Neural Message Passing for Jet Physics

Isaac Henrion, Johann Brehmer, Joan Bruna, Kyunghyun Cho, Kyle Cranmer, Gilles Louppe, Gaspar Rochette

Courant Institute & Center for Data Science

Introduction

Introduction

```python
from physics.knowledge import *
```

Introduction

```
from physics.knowledge import *
```

$+$

```
from ml.algorithms import *
```

Introduction

```
from physics.knowledge import *
```

+

```
from ml.algorithms import *
```

[...]

```
from __future__ import nobel.prize
```

Introduction

```
from physics.knowledge import *
```

+

```
from ml.algorithms import *
```

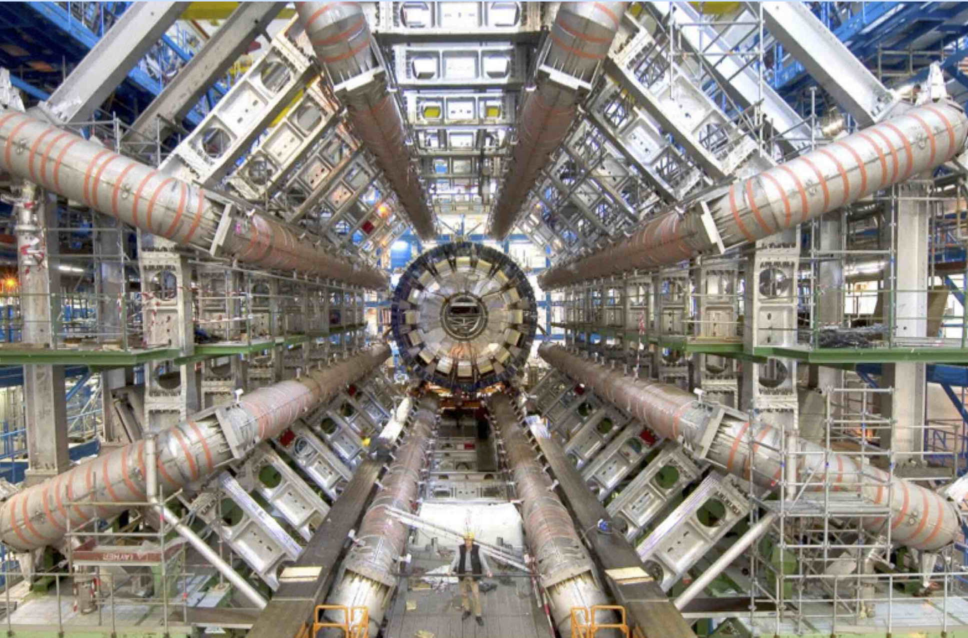
[...]

```
from __future__ import nobel.prize
```
†

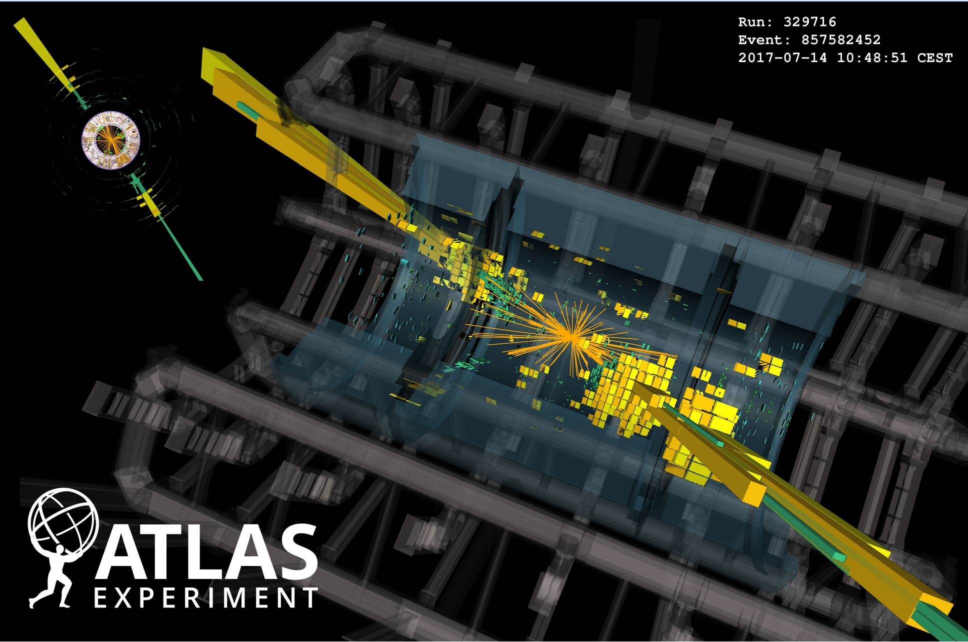
†[K. Cranmer, '17]

