

Machine Learning

Lecture 2

–

Exploiting Structures

Jan Kieseler

jan.kieseler@kit.edu

Goal: Turn your own Physics Problem into a Well-Posed ML Problem

Lecture 1: Learning a Function

understand the complete basic learning loop and how a neural network is trained to approximate a deterministic function

Lecture 2: Exploiting Structure

reason from the structure and symmetries of a problem to an appropriate representation and architecture

Lecture 3: Learning a Process

understand how random inputs turn deterministic neural networks into generators, and how to train and evaluate them.

*You seem to prefer at the end,
but still..*

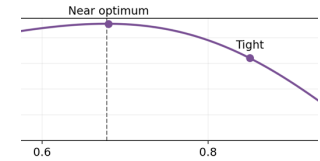
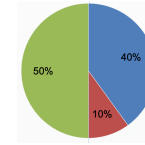
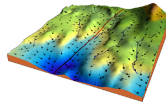
Ask questions at any time

**Distill the most important take-aways:
we will write the conclusions together at the end of each lecture**

Recap (DIY Conclusions)

$$\Phi(x; \omega)$$

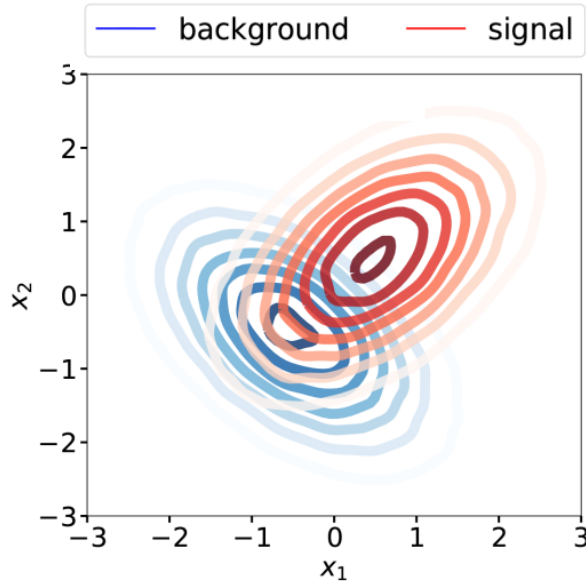
$$L(\hat{y}, y)$$



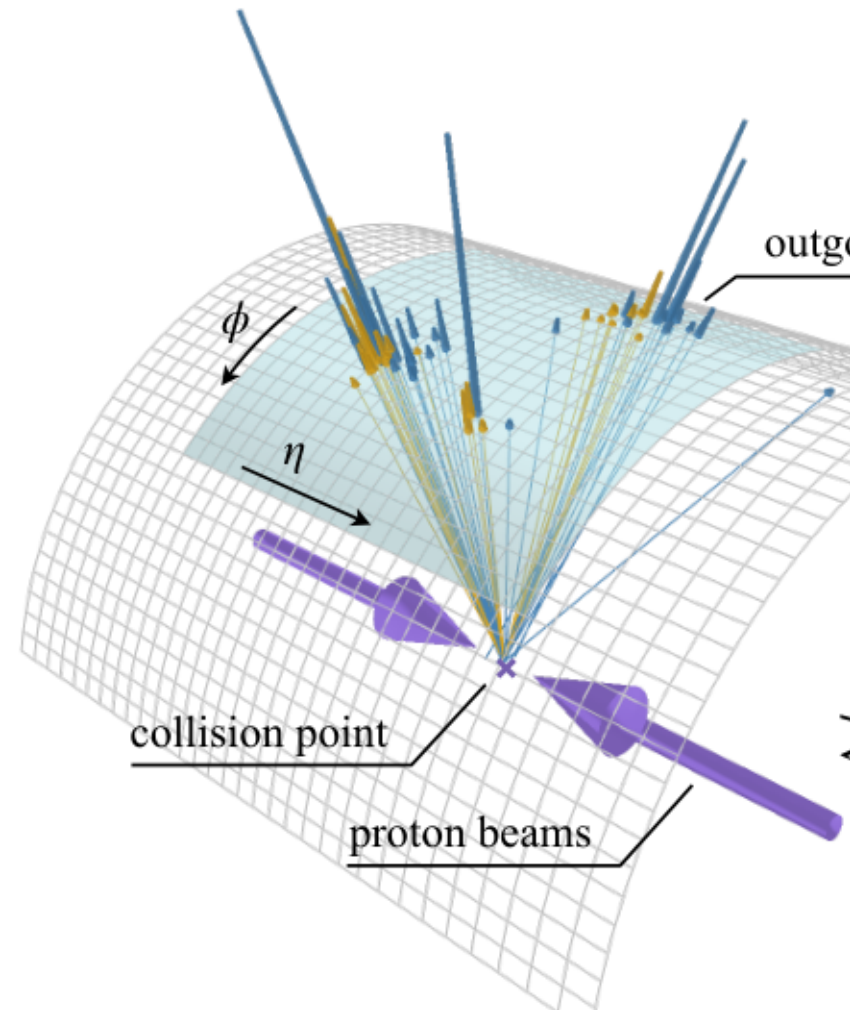
- Neural networks are universal function approximators
- The loss quantifies the discrepancy of the truth value and the NN output; often comes from a likelihood
- If validation loss increases over training loss, we overfit; prevent by using more data or with regularisation
- We find the minimum with gradient descent
- The learning rate is an important hyper parameter
- Physics performance decides what the best model is, not just ROC curves etc.
- The dataset should be representative of the problem

Inputs can have Structure and Symmetries

- $\Phi(x; \omega)$



Come from

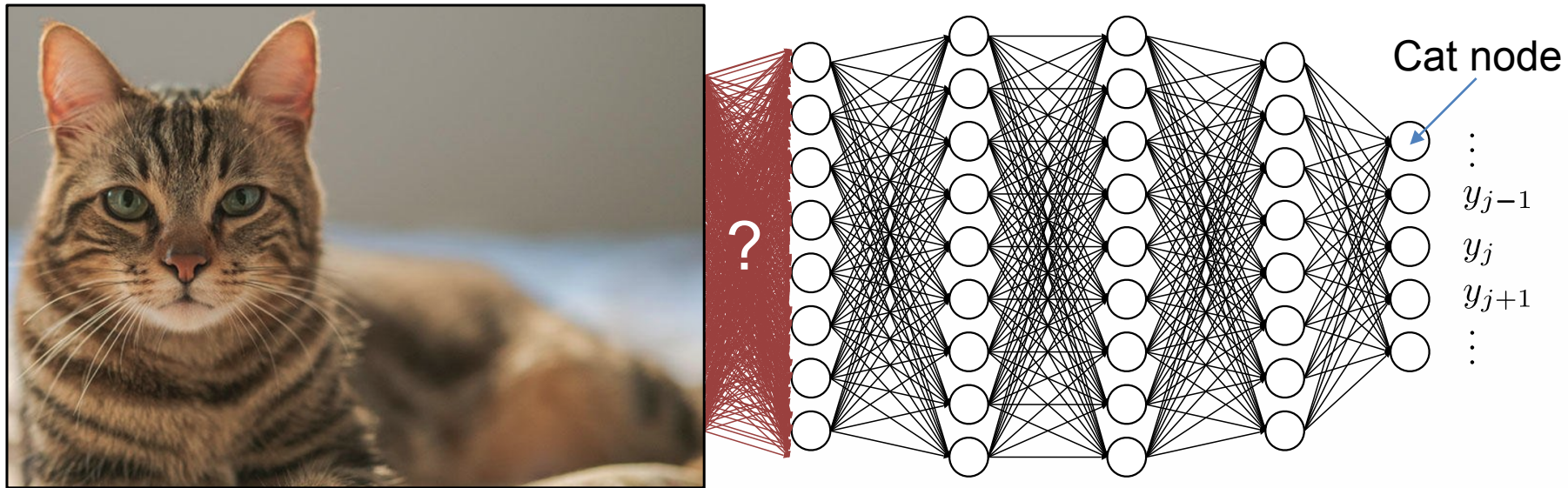


- Architecture needs to fit the **input data**
- Architecture needs to fit the **desired output**

H. Qu, et al., arXiv:2202.03772

Structure counts

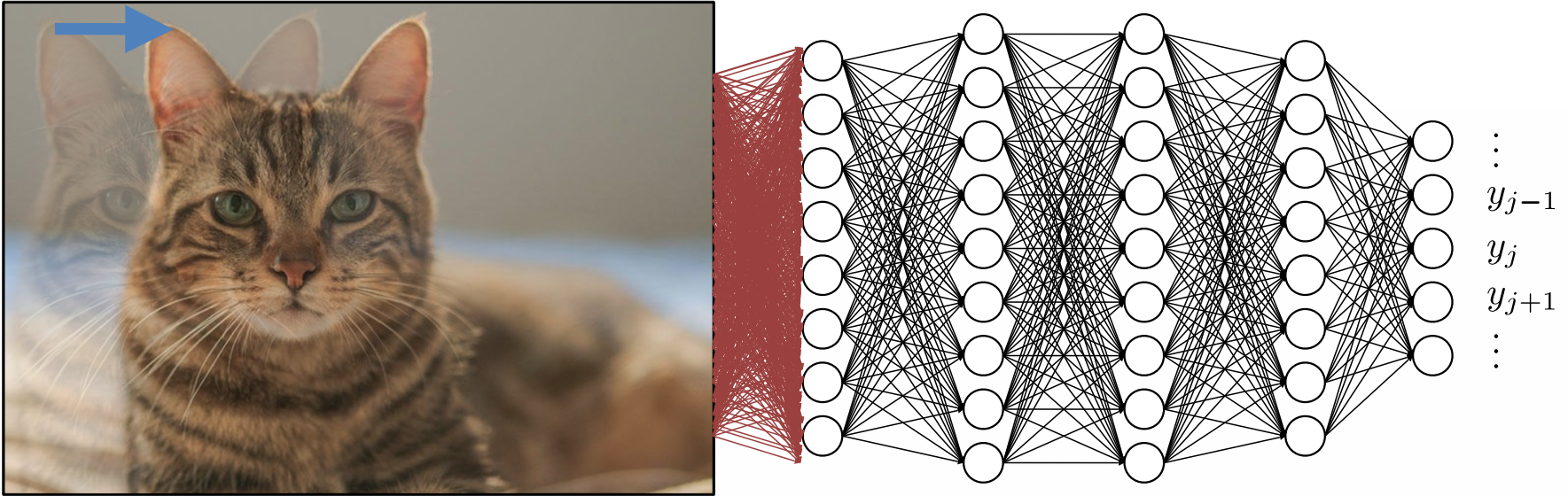
- Is this an image of a cat?



- Typical (phone) cameras 10-50 MP
- How many parameters does the first layer have?
- In this example: **80 - 400 million parameters** in first layer
- Also, this architecture will not perform well

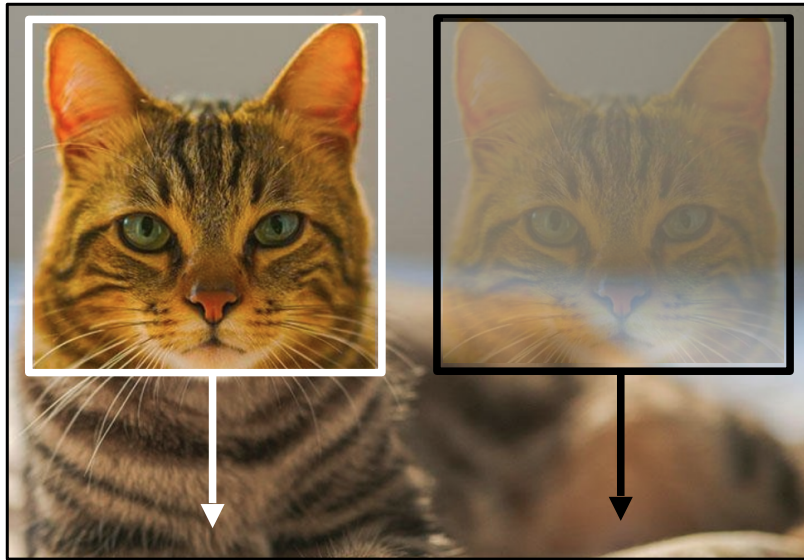
Structure counts

- What if the cat moved?



- Present **entirely different** input to the DNN
- This complexity cannot be captured by as little as 8 nodes
 - Lack of expressivity
- Solution: exploit the **structure** of the data

Introducing filters



Very cat-like:
Score = 1

Not at all cat-like
Score = 0

- Create a cat-face filter (no ML here)
- Slide it over the image
- Take maximum of all cat scores:
image cat score
- We found the cat



Cats come in different shapes



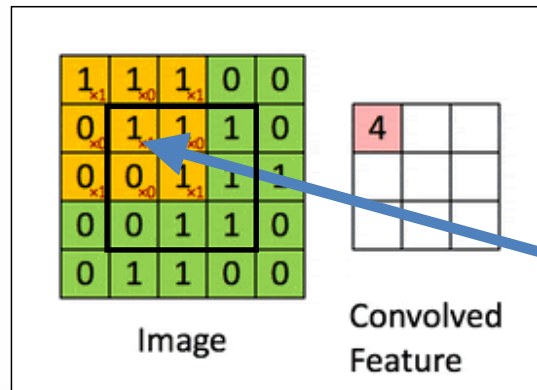
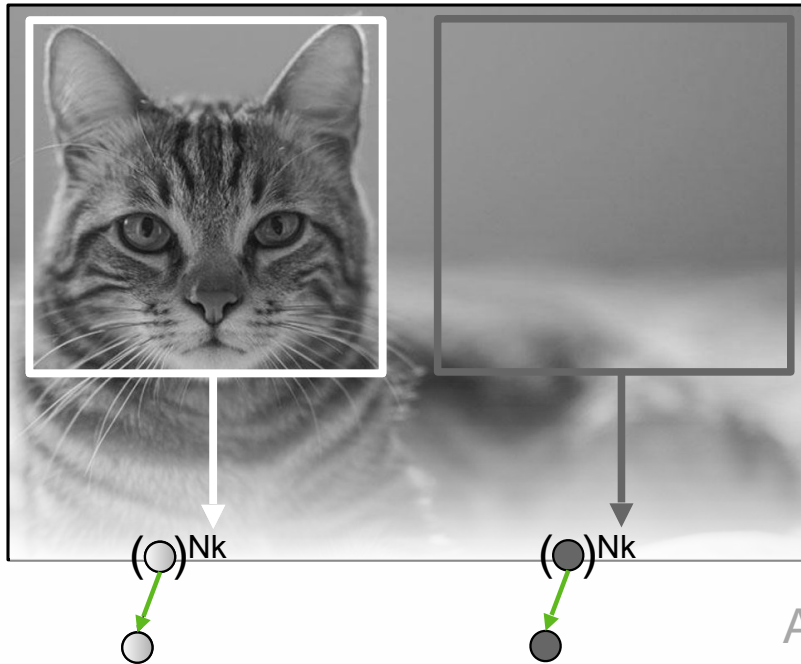
Not a cat



Not a cat

- Many different very complex filters are needed
- Can be solved by
 - **Learning filters from examples**
 - Abstraction

A CNN kernel: step by step



conditions at the edges → wait a few slides

- Inputs x
- For one *channel*: Learnable bias

$$y_j = \theta \left(\sum_i^{N_k} W_i x_{I(j,i)} + b \right)$$

Diagram illustrating the equation for the output y_j . The equation is $y_j = \theta \left(\sum_i^{N_k} W_i x_{I(j,i)} + b \right)$. Annotations include: N_k is labeled 'Kernel size', b is labeled 'Learnable bias', θ is labeled 'Activation function', and W_i is labeled 'Learnable weights: Relative position to j'.

