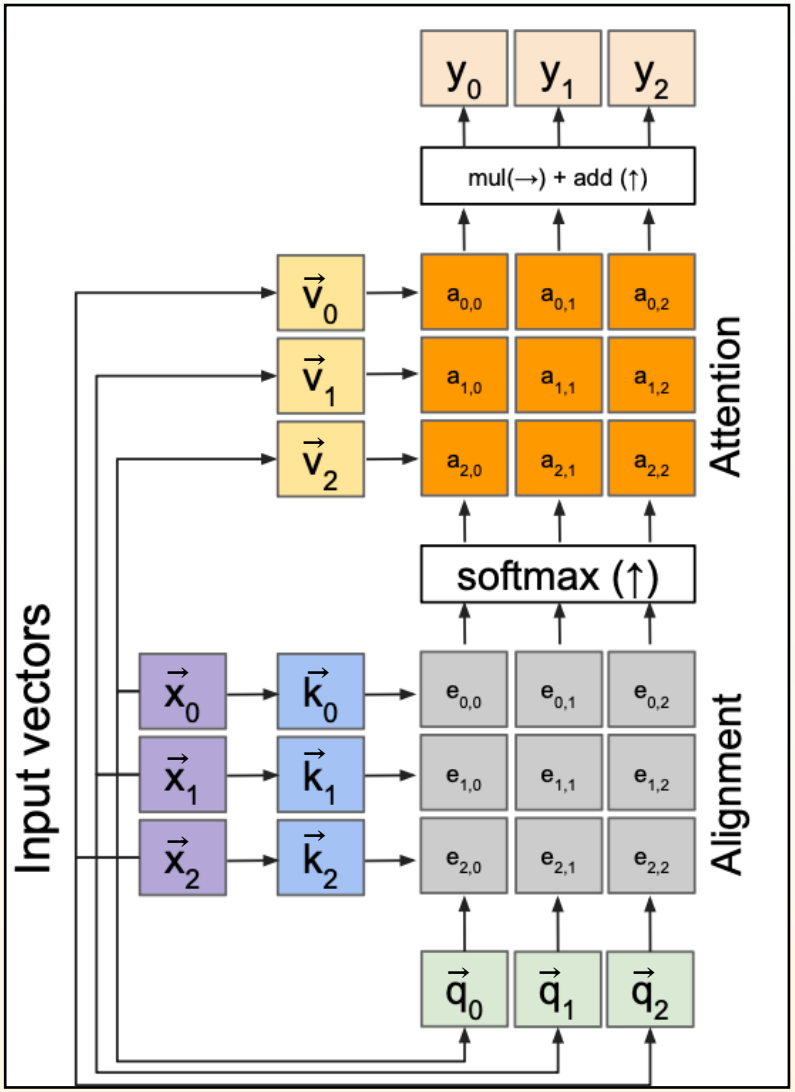


# The final self-attention



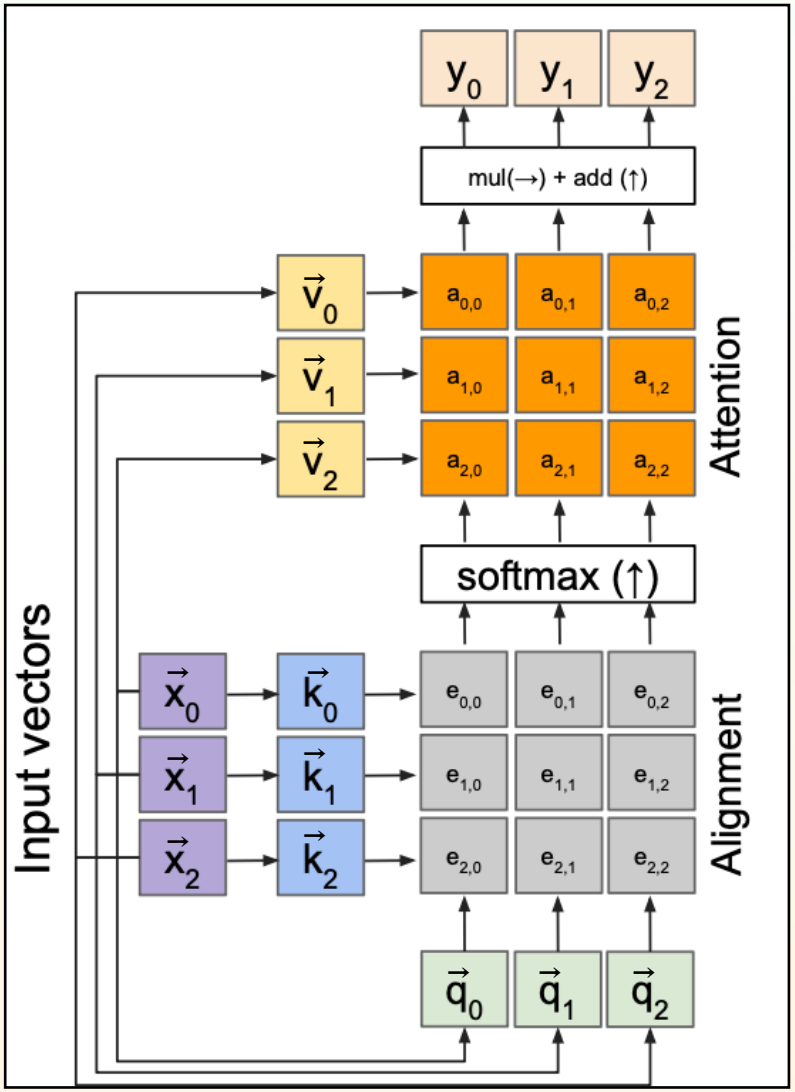

- We can generate the queues from the inputs directly: “self-attention” with an MLP
- Add expressivity by adding transformations (=MLPs) to build *keys* and *values*
- Keys must have same dimension as queries
  (so that scalar product is defined)
- Counting dimensions:
  - $N \times d_x$: input
  - $N \times d_k$: keys, queries
  - $N \times d_v$: values, outputs

# The final self-attention




- In equations:

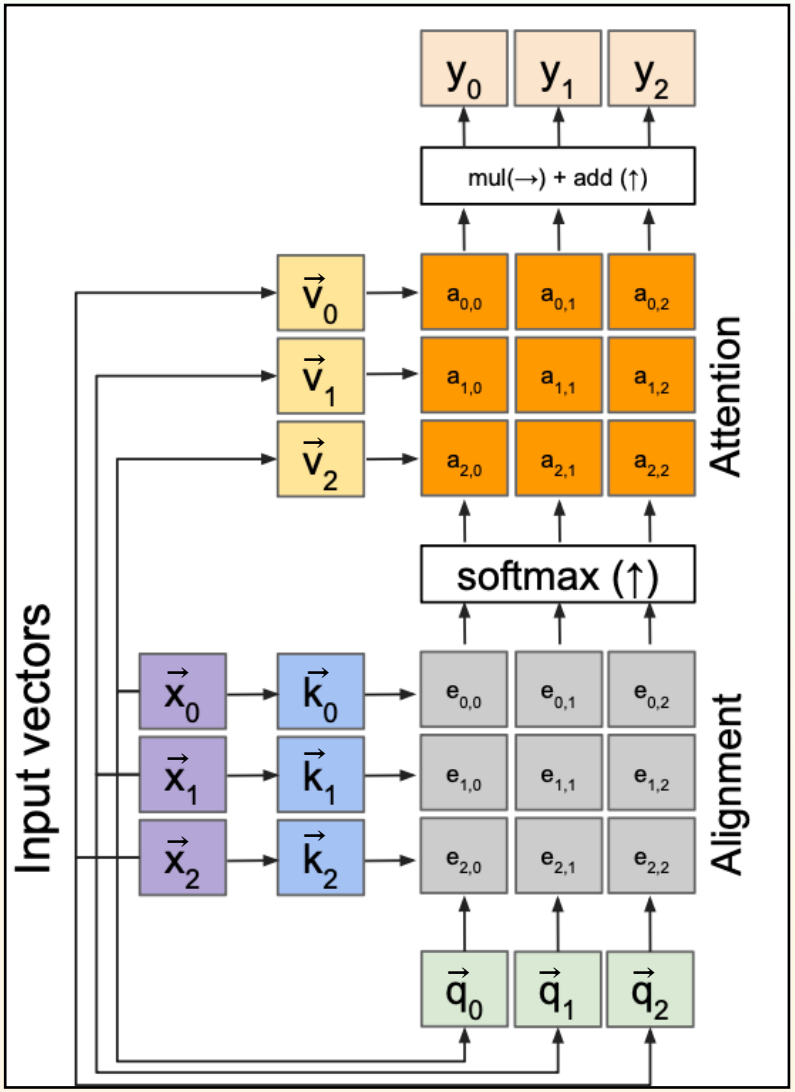
# The final self-attention




- In equations:

$$y_{j\beta} = \sum_{i}^{N} a_{ij}(q_j, k_i)\ v_{i\beta}$$

# The final self-attention

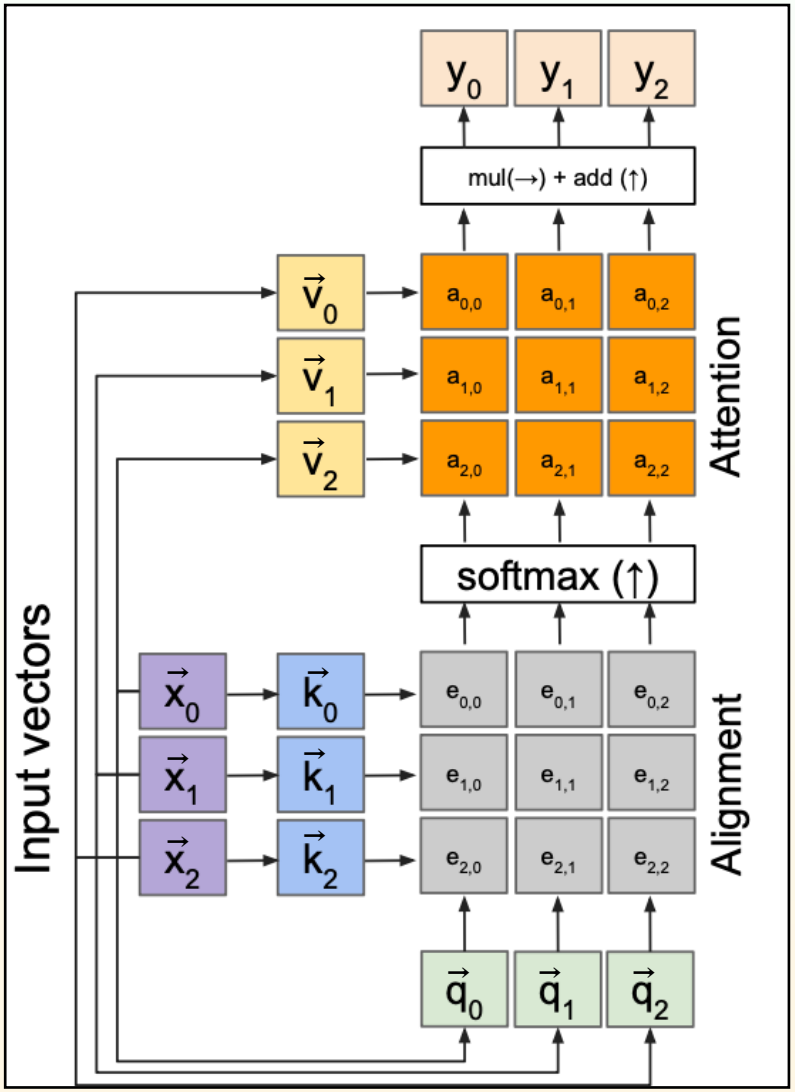



- In equations:

$$y_{j\beta} = \sum_{i}^{N} a_{ij}(q_j, k_i)\, v_{i\beta}$$

- $a_{ij} = \text{softmax}_j(\vec{k}_i \cdot \vec{q}_j / \sqrt{d_k})$

# The final self-attention

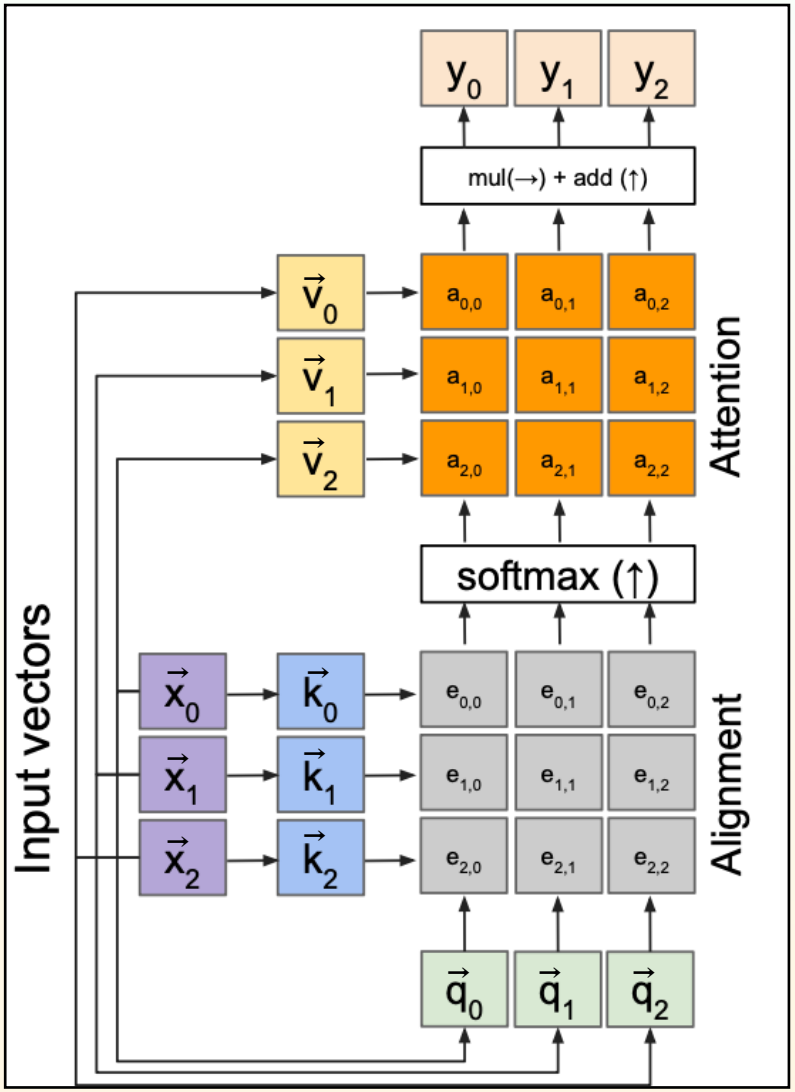



- In equations:

$$y_{j\beta} = \sum_{i}^{N} a_{ij}(q_j, k_i)\ v_{i\beta}$$

- $a_{ij} = \text{softmax}_j(\vec{k}_i \cdot \vec{q}_j / \sqrt{d_k})$
- $\vec{q}_i = \vec{x}_i \omega^q \quad \omega^q \epsilon\ \mathbb{R}^{d_x \times d_k}$