Jet physics

Large Hadron Collider

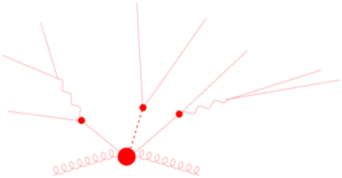


ATLAS


Run: 329716
Event: 857582452
2017-07-14 10:48:51 CEST
ATLAS
EXPERIMENT

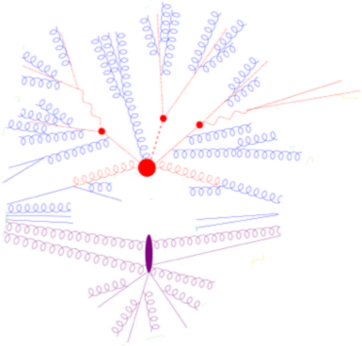
Microscopic picture

Microscopic picture



pencil and paper calculable
from first principles

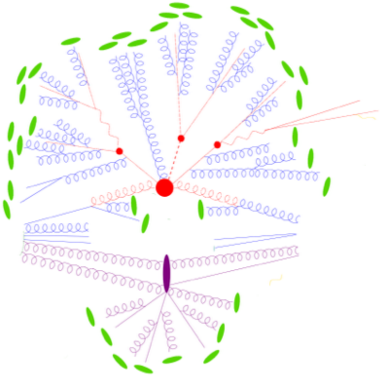
Microscopic picture



pencil and paper calculable
from first principles

controlled approximation of
first principles

Microscopic picture

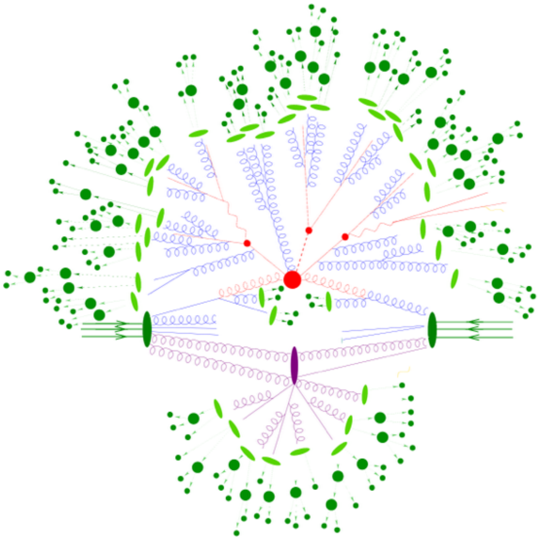


pencil and paper calculable from first principles

controlled approximation of first principles

phenomenological model

Microscopic picture

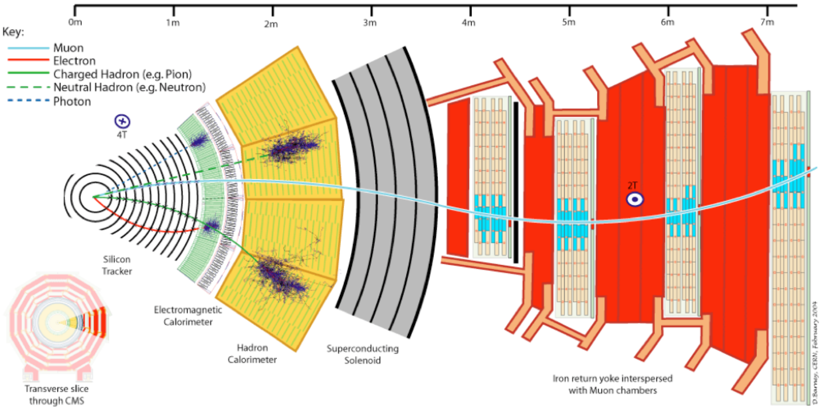


pencil and paper calculable from first principles

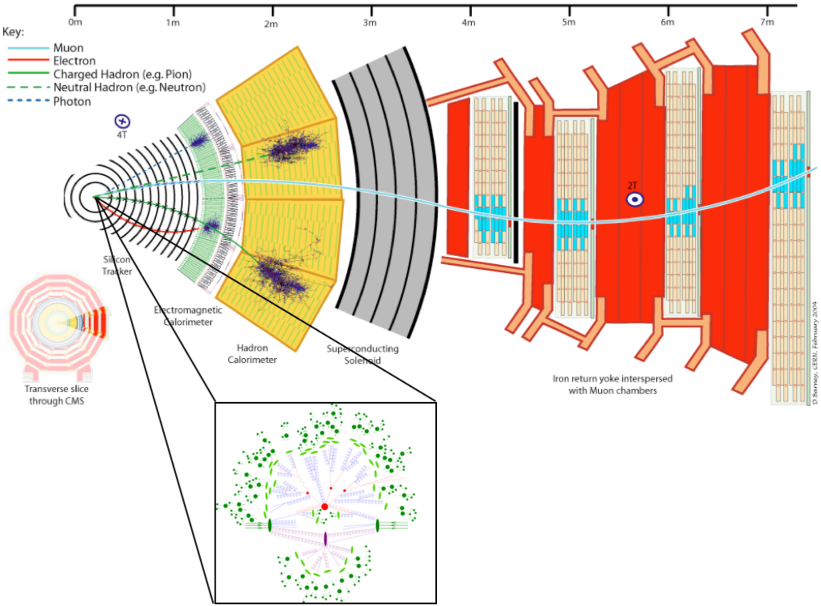
controlled approximation of first principles