Activation function

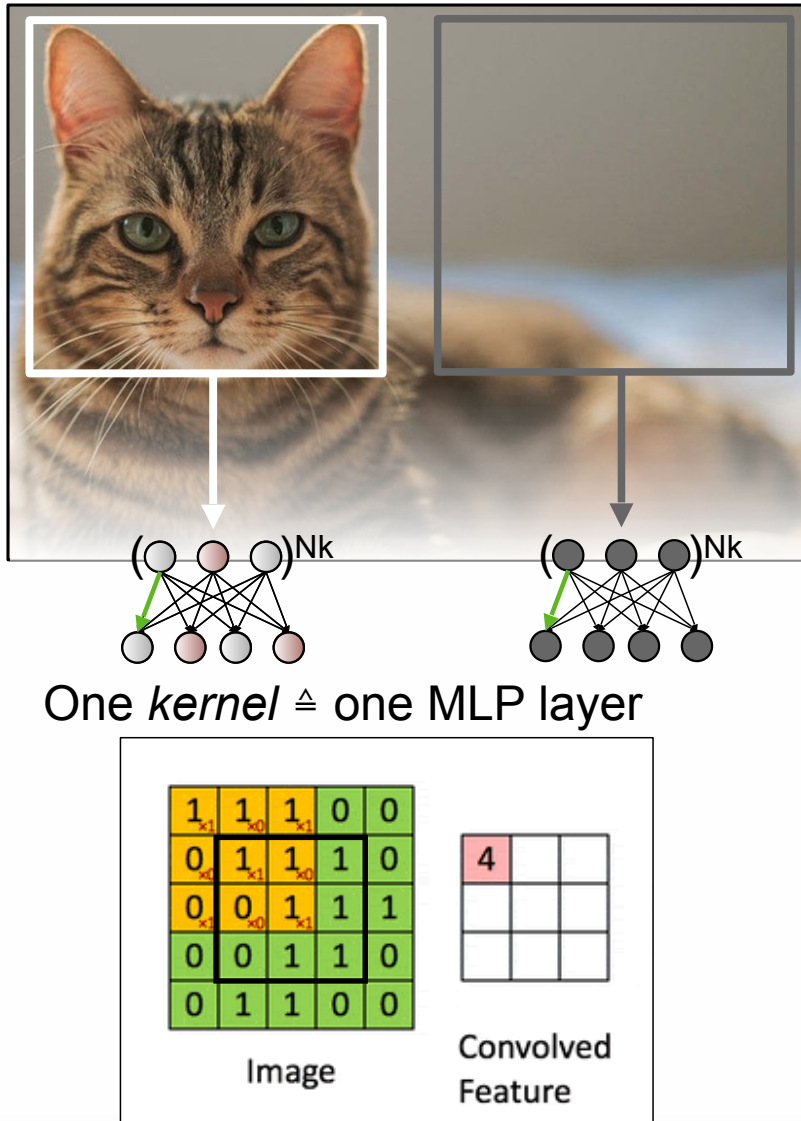
Learnable weights:
Relative position to j

Index m of the pixel on the i -th place in the neighbourhood of j

$$I(7,i) = \left\{ \begin{array}{l} +2 \text{ for full row} \\ +2 \text{ for full row} \end{array} \right\}$$

1	2	3
6	7	8
11	12	13

Multiple Input and Output Channels



- Inputs x
- For N_c input channels/features

$$\vec{y}_j = \theta \left(W \vec{X}_{I(j)} + \vec{b} \right)$$

Still strictly relative

$$\vec{X}_{I(j)} = \begin{bmatrix} \vec{x}_{I_1(j)} \\ \vec{x}_{I_2(j)} \\ \vdots \\ \vec{x}_{I_{N_k}(j)} \end{bmatrix}$$

- This is a complete convolutional layer

Translational equivariance as direct consequence



The convolution commutes with translations, meaning that

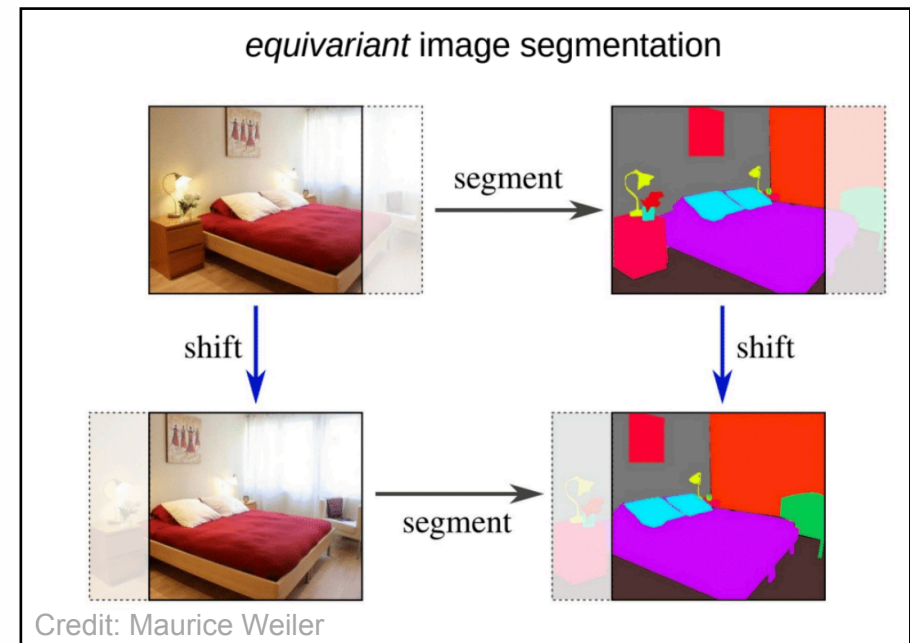
$$\tau_x(f * g) = (\tau_x f) * g = f * (\tau_x g)$$

where $\tau_x f$ is the translation of the function f by x defined by

$$(\tau_x f)(y) = f(y - x).$$

<https://en.wikipedia.org/wiki/Convolution>

- Convolutions and translation commute
- Shift + convolution is the same as convolution + shift
- This is referred to translation **equivariance** (not invariance)



Cats (still) come in different shapes



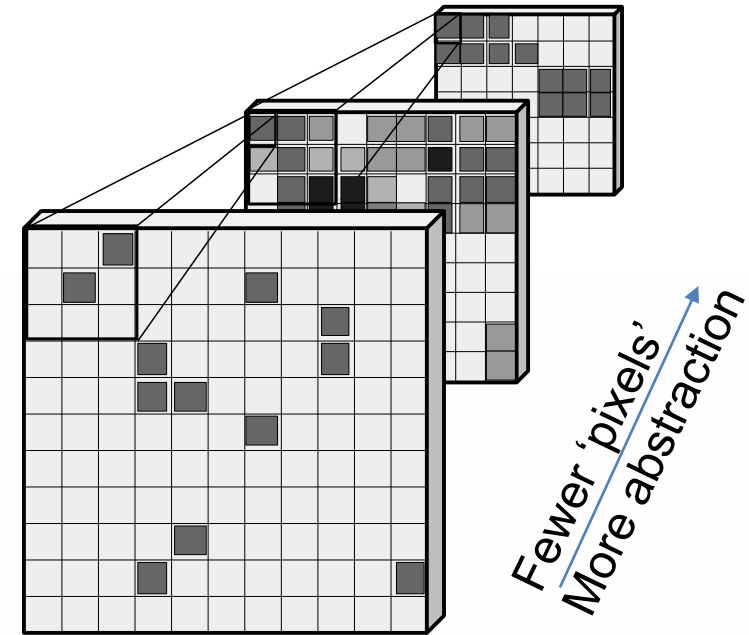
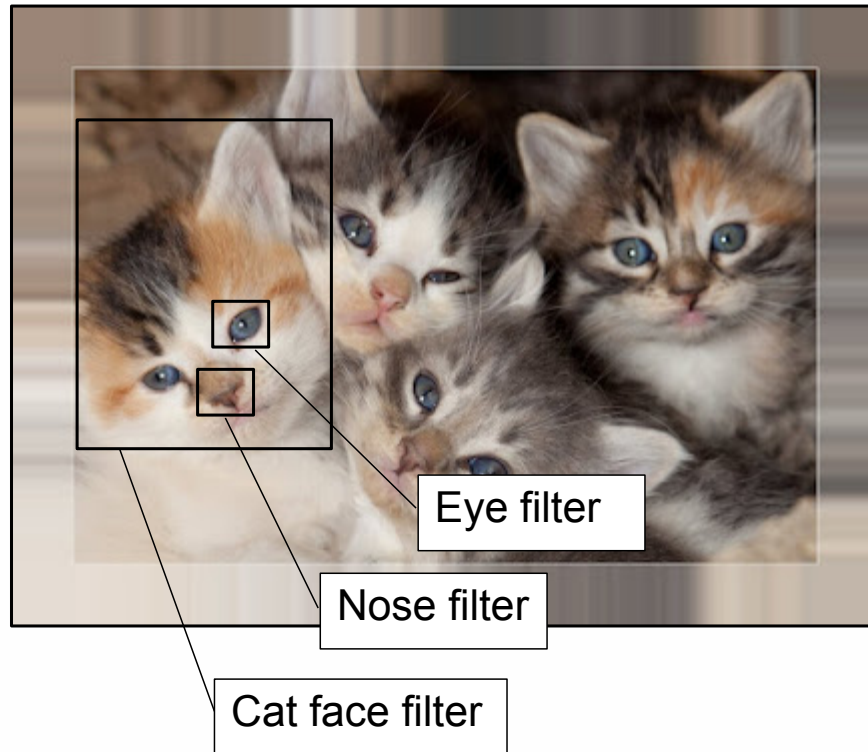
Not a cat



Not a cat

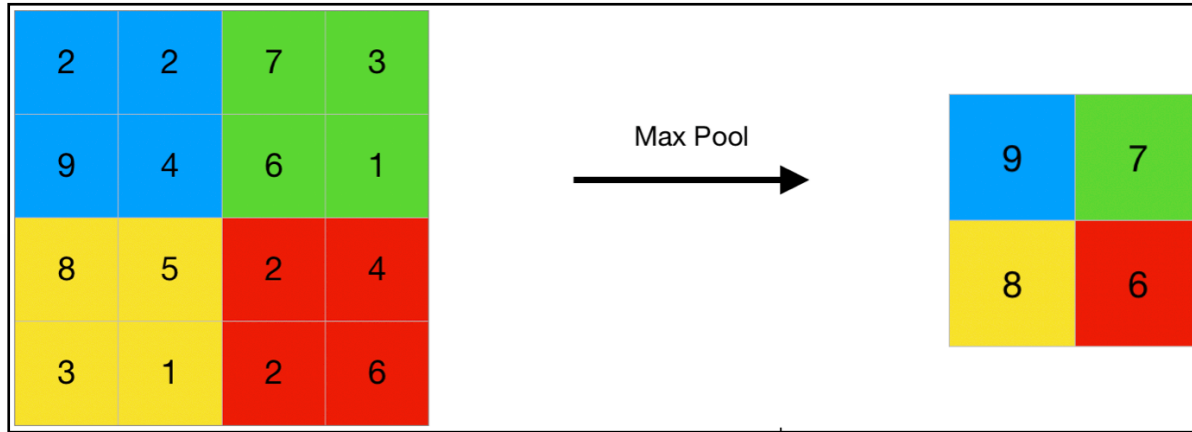
- Many different very complex filters are needed
- Can be solved by
 - Learning filters from examples ✓
 - **Abstraction**

Abstraction



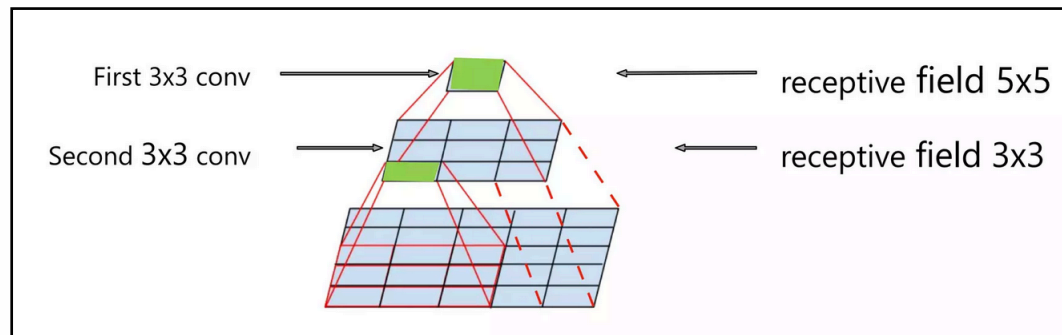
- Use smaller kernels to capture individual features
- Summarise the filter outputs of several neighbouring pixels

Pooling as One Way of Building Summaries



<https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>

- Summarises information from neighbouring pixels
- Increases receptive field and incentivises abstraction



CNNs are very powerful: fewer parameters

- CNNs break down the large number of input pixels with a **much** smaller number of parameters



- The filter weights are shared for all j
$$\vec{y}_j = \theta \left(W \vec{X}_{I(j)} + \vec{b} \right)$$
- They are trained for **every** y_j :
 - ω 'see' (sample size $\cdot$ number of pixels) effective training examples
- There are (almost) always **multiple** benefits from using the structure of the data

