ABCNet: Jet-tagging and part segmentation

Vinicius M. Mikuni, Florencia Canelli
Introduction

- How do we **best extract information** out of a data set?
- Not a straightforward answer:
  - What kind of information we want?
  - What is the data format?
- Is there a more **natural approach** to treat **collider data**?
- **Point clouds**: set of unordered data points, in a well defined space, representing a system
ABCNet: Easy as 1-2-3

- Enters Attention-Based Cloud Net (ABCNet) [arXiv:2001.05311]
- Uses the flexibility of point clouds with attention mechanisms
- Based on the GAPNet implementation
- How does it work?

**Attention:** Let the method learn the relevant parts for the task at hand (like the bold text I’m using in this presentation)
GAPLayers

- The core component of ABCNet are Graph attention pooling layers GAPLayers
- Start with a particle \( x_i \) (node)
- Take the **k-nearest neighbors** (in \( \eta-\phi \) space or even in the latent space)
- Define the **edge features** \( y_{ij} \)
- Encode the nodes and edges
- Create self- and local-attention coefficients
- Merge
- Align the coefficients using Softmax
- Create 1 coefficient per node

Nodes: \( x_i \)
Edge features: \( y_{ij} = x_i - x_{ij} \)
GAPLayers

- The core component of ABCNet are Graph attention pooling layers GAPLayers
- Start with a particle $x_i$ (node)
- Take the $k$-nearest neighbors (in $\eta$-$\phi$ space or even in the latent space)
- Define the edge features $y_{ij}$
- Encode the nodes and edges
- Create self- and local-attention coefficients
- Merge
- Align the coefficients using Softmax
- Create 1 coefficient per node

 Encode the nodes (and edges) by passing it to a 2 layers NN with output size $F$ and 1

Self-attention: encoded nodes $x'_i$
Local-attention: encoded edge $y'_{ij}$
The core component of ABCNet are Graph attention pooling layers (GAPLayers).

- Start with a particle $x_i$ (node).
- Take the k-nearest neighbors (in $\eta - \phi$ space or even in the latent space).
- Define the edge features $y_{ij}$.
- Encode the nodes and edges.
- Create self- and local-attention coefficients.

- Merge
- Align the coefficients using Softmax.
- Create 1 coefficient per node.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$
GAPLayers

- The core component of ABCNet are Graph attention pooling layers GAPLayers
- Start with a particle $x_i$ (node)
- Take the k-nearest neighbors (in $\eta$-$\phi$ space or even in the latent space)
- Define the edge features $y_{ij}$
- Encode the nodes and edges
- Create self- and local-attention coefficients
- Merge
- Align the coefficients using Softmax
- Create 1 coefficient per node

Each node $x_i$ has receives 1 attention feature written as:

$$a_i = \text{ReLU}(\Sigma_j c'_{ij} y'_{ij})$$

GAPLayer outputs:

Graph features: $y'_{ij}$
Attention features: $a_i$
Applications: Classification

- **Quark-Gluon discrimination**
- Identify if a jet was quark or gluon initiated
- Use the same samples from **Energy Flow Networks**
- **Signal:** $Z(\nu\nu) + (u,d,s)$
- **Background:** $Z(\nu\nu) + g$
- Consider up to 100 particles
- **Input variables** per particle:
  - $\Delta \eta$
  - $\Delta \phi$
  - $\log p_T$
  - $\log E$
  - $\log (p_T/p_T(jet))$
  - $\log (E/E(jet))$
  - $\Delta R$
  - PID

Global features: jet mass and $p_T$
Applications: Classification

- **Same acc. as ParticleNet**
- **15-20% higher background rejection** for the available signal thresholds
- **40% less trainable parameters**
Applications: Classification

- Can we look at what ABCNet is learning?
  - Look at the self-coefficients
  - Only plot the 5% particles inside a jet with the highest self-coefficients
- Particles with the highest importance for the classification are closer to the axis: Colour factor
Applications: Part segmentation

- **Pileup mitigation**
- **Identify if each particle** in the event originates from the primary vertex
- **Use the same samples from PUMML**
- **Signal:** Hypothetical particle $m\phi$ with mass $500$ GeV decaying to $qq$
- **Background:** Soft QCD overlaying each event
- **Consider up to 500 particles**
- **Input variables** per particle:
  - $\eta$
  - $\phi$
  - $\log p_T$
  - Charge
  - $\log (p_T/p_T(jet))$
  - $\log (E/E(jet))$
  - PUPPI weight
  - SoftKiller Flag

**Global features:** number of pileup interactions and particles in the event

---

- $k$ - Hidden layer size
- $H$ - Number of heads
Applications: Part segmentation

- ABCNet returns a **probability for each particle**
- Use the probability to **weight the 4-momentum** for each particle (Like PUPPI does)

\[
<NPU> = 140
\]
Conclusion

- **ABCNet** uses a **graph-based approach** to collider data **enhanced by attention mechanisms**
- **State-of-the-art performance** for the tasks investigated
- The attention allows the **learning** to be **more effective**: Compact architecture
- Easy to **find** the **relevant objects** for a certain task: **Visualization**
- Applicable to other tasks: **flavour tagging, particle tracking, boosted jet identification**
Thanks!

Any questions?
Backup
Multi-head mechanism

- To increase the stability of the network, a GAPLayer can be repeated multiple times.
- The results are concatenated in the end (multi-head).
- Apply an average pooling on all heads to combine the results.
Training details

- Pure **Tensorflow** implementation
- **Adam** optimizer
- **Learning rate:** linear decrease from 1e-3 up to 1e-7
- **Batch size:** 64
- **Loss:** Cross entropy
- **Early stop:** 5 epochs without improvement
- **Quark-Gluon:**
  - Bets epoch: **Largest accuracy**
  - Training-testing events: **1.6M/200k**
- **PU:**
  - Bets epoch: **Minimum loss**
  - Training-testing events: **44k/7k**
How PUPPI works

- PileUp Per Particle Identification (PUPPI)
- It takes as input particle flow objects (charged/neutral hadrons, photon and charged leptons)
  - and it gives weight for each of them.
- Defines $\alpha$ of each particle ($i$)
  using other particles ($j$) around it.
  For example
  $$\alpha_i = \log \left( \sum_{j \text{ event}} \frac{P_T^j}{\Delta R_{ij}} \Theta(R_{\text{min}} < \Delta R_{ij} < R_0) \right)$$
  PT sum weighted with distance
  Step function to take into account only particles around it.

- Transforms the distribution of $\alpha$ in a weight
  (1 for particle from LV, 0 for particles from PU)
- This weight can be also defined at large rapidities where there is no coverage from the tracker.
- Then, jet reconstruction algorithm can run on the particles with the weight (but PUPPI is not only for jet.).
SoftKiller

Eliminates particles with $p_T$ below the minimum threshold that ensures the median of the distribution to be 0.