phenomenological model

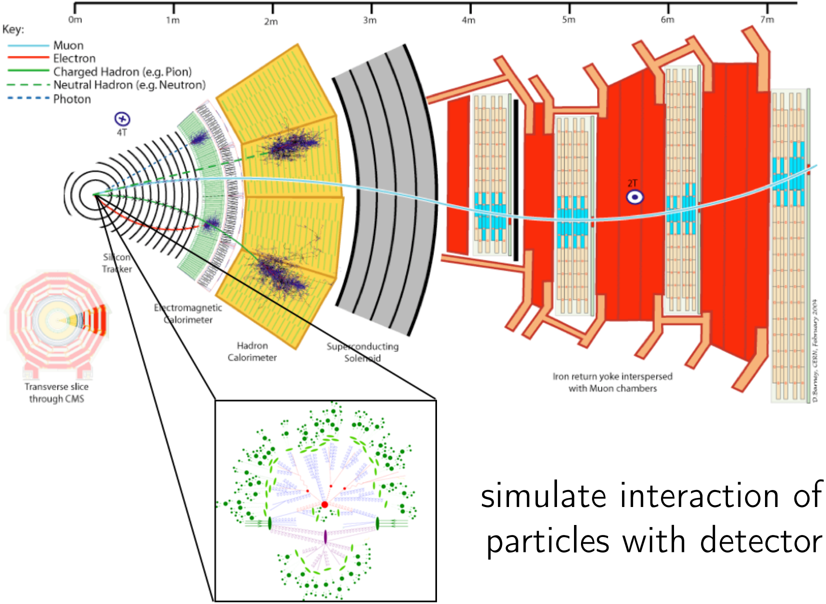
Macroscopic picture



Macroscopic picture



Macroscopic picture



simulate interaction of particles with detector

Classification of W-bosons

Classification of W-bosons

Input

Momentum estimates of jet constituents

$$\{x_1, \ldots, x_n\}$$

Classification of W-bosons

Input

Momentum estimates of jet constituents

Goal

Infer the progenitor particle of the jet.

$$\{x_1, \ldots, x_n\} \rightarrow \begin{cases} W\text{-boson (signal)} \\ \text{QCD (background)} \end{cases}$$

Classification of W-bosons

Input

Momentum estimates of jet constituents

Goal

Infer the progenitor particle of the jet.

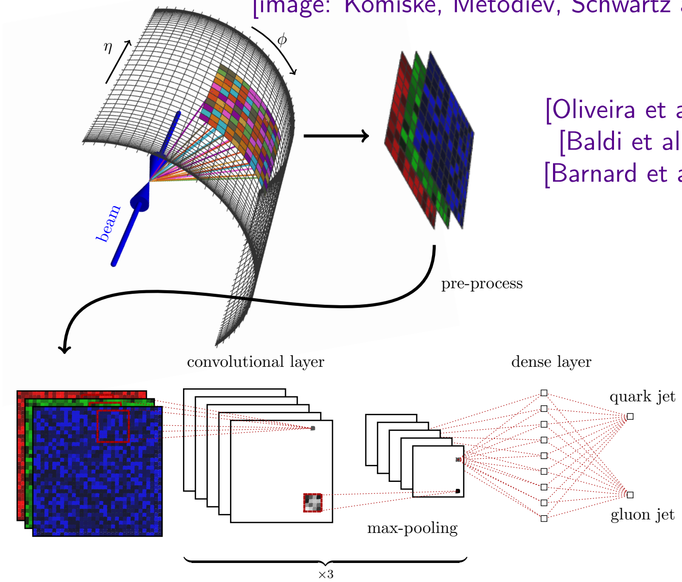
$$\{x_1, \ldots, x_n\} \rightarrow \begin{cases} W\text{-boson (signal)} \\ \text{QCD (background)} \end{cases}$$

Binary classification problem!

Previous work

Jet images

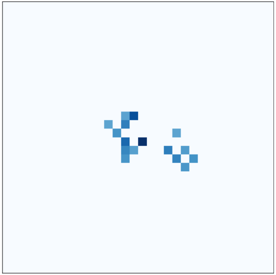
[image: Komiske, Metodiev, Schwartz arxiv:1612.01551]

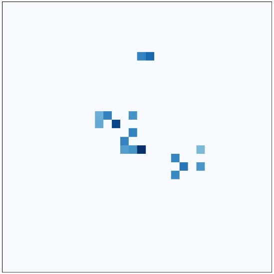


[Oliveira et al arXiv:1511.05190]
[Baldi et al arXiv:1603.09349]
[Barnard et al arXiv:1609.00607]