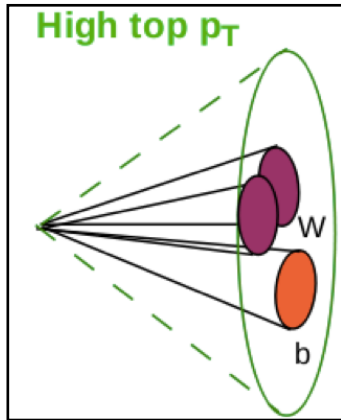
Inductive Bias in CNNs

Grid like structure with objects composed of local hierarchical patterns

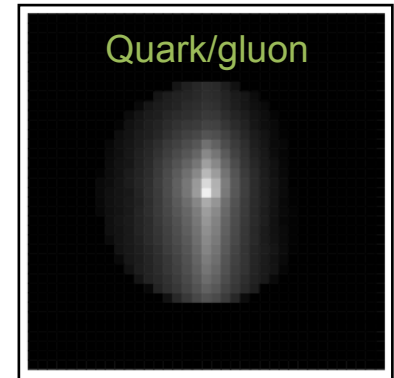
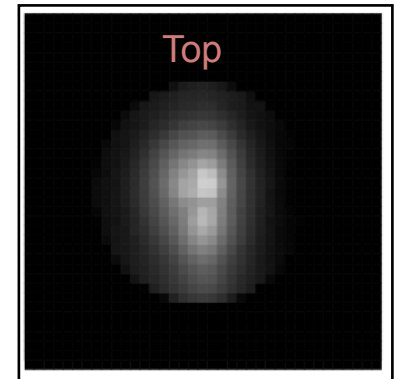
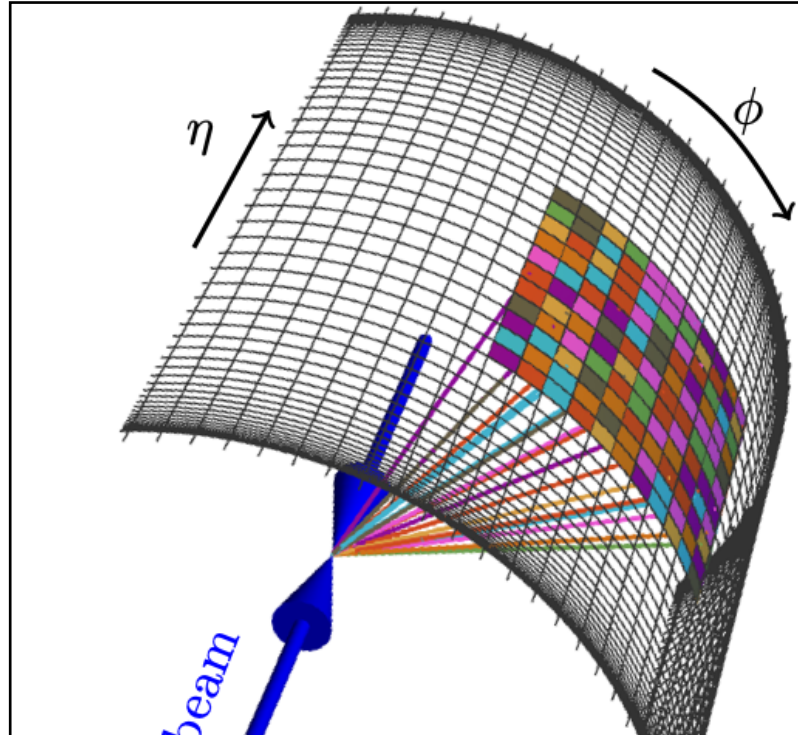
- Exact locality
- Weight sharing
- Translation equivariance
- Hierarchical composition & aggregation



Jets as images

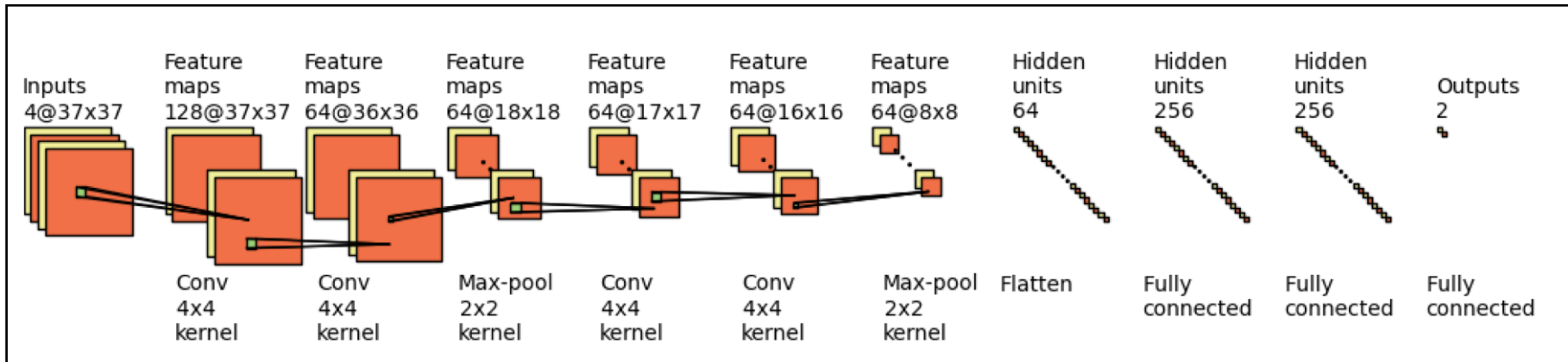


arxiv:1803.00107
(and many others)



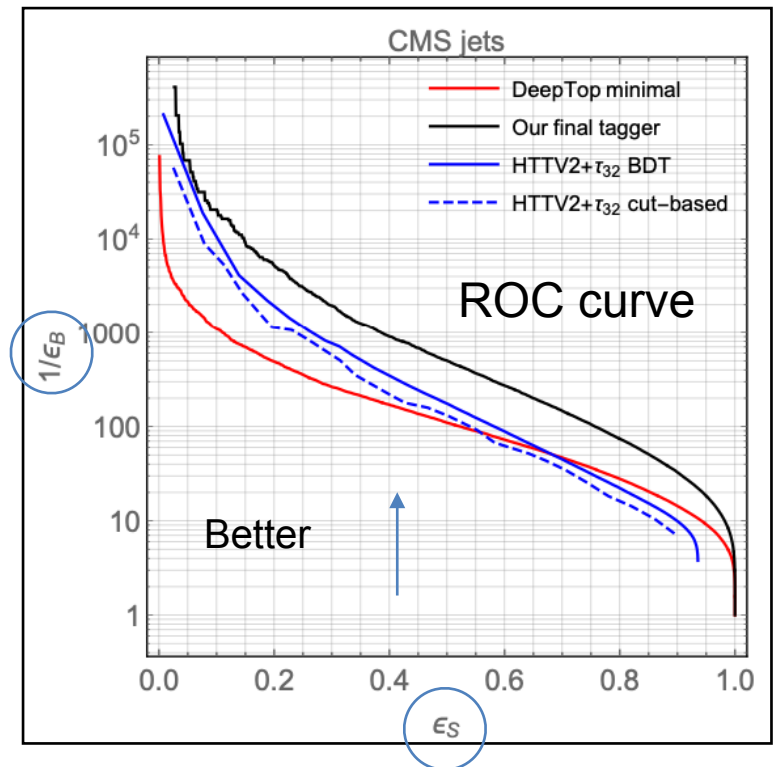
- Identifying origin of a jet very useful for many physics analyses
- Treat the jet deposits (e.g. in the calorimeter) as an image

Physics examples: jet tagging

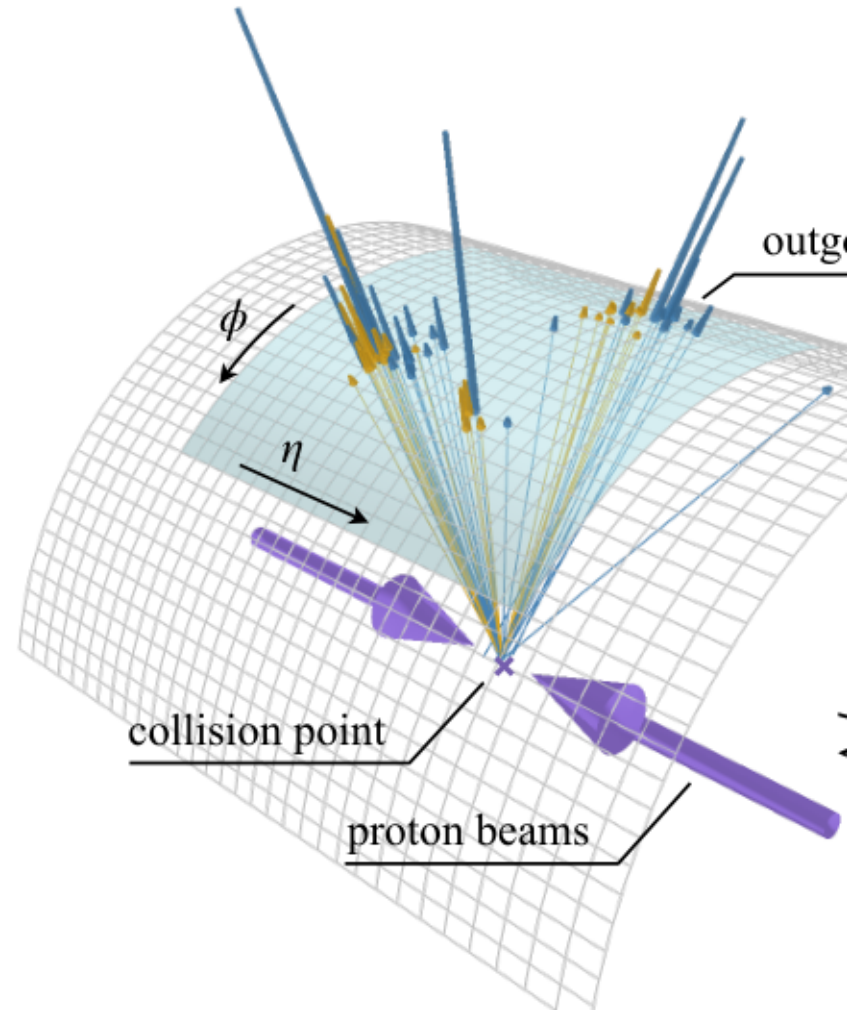
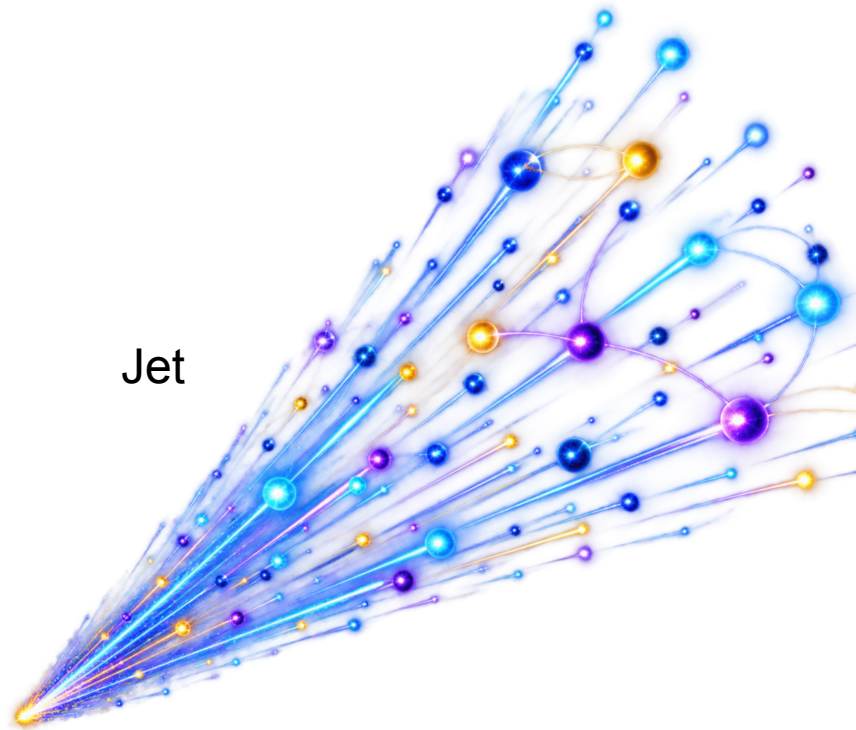


arxiv:1803.00107
(and many others)

- Typical CNN architecture
- Performance gain over an MLP trained with human-engineered high-level variables