# The final self-attention

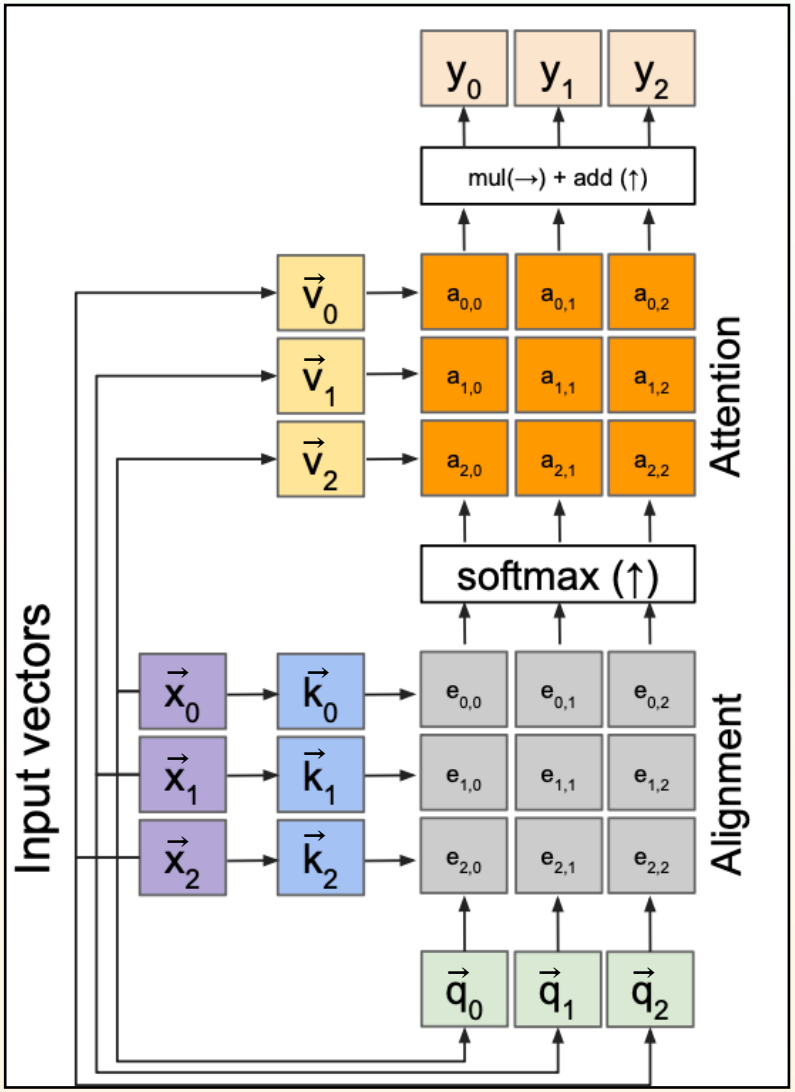



- In equations:

$$y_{j\beta} = \sum_{i}^{N} a_{ij}(q_j, k_i)\, v_{i\beta}$$

- $a_{ij} = \text{softmax}_j(\vec{k}_i \cdot \vec{q}_j / \sqrt{d_k})$
- $\vec{q}_i = \vec{x}_i \omega^q \quad \omega^q \epsilon\ \mathbb{R}^{d_x \times d_k}$
- $\vec{k}_i = \vec{x}_i \omega^k \quad \omega^k \epsilon\ \mathbb{R}^{d_x \times d_k}$