Jet images

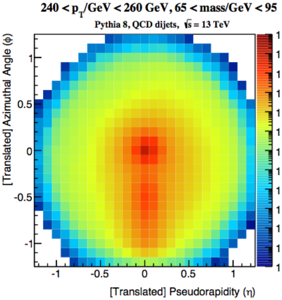
Single QCD jet

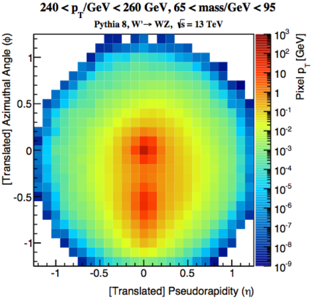




Single W jet

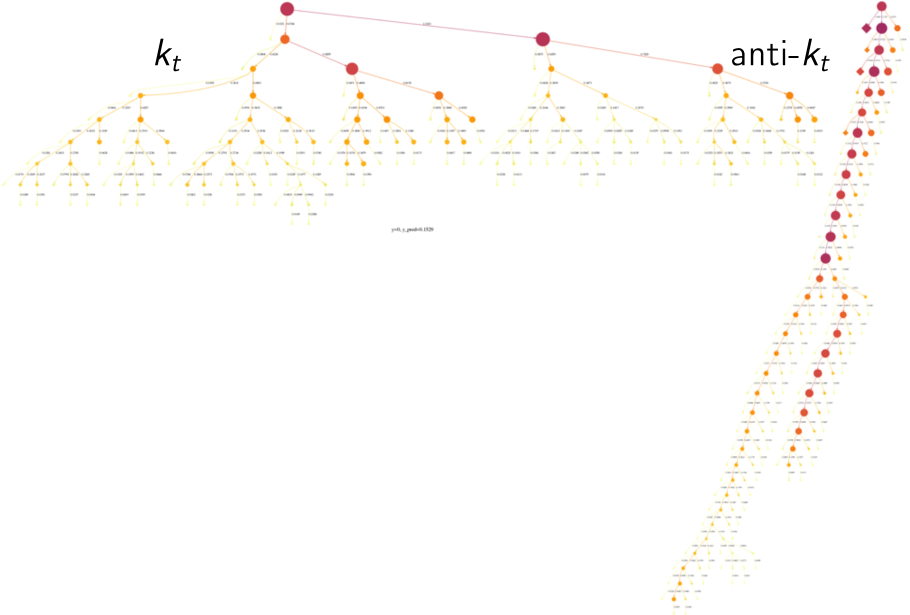
Average QCD jet





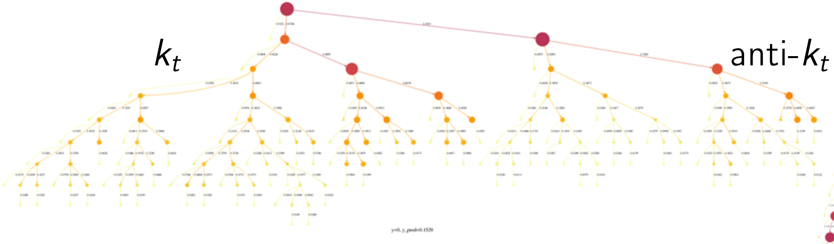
Average W jet

Jet parse trees



Jet parse trees

k_t

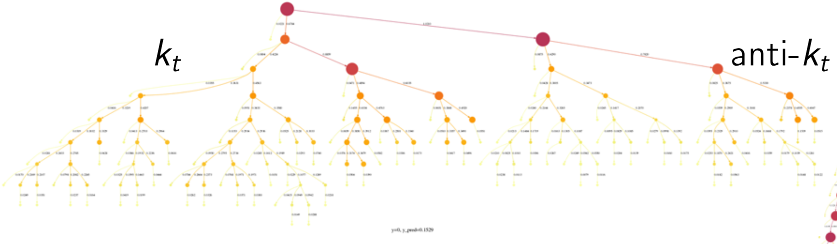


anti-k_t

- Attempt to reverse the generative process

Jet parse trees

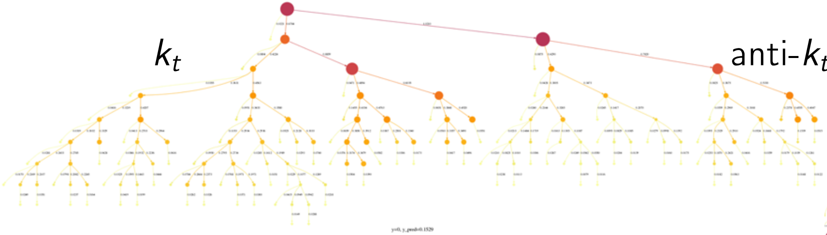
k_t



anti-k_t

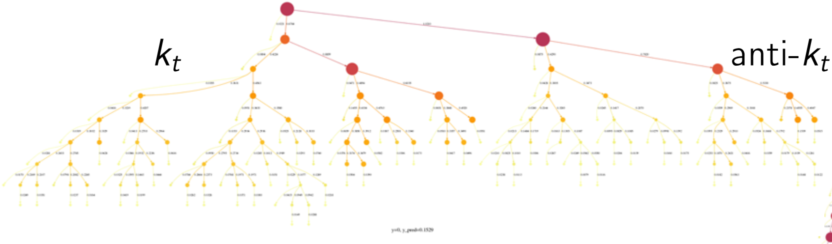
- Attempt to reverse the generative process
- Sequential recombination algorithms