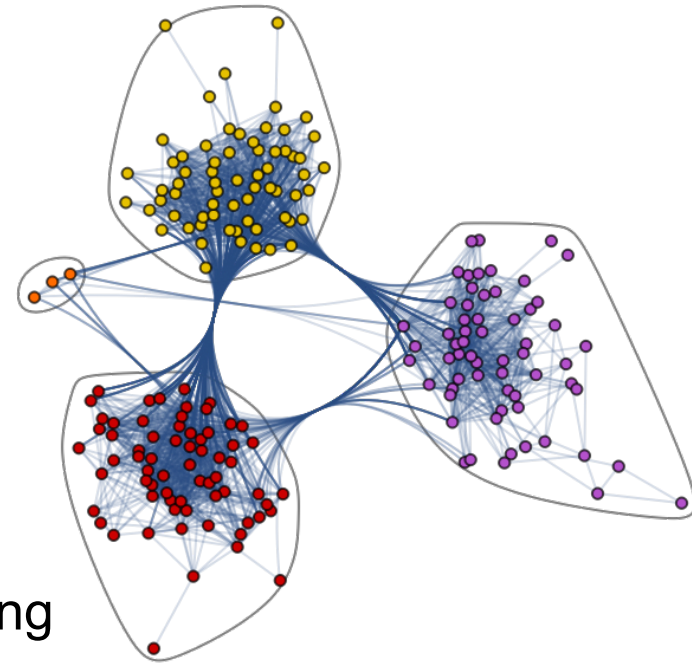
Learning Relationships



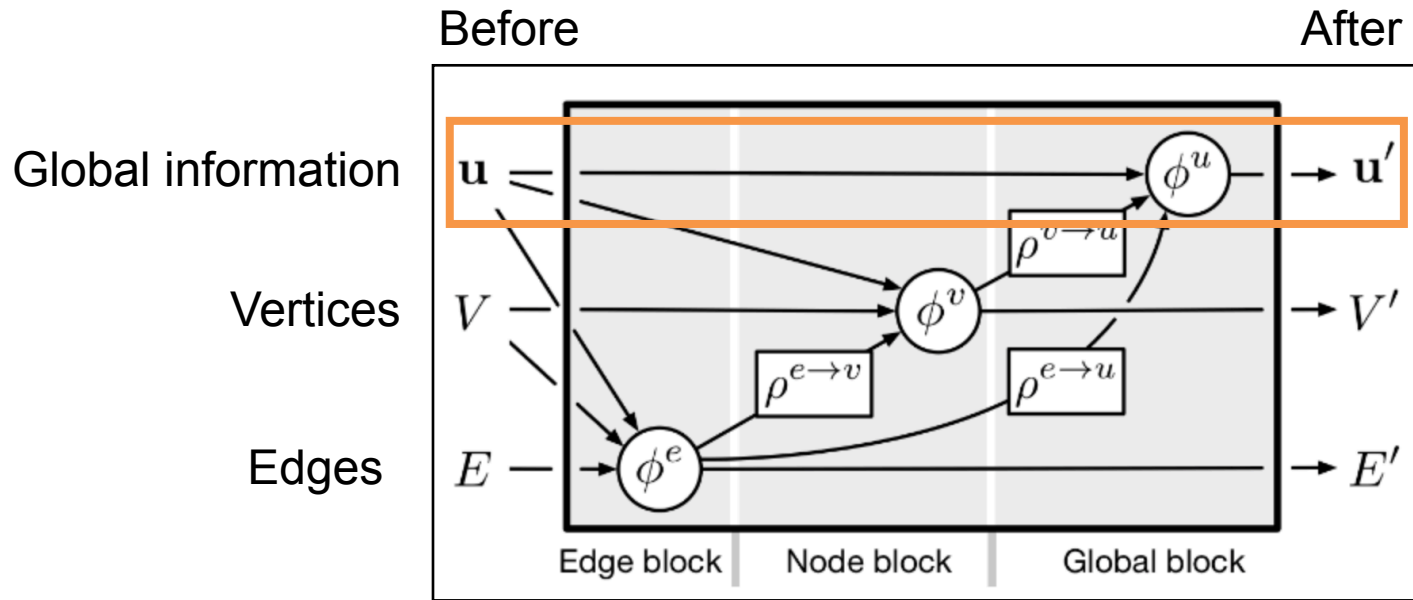
H. Qu, et al., arXiv:2202.03772

Particles and Their Relations for a Graph

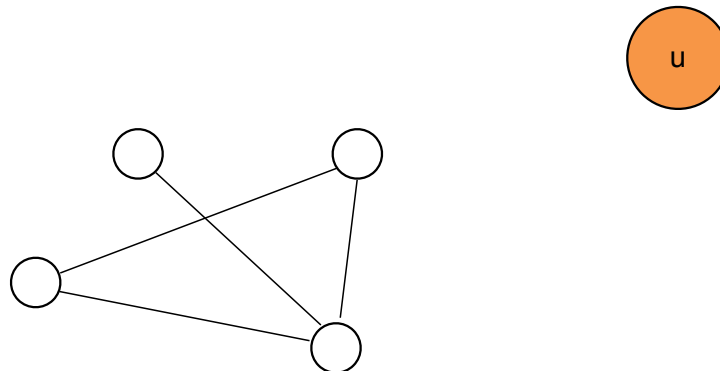
- A graph is a collection of vertices and edges connecting them
- Vertices can have properties
- Edges can have properties, and direction
- Graphs are a very generic tool to describe “things” and “relations between them”
- Node ordering does not carry physical meaning
- No grid
- No fixed size



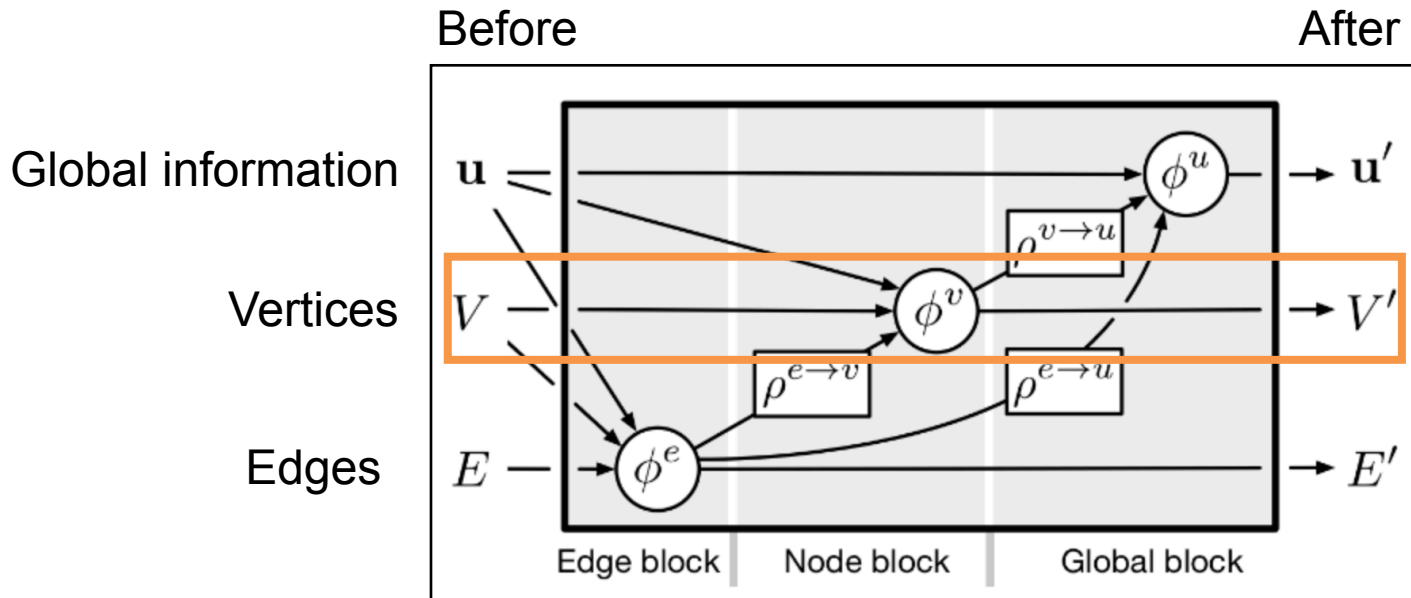
Information exchange in a graph



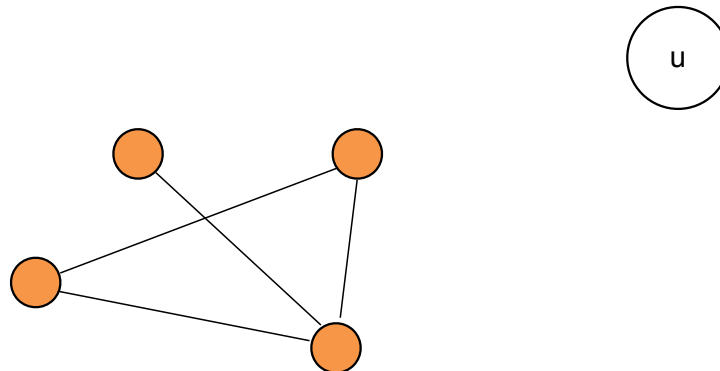
arXiv:1806.01261



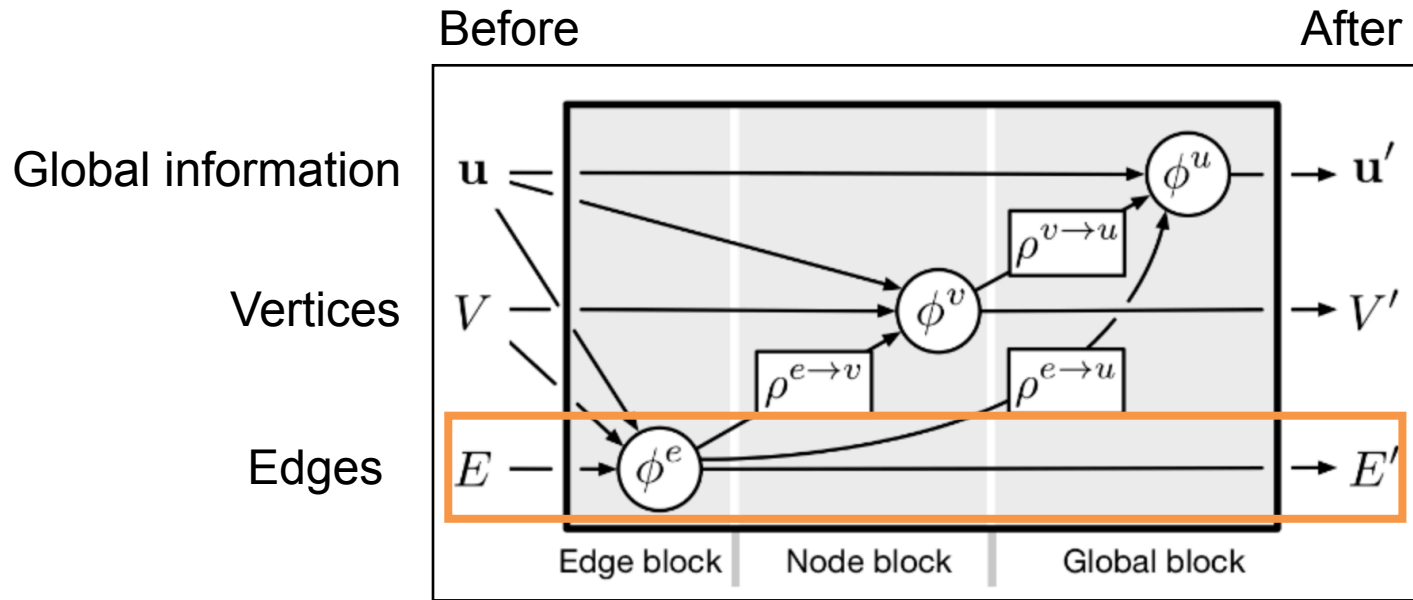
Information exchange in a graph



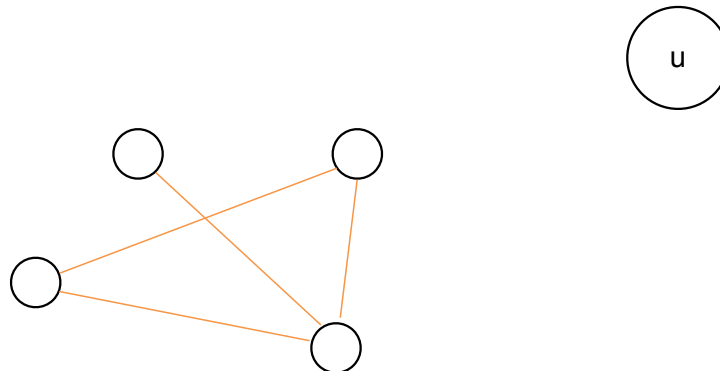
arXiv:1806.01261



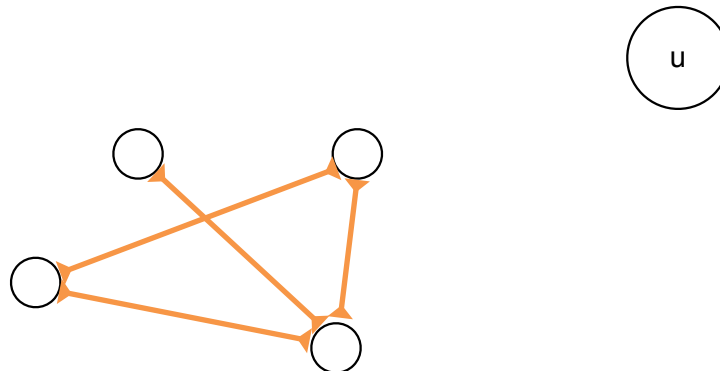
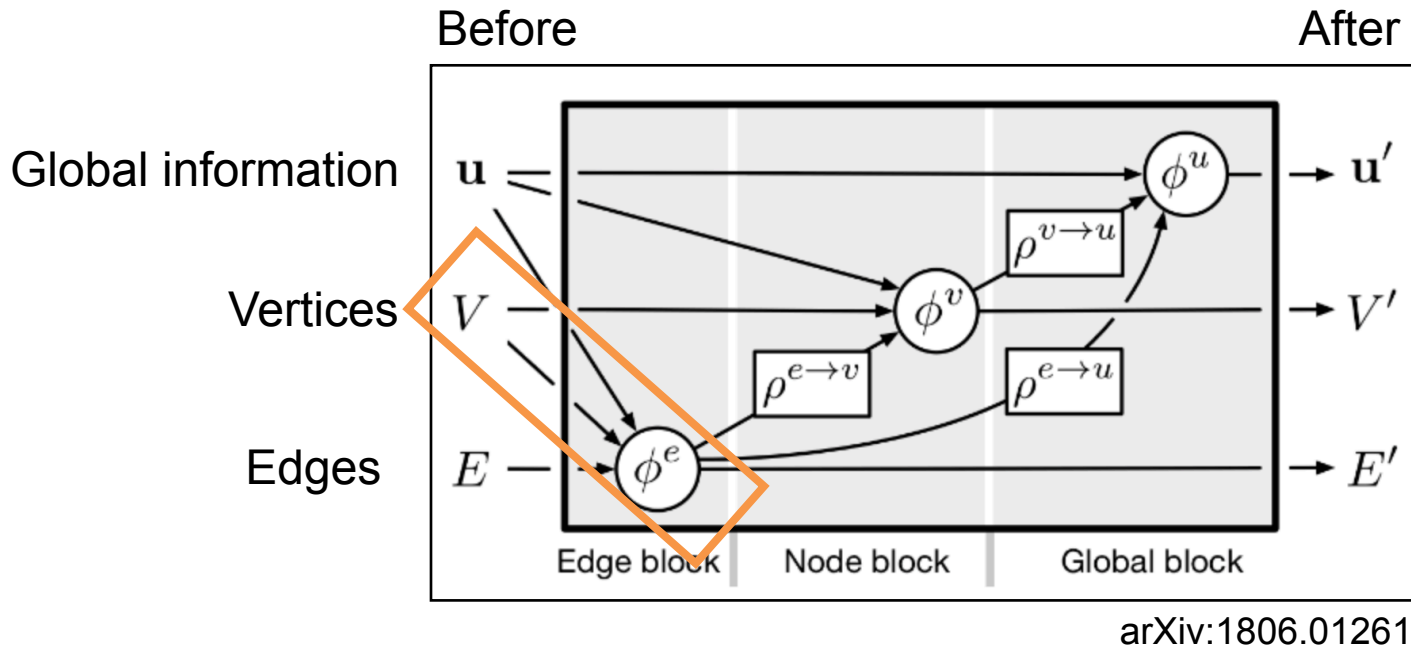
Information exchange in a graph