# The final self-attention

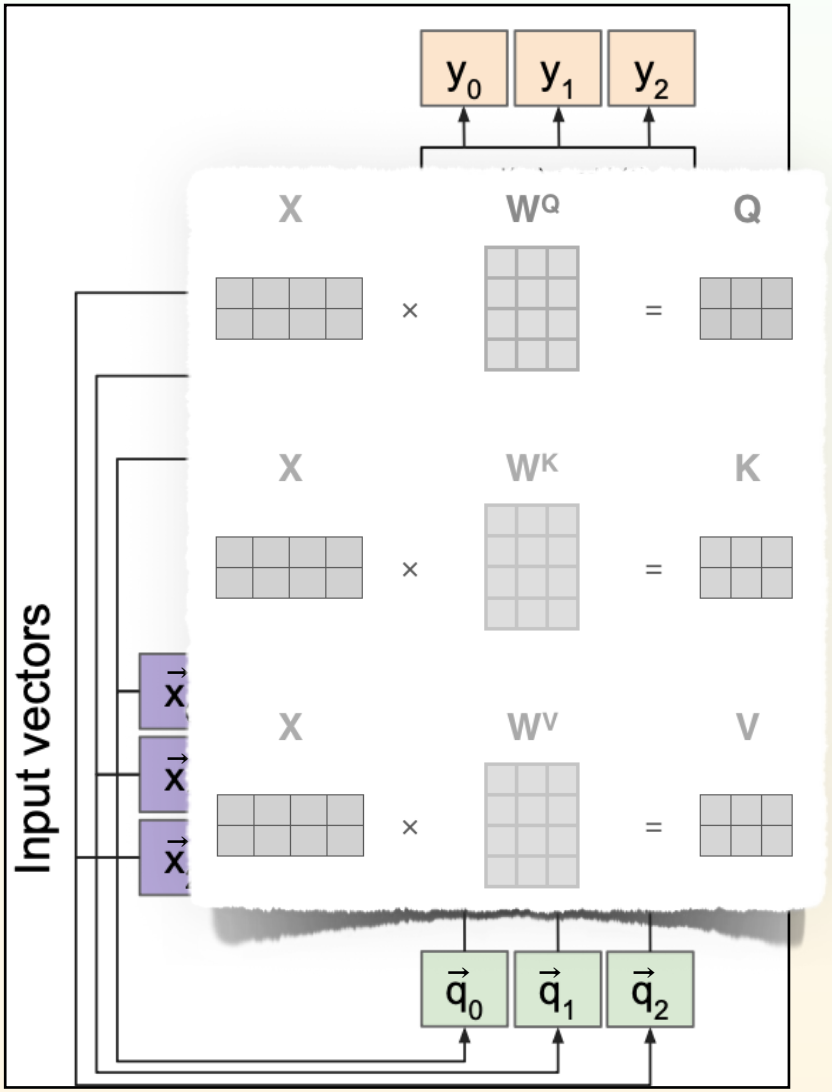



- In equations:

$$y_{j\beta} = \sum_{i}^{N} a_{ij}(q_j, k_i)\, v_{i\beta}$$

- $a_{ij} = \mathrm{softmax}_j(\vec{k}_i \cdot \vec{q}_j / \sqrt{d_k})$
- $\vec{q}_i = \vec{x}_i \omega^q \quad \omega^q \epsilon\ \mathbb{R}^{d_x \times d_k}$
- $\vec{k}_i = \vec{x}_i \omega^k \quad \omega^k \epsilon\ \mathbb{R}^{d_x \times d_k}$
- $\vec{v}_i = \vec{x}_i \omega^v \quad \omega^v \epsilon\ \mathbb{R}^{d_x \times d_v}$

# The final self-attention




- In equations:

$$y_{j\beta} = \sum_{i}^{N} a_{ij}(q_j, k_i)\, v_{i\beta}$$

- $a_{ij} = \text{softmax}_j(\vec{k}_i \cdot \vec{q}_j / \sqrt{d_k})$
- $\vec{q}_i = \vec{x}_i \omega^q \quad \omega^q \epsilon\ \mathbb{R}^{d_x \times d_k}$
- $\vec{k}_i = \vec{x}_i \omega^k \quad \omega^k \epsilon\ \mathbb{R}^{d_x \times d_k}$
- $\vec{v}_i = \vec{x}_i \omega^v \quad \omega^v \epsilon\ \mathbb{R}^{d_x \times d_v}$