Jet parse trees



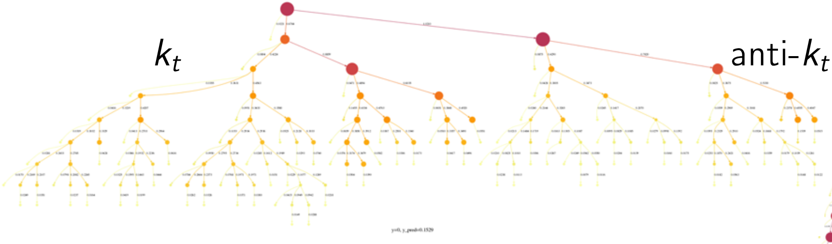
- Attempt to reverse the generative process
- Sequential recombination algorithms
- Cambridge-Aachen, k_t, anti-k_t

Jet parse trees



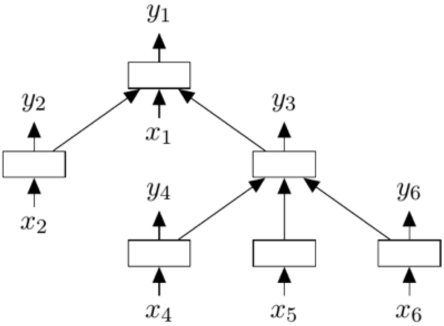
- Attempt to reverse the generative process
- Sequential recombination algorithms
- Cambridge-Aachen, k_t, anti-k_t
- Binary tree representation

Jet parse trees



- Attempt to reverse the generative process
- Sequential recombination algorithms
- Cambridge-Aachen, k_t, anti-k_t
- Binary tree representation
- NLP methods for parse trees

Recursive neural network

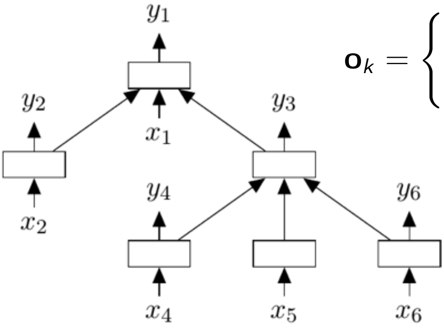


Recursive neural network

$$\mathbf{y}_k = \begin{cases} \mathbf{u}_k & \text{if } k \text{ a leaf} \\ \sigma\left(W_h \begin{bmatrix} \mathbf{y}_{k_L} \\ \mathbf{y}_{k_R} \\ \mathbf{u}_k \end{bmatrix} + \mathbf{b}_h \right) & \text{otherwise} \end{cases}$$

$$\mathbf{u}_k = \sigma\left(W_u g(\mathbf{o}_k) + \mathbf{b}_u\right)$$

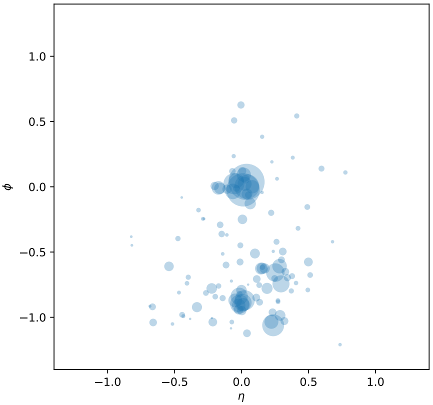
$$\mathbf{o}_k = \begin{cases} \mathbf{x}_{i(k)} & \text{if } k \text{ a leaf} \\ \mathbf{o}_{k_L} + \mathbf{o}_{k_r} & \text{otherwise} \end{cases}$$



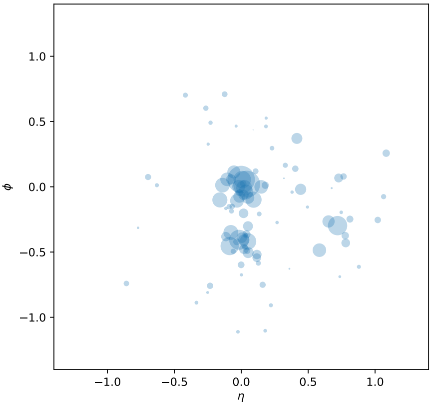
[Louppe et al. 2017]