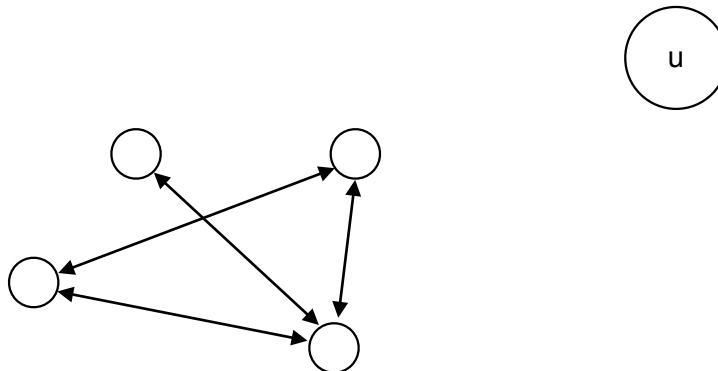
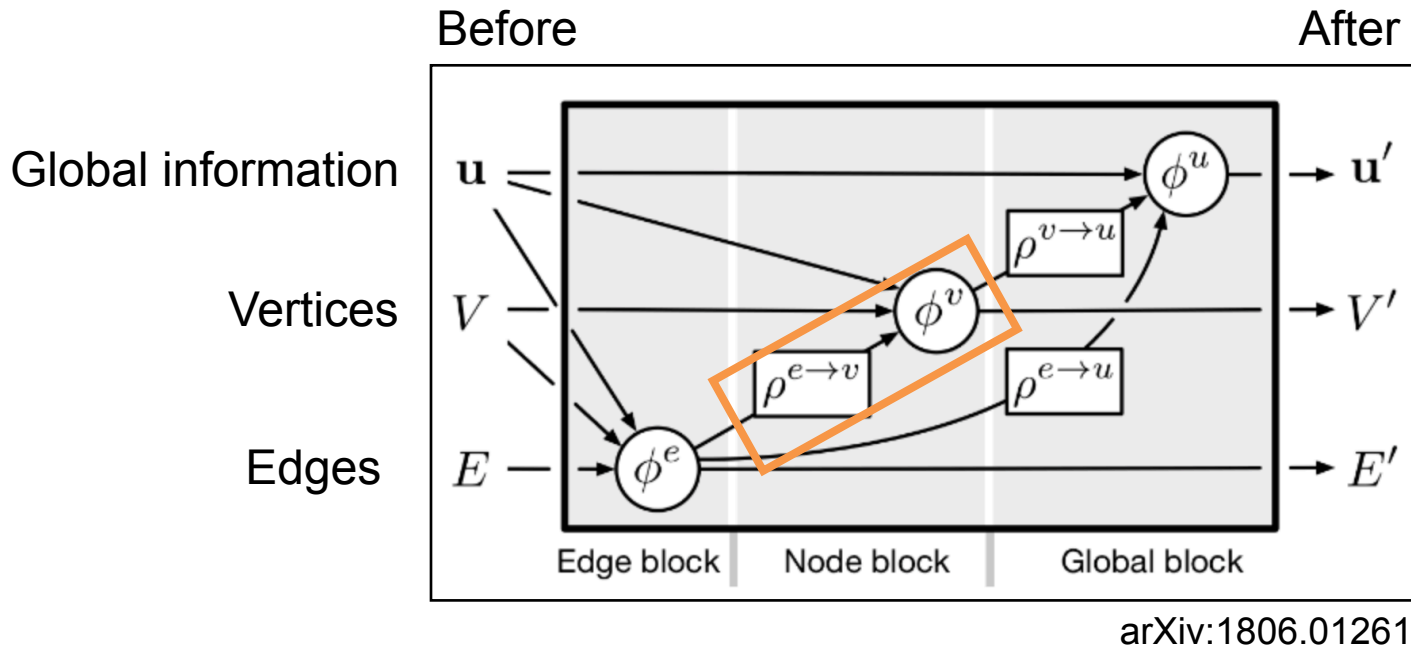
arXiv:1806.01261



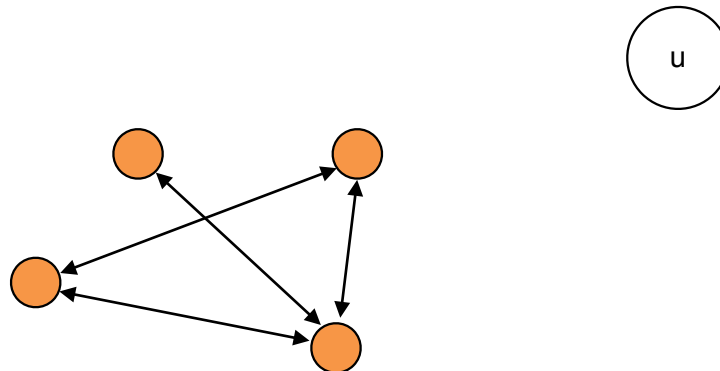
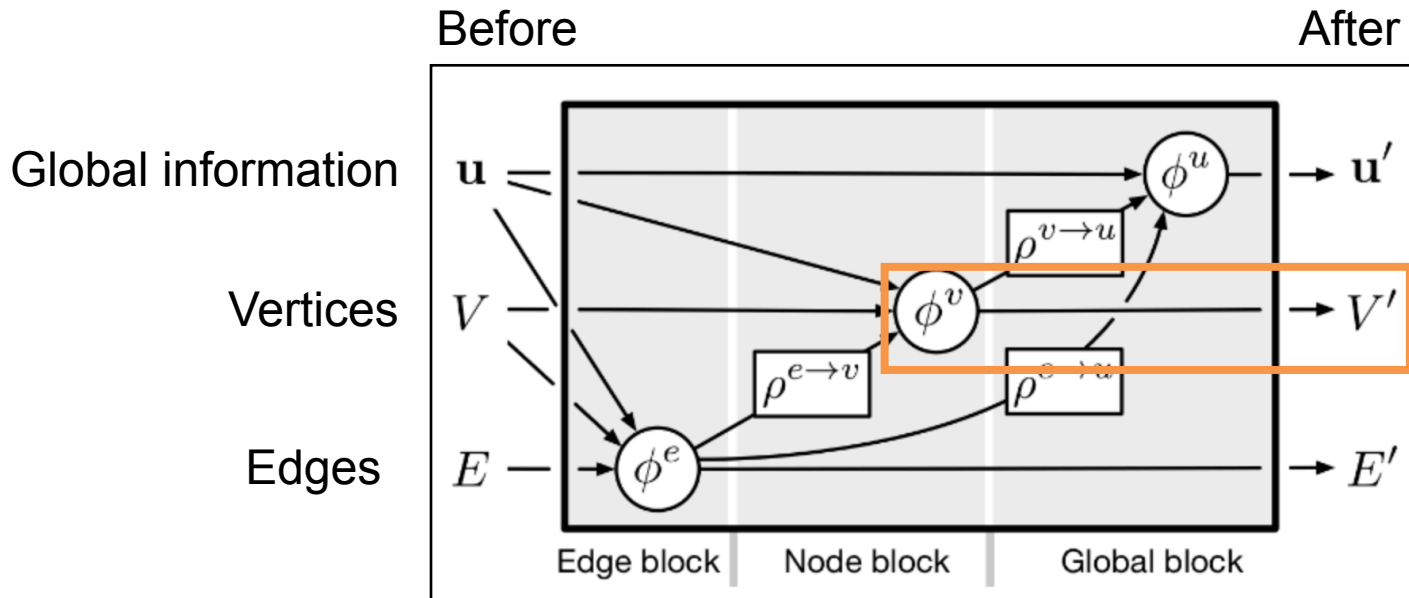
Information exchange in a graph



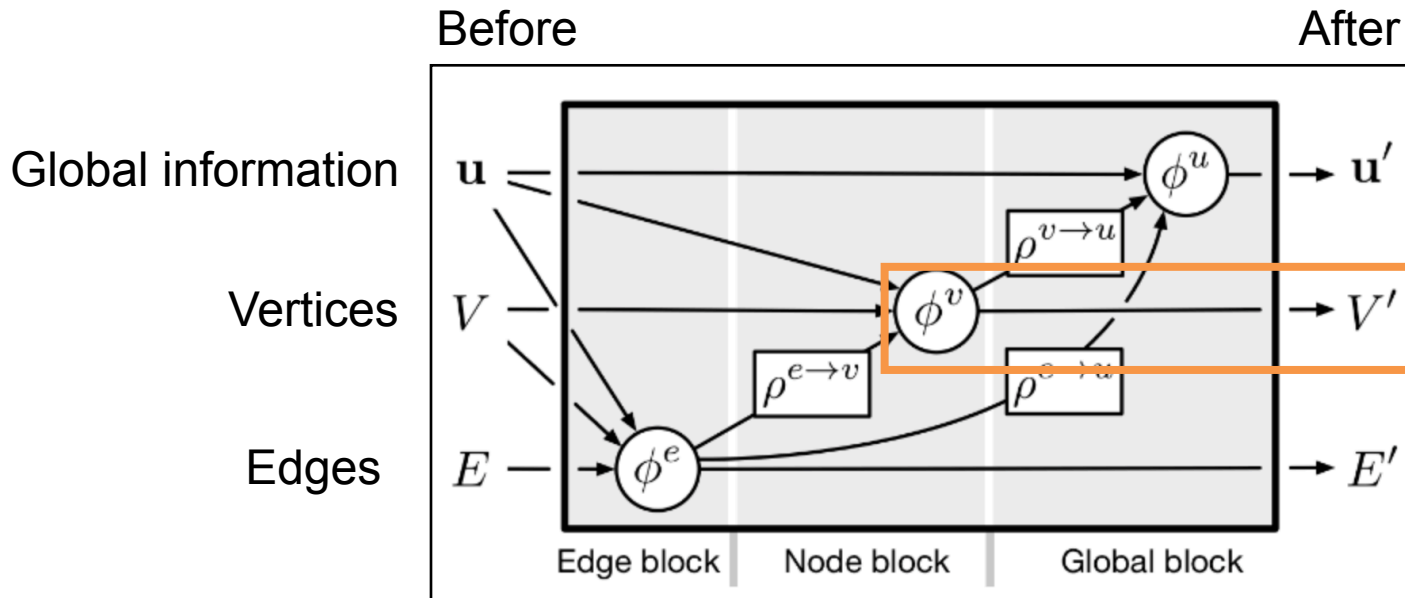
Information exchange in a graph



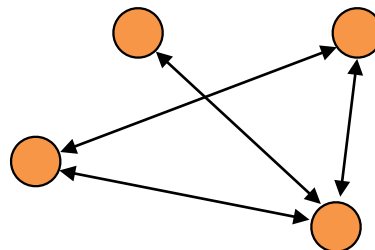
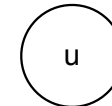
Information exchange in a graph



Information exchange in a graph



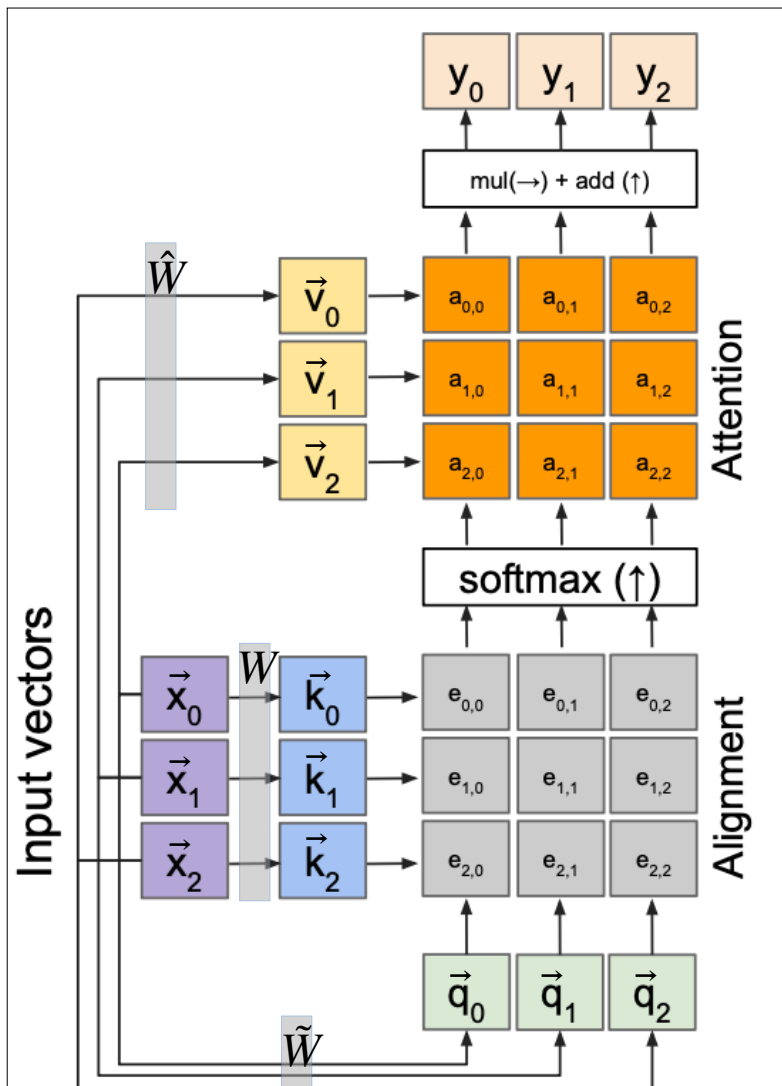
arXiv:1806.01261



This is a message passing,
the most common form of
information exchange in a graph

$$\vec{y}_j = \bigoplus_{i \in N(j)} M(x_i, x_j, e_{ij})$$

Self-Attention: a special form of message passing



- The node features are transformed to

- Values: $\hat{W}\vec{x}_i = v_i$
- Keys: $W\vec{x}_i = k_i$
- Queries: $\tilde{W}\vec{x}_i = q_i$

Fully connected

$$\vec{y}_j = \sum_i^V a_{ij} \hat{W}\vec{x}_i = \sum_i^V \text{Message} a_{ij} \vec{v}_i$$

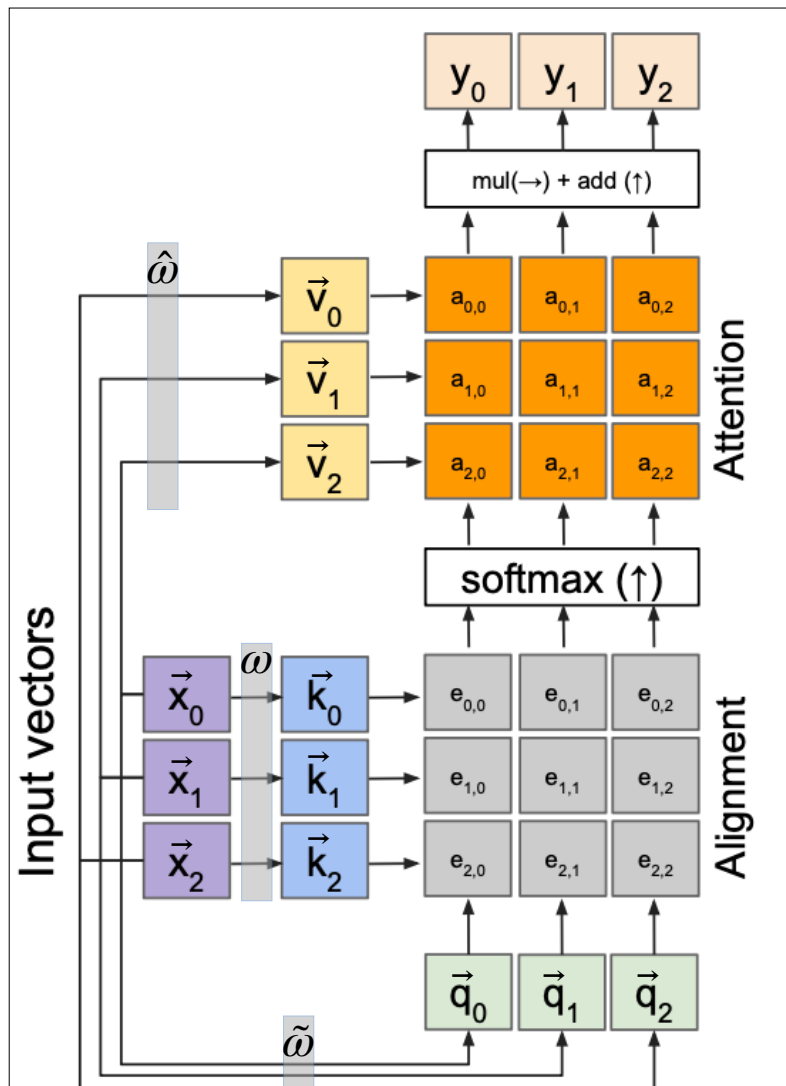
$$a_{ij} = \sigma_i \left(\frac{k_i \cdot q_i}{\sqrt{d_k}} \right)$$

Softmax

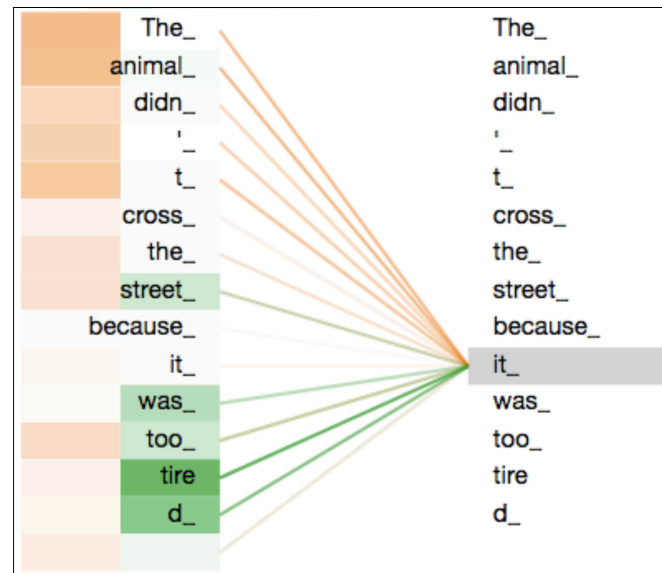
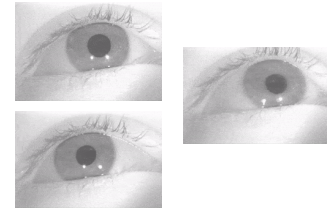
Learnable,
creates keys

Learnable,
creates queries

Multi-head attention



- Several parallel message functions can learn different kinds of particle relationships.

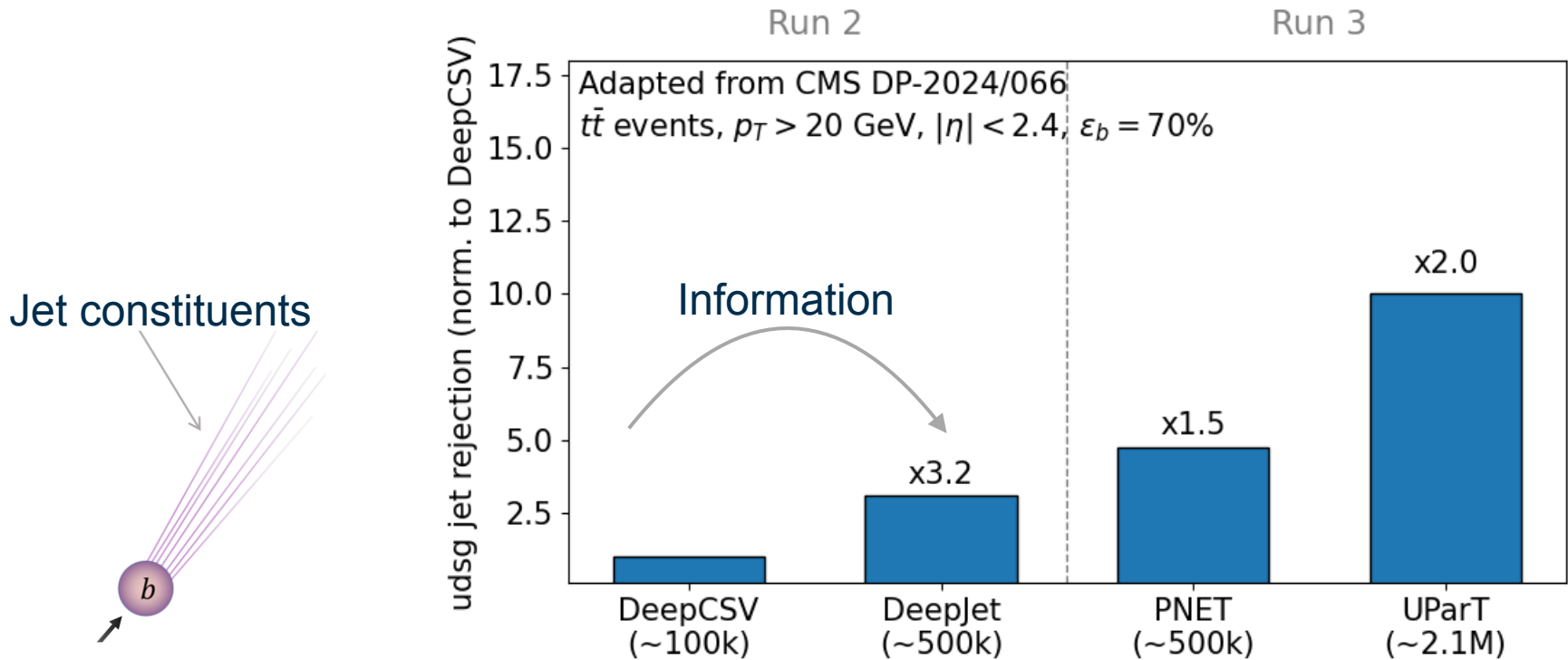


- Simply create multiple heads and concatenate their output:

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

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

Using the Structure has Huge Pay-Offs



- In some cases, the physics gain* is equivalent to decades of data taking

*confirmed in data

NB: Also attention is the building block of LLMs

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

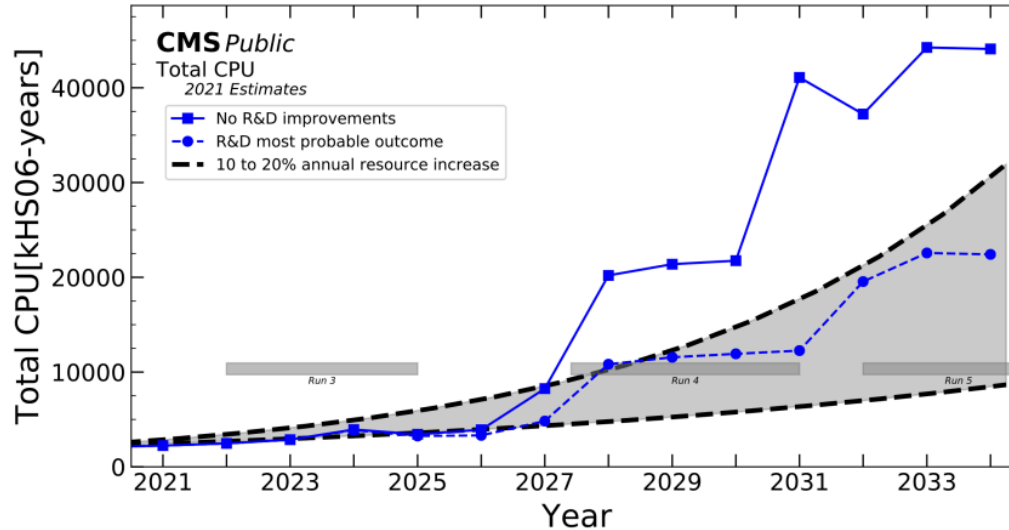
Łukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin* †
illia.polosukhin@gmail.com

>80k citations!

Why ML (2): Ressources

- ML can bring significant gain or physics performance

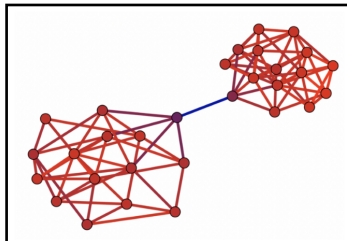
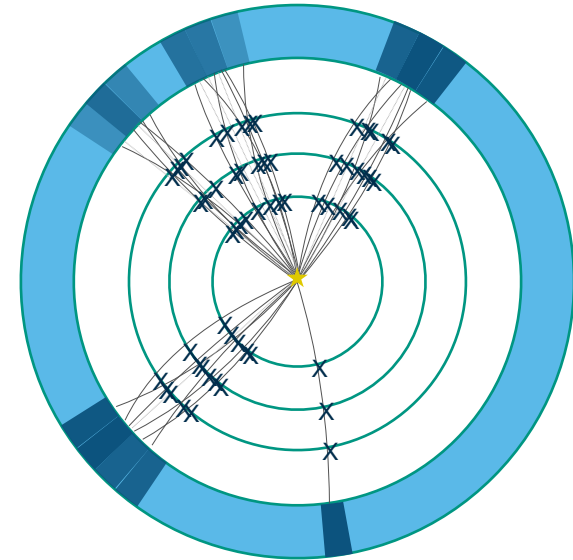


CMS NOTE -2022/008

- ML algorithms open the option to lower resource requirements

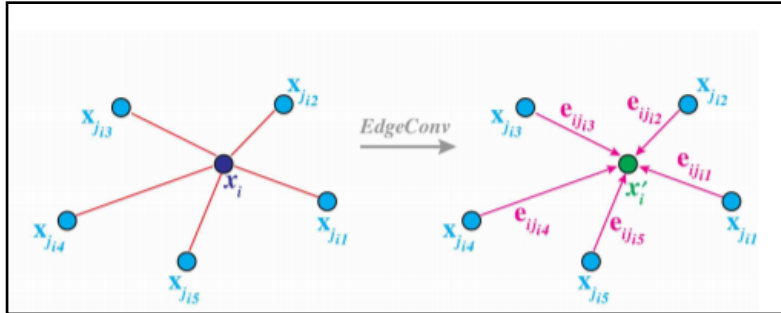
Creating Edges for Reconstruction

- Fully connected graphs (like attention) do not scale well
- They don't encode which relationships are likely to matter
- Connect in physical space - everything within a radius?
 - Can lead to weak connections (or too many)



Dynamic Graphs

DGCNN

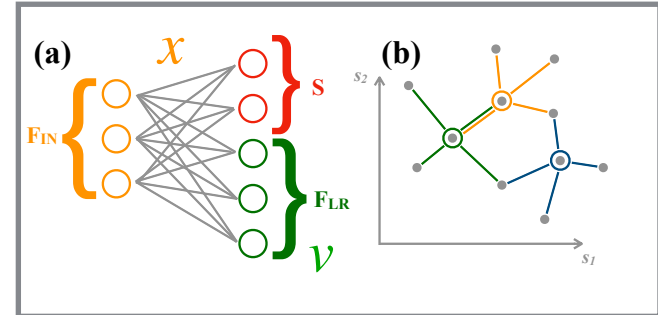


$$\vec{y}_j = \bigcap_{i \in N(j)} \Phi(x_j, x_j - x_i)$$

- Use full feature space to define neighbourhoods
- Learn best connections through feature-similarity
- Computationally expensive

arXiv:1801.07829

GravNet

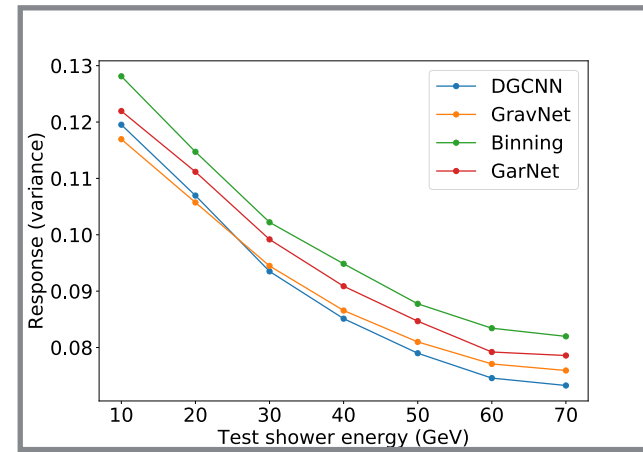
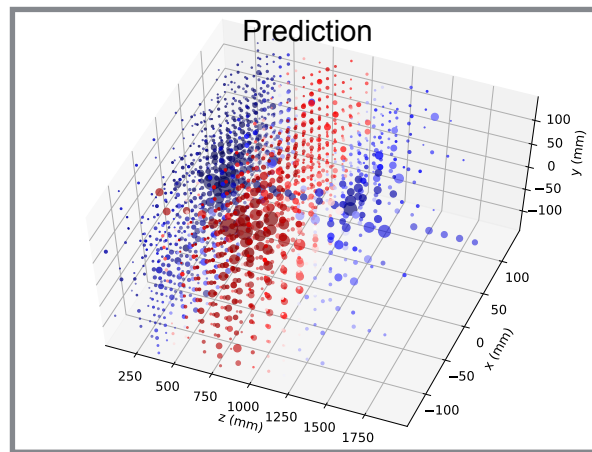
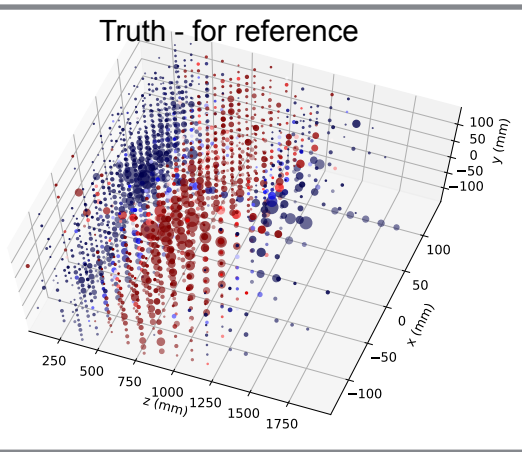


$$\vec{y}_j = \bigcap_{i \in N(j)} \exp\left(-(\vec{s}_i - \vec{s}_j)^2\right) \cdot \vec{v}_i$$

- Use low-dimensional projection S to define neighbourhoods
- Learn best connections through explicit weights
- Scales well to a lot of inputs