Our work

Jet graphs

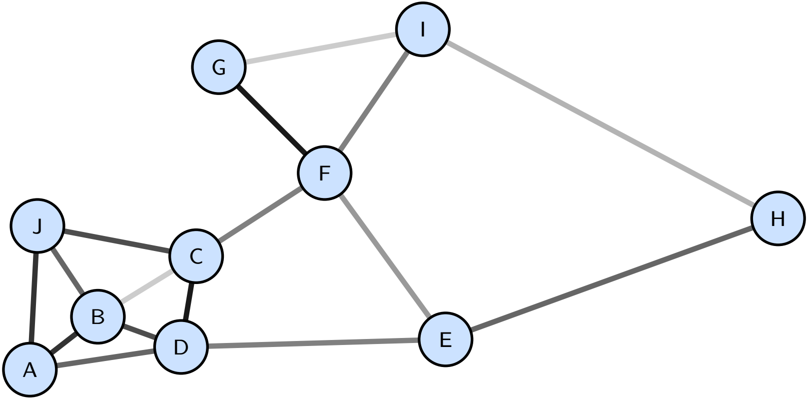


QCD jet

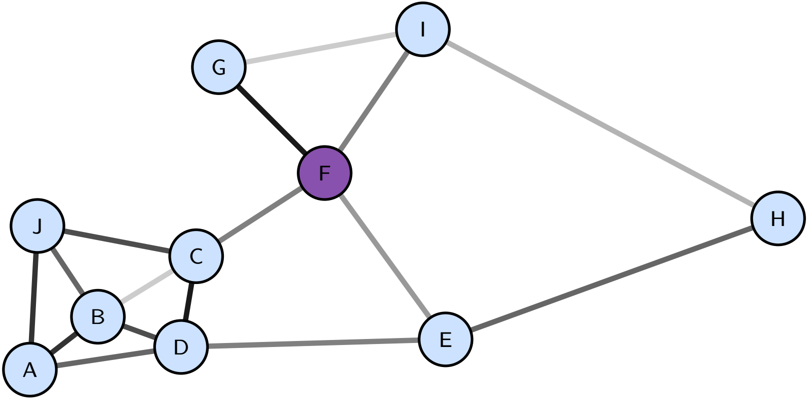


W jet

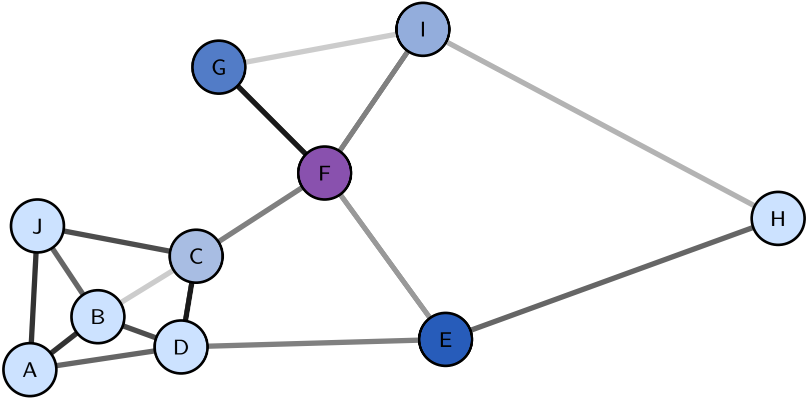
Graph neural networks



Graph neural networks

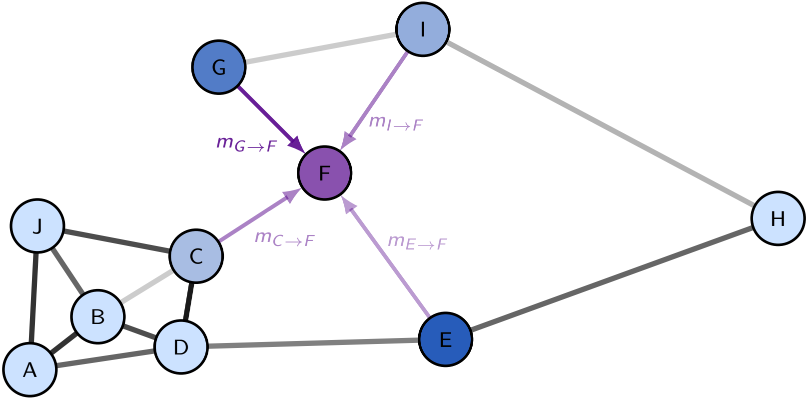


Graph neural networks



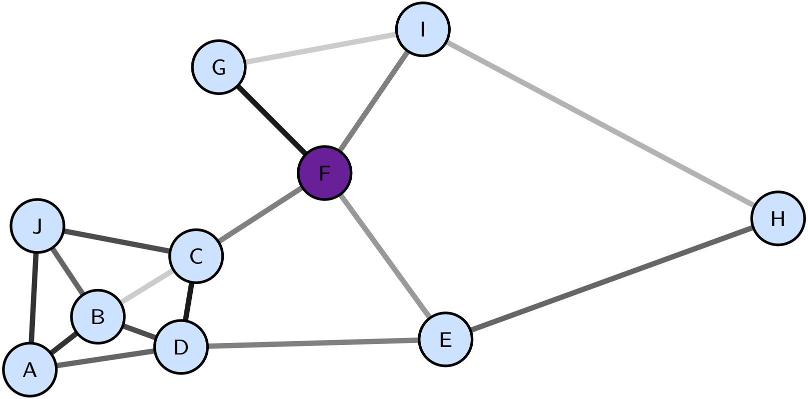
$$\tilde{m}_j^t = f(h_j^{t-1})$$

Graph neural networks



$$\tilde{m}_j^t = f(h_j^{t-1})$$
$$m_{j \to i}^t = \sigma(A_{ij} \tilde{m}_j^t)$$

Graph neural networks



$$\tilde{m}_j^t = f(h_j^{t-1})$$

$$m_{j\to i}^t = \sigma(A_{ij}\tilde{m}_j^t)$$

$$h_i^t = \mathrm{GRU}(h_i^{t-1}, \Sigma_j m_{j\to i}^t)$$

Message Passing Neural Network

Algorithm 1 Message passing neural network

Require: $N \times D$ nodes $\mathbf{x}$, adjacency matrix A

$\mathbf{h} \leftarrow \text{Embed}(\mathbf{x})$
for $t = 1, \ldots, T$ **do**
  $\mathbf{m} \leftarrow \text{Message}(A, \mathbf{h})$
  $\mathbf{h} \leftarrow \text{VertexUpdate}(\mathbf{h}, \mathbf{m})$
end for
$\mathbf{r} = \text{Readout}(\mathbf{h})$
return $\text{Classify}(\mathbf{r})$

A problem with the adjacency matrix

A problem with the adjacency matrix

Question

Where does adjacency matrix come from?

A problem with the adjacency matrix

Question

Where does adjacency matrix come from?

Answer 1

Use a physics-inspired adjacency matrix.

A problem with the adjacency matrix

Question

Where does adjacency matrix come from?

Answer 1

Use a physics-inspired adjacency matrix.

BONUS: import physics knowledge

A problem with the adjacency matrix

Question

Where does adjacency matrix come from?

Answer 1

Use a physics-inspired adjacency matrix.

BONUS: import physics knowledge

Answer 2

Learn the adjacency matrix from the data.

A problem with the adjacency matrix

Question

Where does adjacency matrix come from?

Answer 1

Use a physics-inspired adjacency matrix.

BONUS: import physics knowledge

Answer 2

Learn the adjacency matrix from the data.

BONUS: export physics algorithm

A problem with the adjacency matrix

Question

Where does adjacency matrix come from?

Answer 1

Use a physics-inspired adjacency matrix.
BONUS: import physics knowledge

Answer 2

Learn the adjacency matrix from the data.
BONUS: export physics algorithm

Learning the adjacency matrix

Learning the adjacency matrix

$$s_{ij}^t = F(h_i^{t-1}, h_j^{t-1})$$

Learning the adjacency matrix

$$F(h, h') = v^\top (h + h') + b$$

$$s_{ij}^t = F(h_i^{t-1}, h_j^{t-1})$$

Learning the adjacency matrix

$$F(h, h') = v^\top (h + h') + b$$

$$s_{ij}^t = F(h_i^{t-1}, h_j^{t-1})$$

$$A_{ij}^t = \frac{\exp\{s_{ij}^t\}}{\sum_k \exp\{s_{ik}^t\}} \qquad \text{(directed)}$$

Learning the adjacency matrix

$$F(h, h') = v^\top (h + h') + b$$

$$s_{ij}^t = F(h_i^{t-1}, h_j^{t-1})$$

$$A_{ij}^t = \frac{\exp\{s_{ij}^t\}}{\sum_k \exp\{s_{ik}^t\}}$$

(directed)

$$A_{\text{sym}} = \frac{1}{2}\left(A + A^\top\right)$$

(undirected)

Message Passing Neural Network

Algorithm 2 Message passing neural network

Require: $N \times D$ array of jet constituents $\mathbf{x}$

$\mathbf{h} \leftarrow \text{Embed}(\mathbf{x})$

for $t = 1, \ldots, T$ **do**

  $A \leftarrow \text{AdjacencyMatrix}_t(\mathbf{h})$

  $\mathbf{m} \leftarrow \text{Message}_t(A, \mathbf{h})$

  $\mathbf{h} \leftarrow \text{VertexUpdate}_t(\mathbf{h}, \mathbf{m})$

end for

$\mathbf{r} = \text{Readout}(\mathbf{h})$

return $\text{Classify}(\mathbf{r})$

Experiments

Dataset and metric

Dataset and metric

- Data sampled from the PYTHIA event generator

Dataset and metric

- Data sampled from the PYTHIA event generator
- Same as in [Louppe et al. 2017]

Dataset and metric

- Data sampled from the PYTHIA event generator
- Same as in [Louppe et al. 2017]
- 90k/10k/100k examples (train/validation/test)