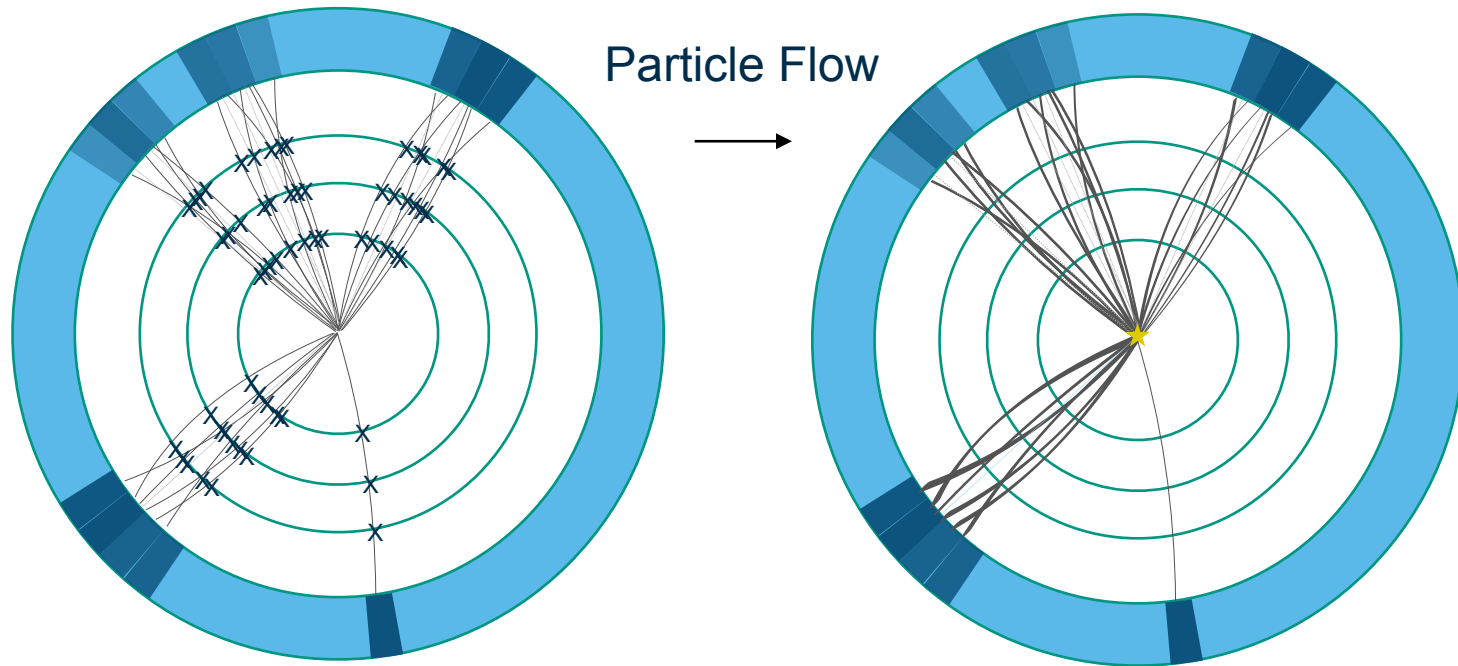
arXiv:1902.07987

Physics Performance



- Predict fraction of energy belonging to one of two showers in simple calorimeter
- GravNet & DGCNN show similar physics performance
- GravNet uses 10-100x fewer resources.

Particle Reconstruction: Output Structure also Matters



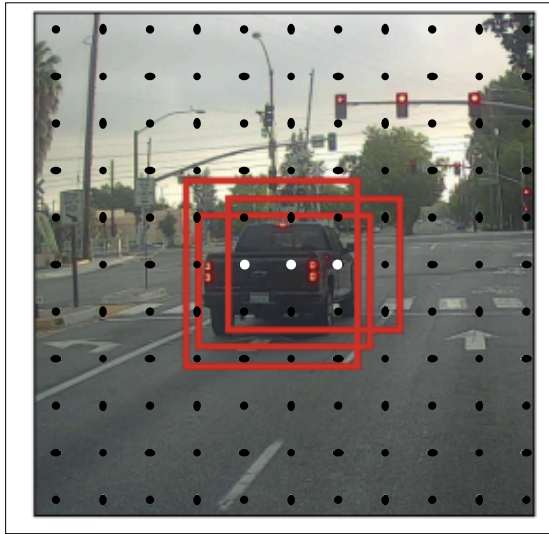
Object reconstruction / detection problem:

V inputs

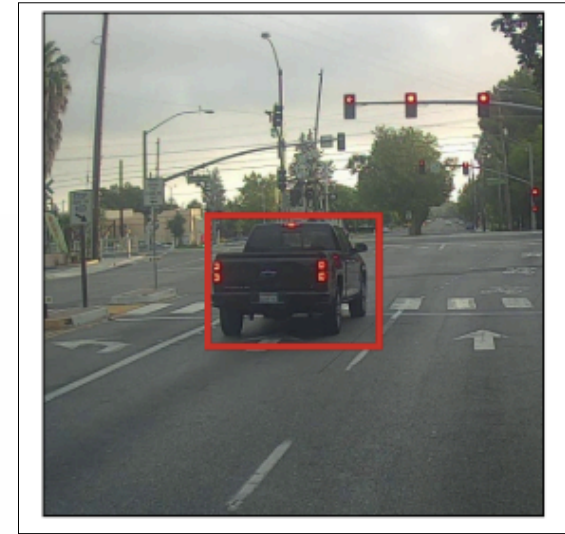


M outputs

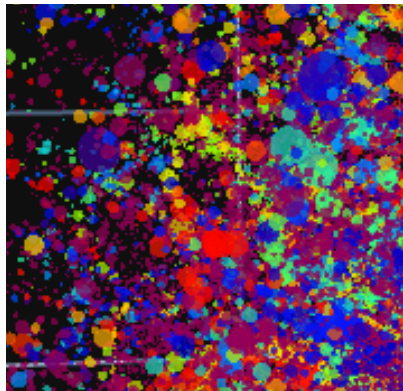
A look at Computer Vision



Overlap removal
→



- Representative points and subsequent overlap removal



- Not directly applicable to HEP problems

How to reconstruct multiple objects (very brief)

- Two particles leave hits

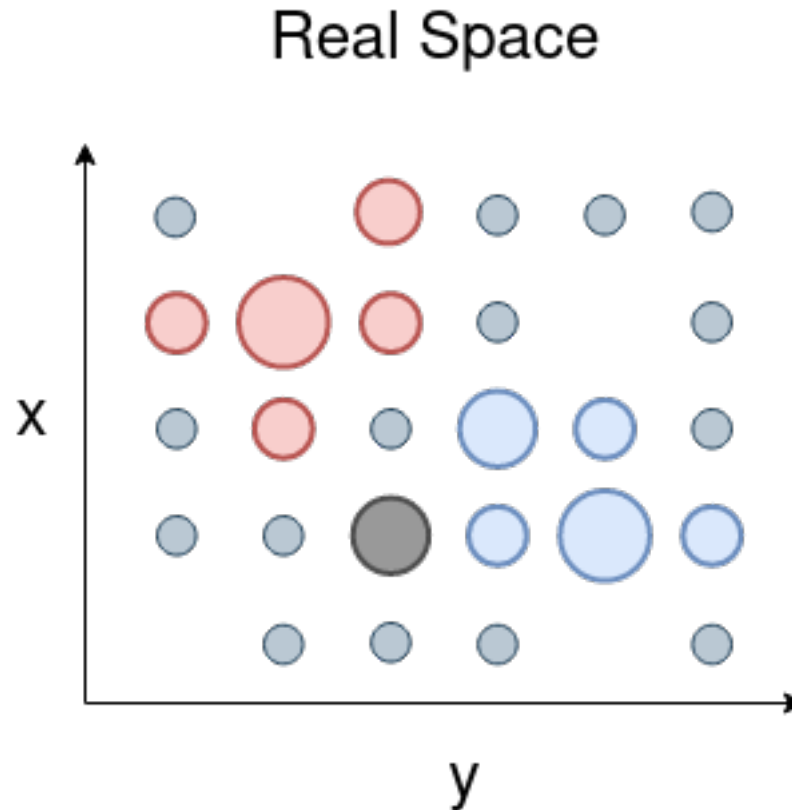
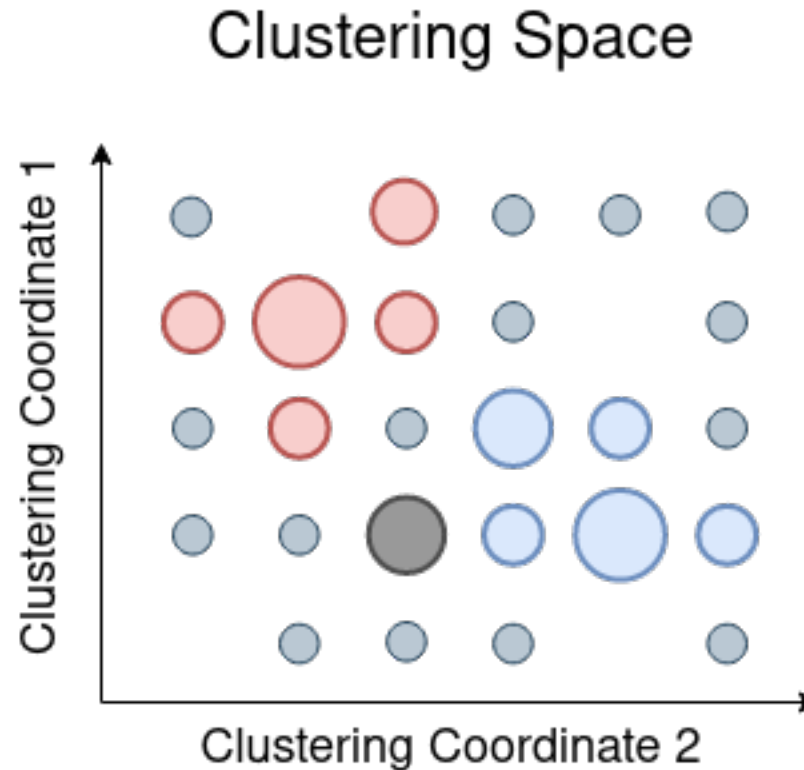


Image credit: Lea Reuter, Isabel Haide

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 Kieseler¹
(jan.kieseler@cern.ch)

¹CERN, Experimental Physics Department, Geneva, Switzerland

[arxiv:2002.03605](https://arxiv.org/abs/2002.03605)

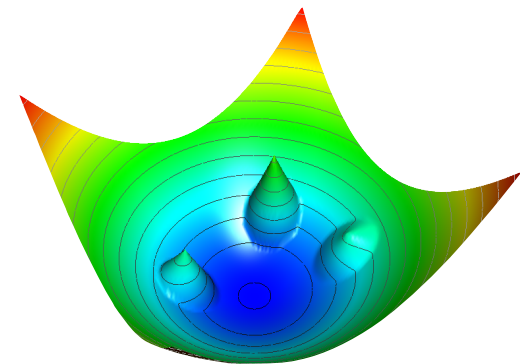
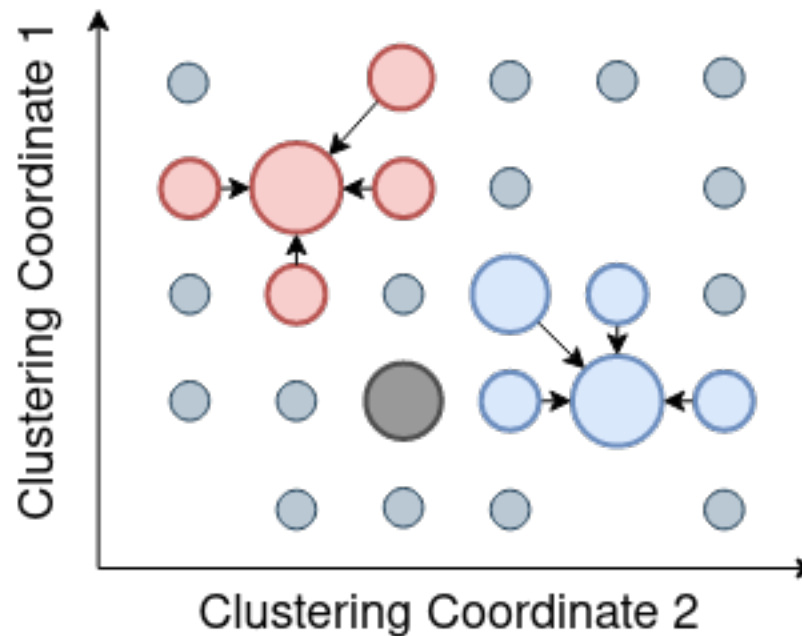
Image credit: Lea Reuter, Isabel Haide [KIT]

Make the network learn to reconstruct objects

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

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

Attractive Potential

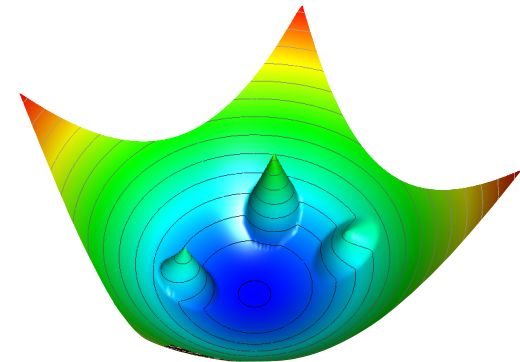
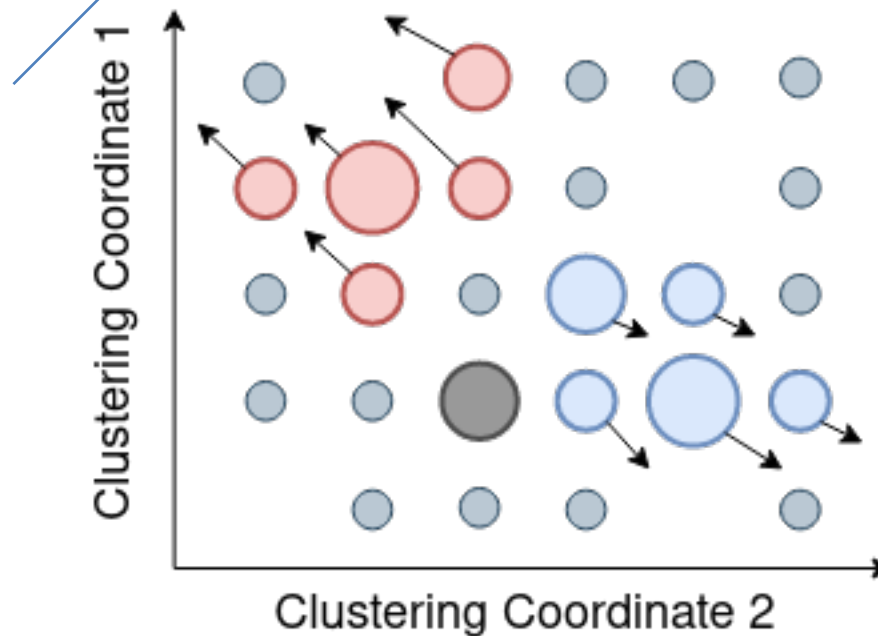