Dataset and metric

- Data sampled from the PYTHIA event generator
- Same as in [Louppe et al. 2017]
- 90k/10k/100k examples (train/validation/test)
- Classes are balanced: 50/50 signal/background

Dataset and metric

- Data sampled from the PYTHIA event generator
- Same as in [Louppe et al. 2017]
- 90k/10k/100k examples (train/validation/test)
- Classes are balanced: 50/50 signal/background
- Metric: 1/FPR @ TPR = 50%

Dataset and metric

- Data sampled from the PYTHIA event generator
- Same as in [Louppe et al. 2017]
- 90k/10k/100k examples (train/validation/test)
- Classes are balanced: 50/50 signal/background
- Metric: 1/FPR @ TPR = 50%
- Binary cross-entropy loss

Classification results

Model	Iterations	$R_{\epsilon=50\%}$
Rec-NN (no gating)	1	70.4 ± 3.6
Rec-NN (gating)	1	$\mathbf{83.3 \pm 3.1}$
MPNN (directed)	1	89.4 ± 3.5
MPNN (directed)	2	$\mathbf{98.3 \pm 4.3}$
MPNN (directed)	3	85.9 ± 8.5
MPNN (identity)	3	74.5 ± 5.2
Relation Network	1	67.7 ± 6.8

Classification results

1/FPR @ TPR = 50%

Model	Iterations	$R_{\epsilon=50\%}$
Rec-NN (no gating)	1	70.4 ± 3.6
Rec-NN (gating)	1	$\mathbf{83.3} \pm \mathbf{3.1}$
MPNN (directed)	1	89.4 ± 3.5
MPNN (directed)	2	$\mathbf{98.3} \pm \mathbf{4.3}$
MPNN (directed)	3	85.9 ± 8.5
MPNN (identity)	3	74.5 ± 5.2
Relation Network	1	67.7 ± 6.8

Classification results

1/FPR @ TPR = 50%

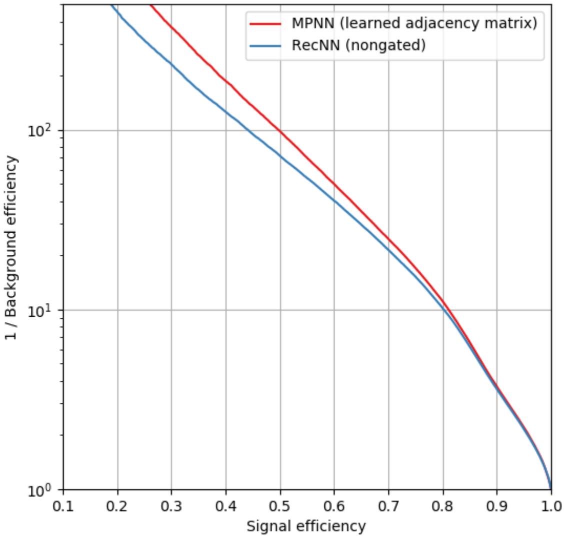
Model	Iterations	$R_{\epsilon=50\%}$
Rec-NN (no gating)	1	70.4 ± 3.6
Rec-NN (gating)	1	83.3 ± 3.1
MPNN (directed)	1	89.4 ± 3.5
MPNN (directed)	2	$\mathbf{98.3 \pm 4.3}$
MPNN (directed)	3	85.9 ± 8.5
MPNN (identity)	3	74.5 ± 5.2
Relation Network	1	67.7 ± 6.8

Classification results

1/FPR @ TPR = 50%

Model	Iterations	$R_{\epsilon=50\%}$
Rec-NN (no gating)	1	70.4 ± 3.6
Rec-NN (gating)	1	$\mathbf{83.3 \pm 3.1}$
MPNN (directed)	1	89.4 ± 3.5
MPNN (directed)	2	$\mathbf{98.3 \pm 4.3}$
MPNN (directed)	3	85.9 ± 8.5
MPNN (identity)	3	74.5 ± 5.2
Relation Network	1	67.7 ± 6.8

Results


MPNN (learned adjacency matrix)
RecNN (nongated)
1 / Background efficiency
Signal efficiency
10^2
10^1
10^0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

Future work

Future work

- MPNN is $O(n^2)$ – can we make the matrix sparse?

Future work

- MPNN is $O(n^2)$ – can we make the matrix sparse?
- Apply MPNN to larger datasets.

Future work

- MPNN is $O(n^2)$ – can we make the matrix sparse?
- Apply MPNN to larger datasets.
- Reduce the number of nodes at each iteration (attention).

Future work

- MPNN is $O(n^2)$ – can we make the matrix sparse?
- Apply MPNN to larger datasets.
- Reduce the number of nodes at each iteration (attention).
- Use QCD-inspired adjacency matrix for message passing.

Future work

- MPNN is $O(n^2)$ – can we make the matrix sparse?
- Apply MPNN to larger datasets.
- Reduce the number of nodes at each iteration (attention).
- Use QCD-inspired adjacency matrix for message passing.
- Export adjacency matrix for sequential recombination jet algorithms.

Thank you!