Separate different objects

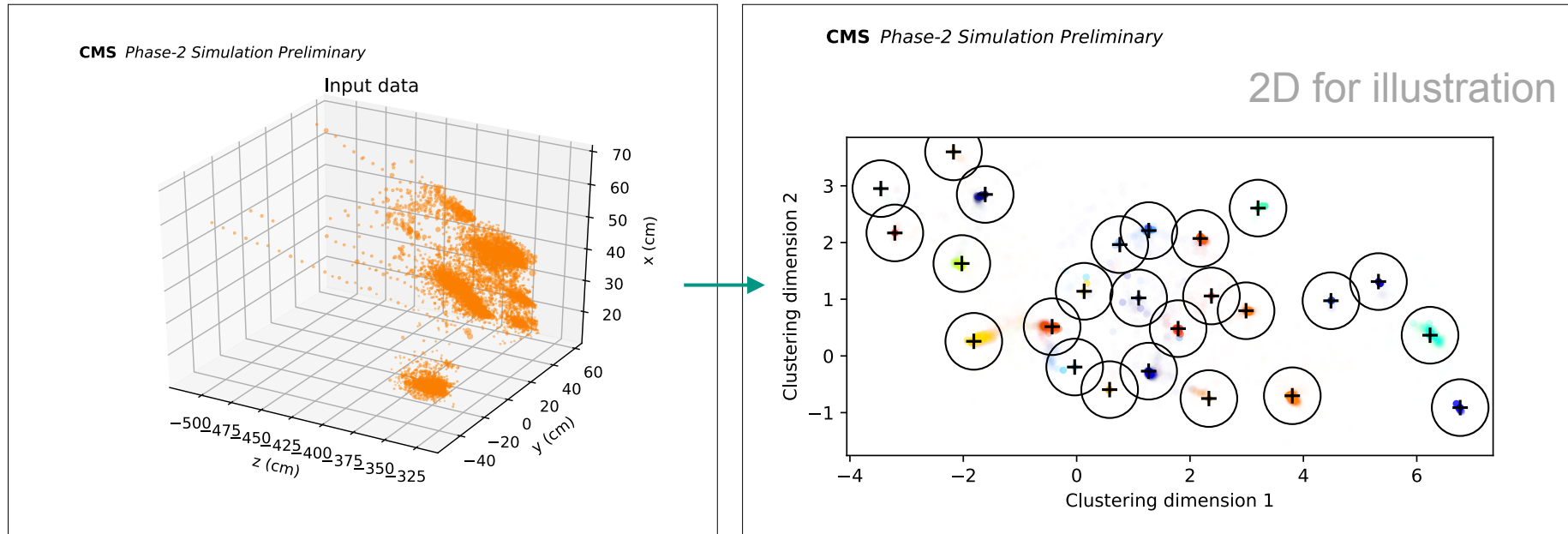
- Write a loss function (Object Condensation) that repels points from other objects

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

Repulsive Potential



Simplifying the Problem



- A complex clustering problem becomes a simple clustering problem

Also learn properties on one go

- Learn representative points α (high beta score)
 - Will be in the center of the cluster (in clustering space)
 - Will contain best property estimate

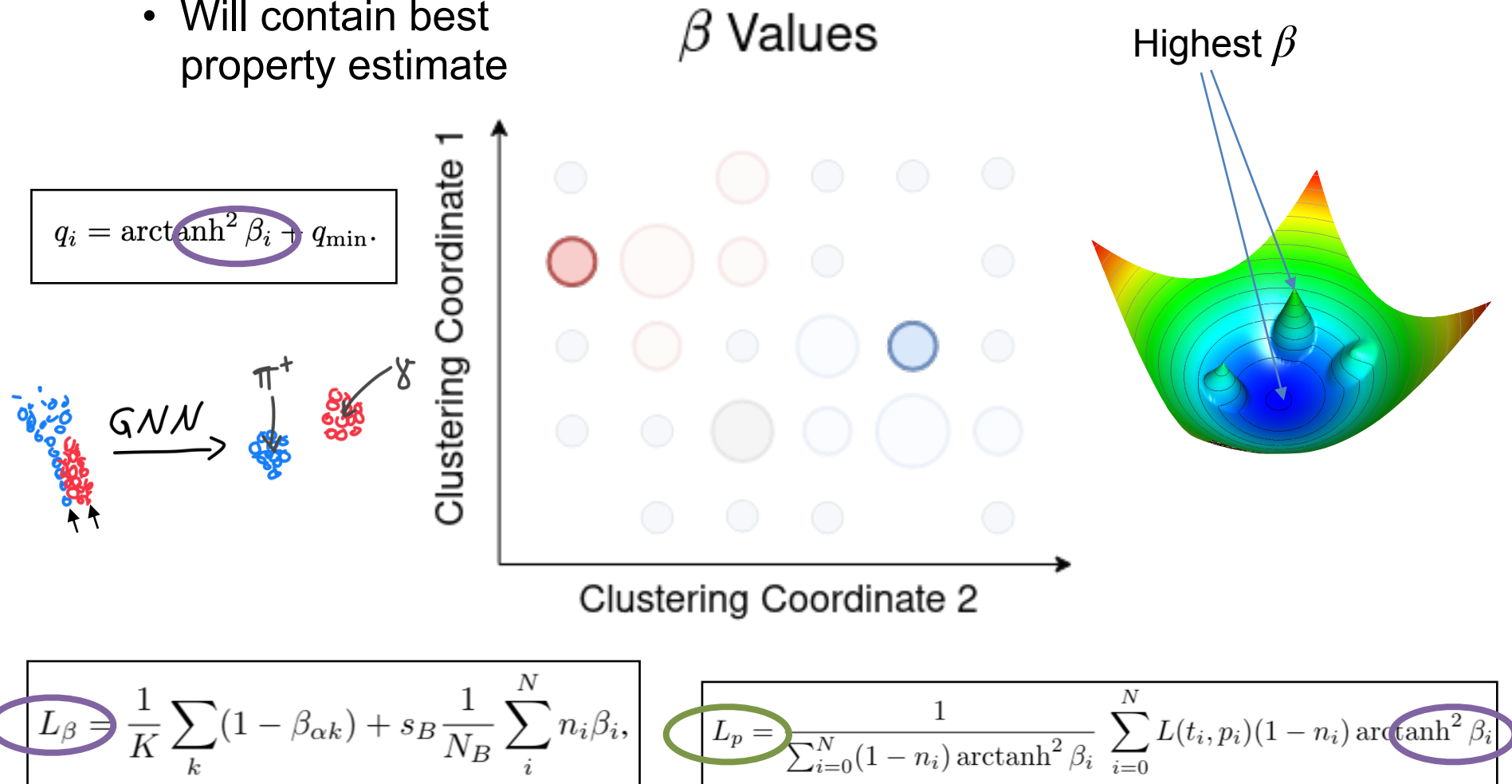
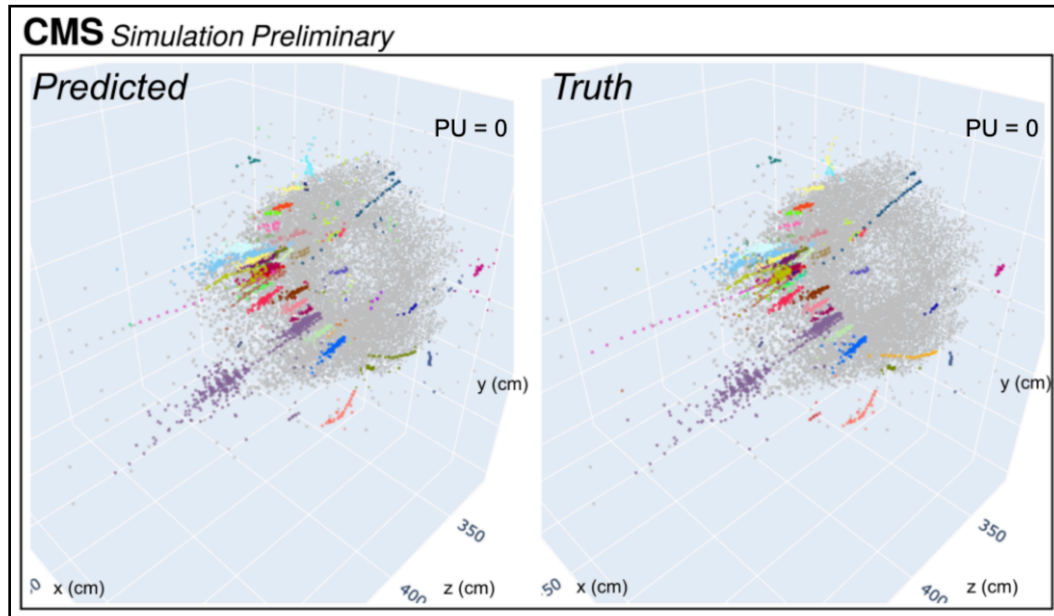
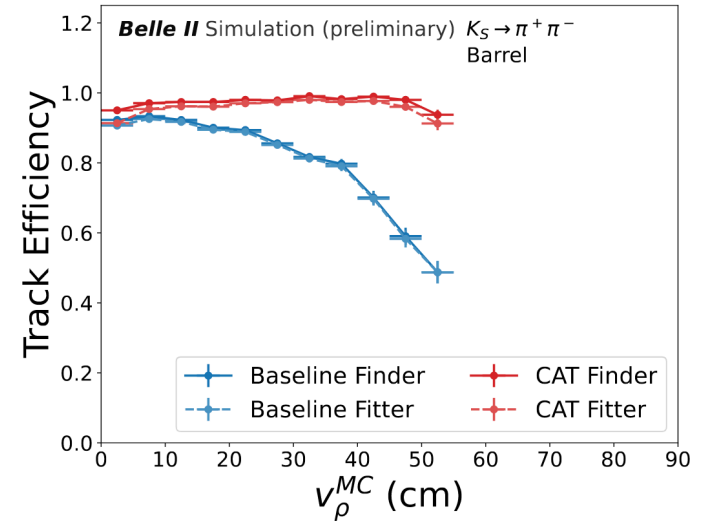


Image credit: Lea Reuter, Isabel Haide

Physics Performance and Adaptation



CMS-DP-22-001



(d) $K_S^0 \rightarrow \pi^+ \pi^-$, v_ρ^{MC} .

L. Reuter et al., CSBS (2024), arXiv:2411.13596

[*]

G. Matousek, A. Vossen, arXiv:2503.11277

T. Suehara, et al., arXiv:2410.08772

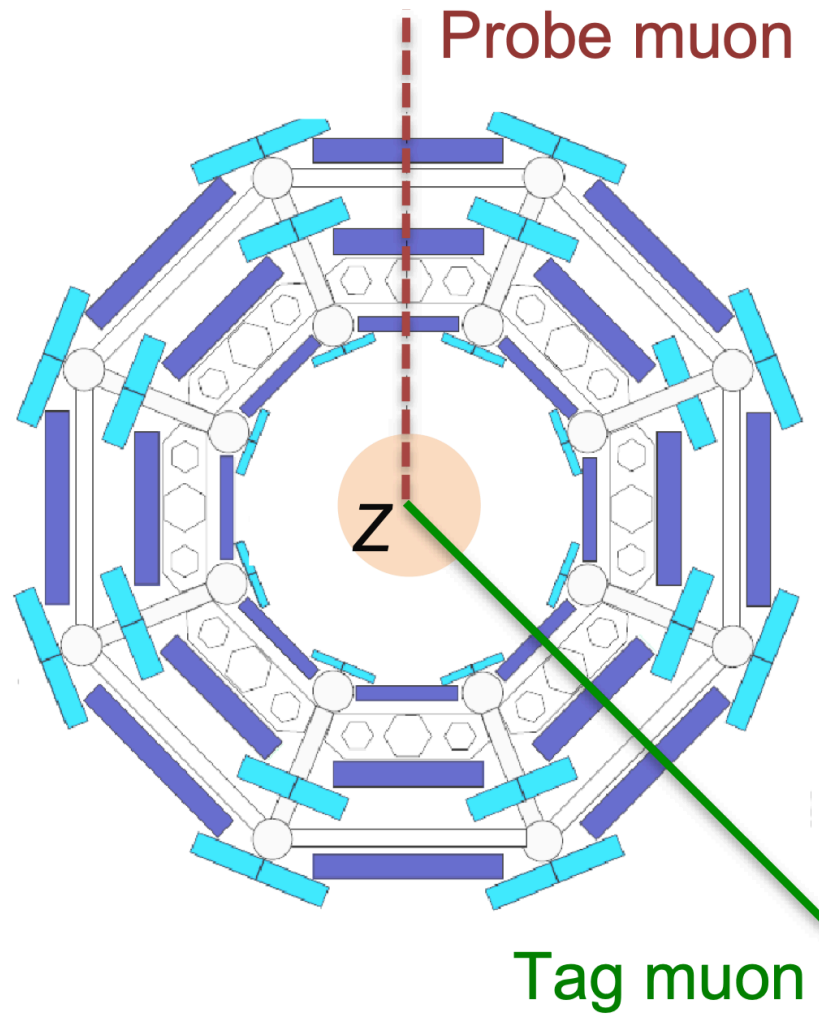
D. Garcia, et al., NeurIPS 2023

S. Qasim, et al., arXiv:2204.01681

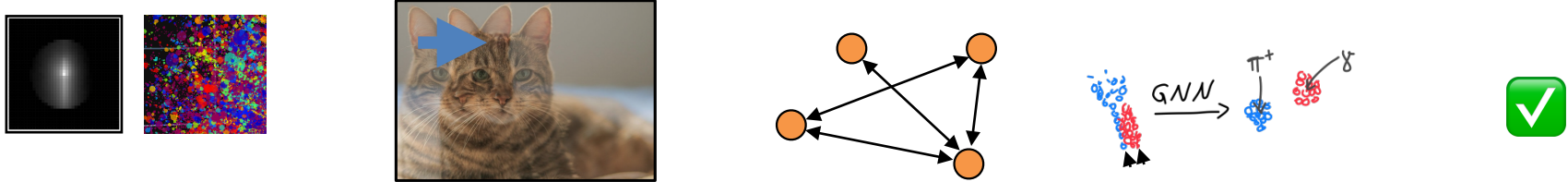
CMS, arXiv:2203.01189,

S. Qasim, et al., arXiv:2106.01832

Calibration



DIY conclusions



- Finding structures and symmetries can help reduce parameter counts and make computation cheaper
- CNNs find relations between neighbouring pixels and abstract them
- Reconstruction tasks in HEP are similar to object detection tasks in CV, but methods don't apply 1:1
- Particle relations can be represented through graphs (vertices and edges)
- Dynamical graphs can be less computationally expensive
- The algorithm needs to be calibrated