ML-enabled End-to-End Tracking Reconstruction and Trigger Detection

Giorgian Borca-Tasciuc
Problem Overview

- Raw Pixels
- Pixel Clustering
- Edge Candidates Selection
- Segment Classification (Hit Graph)
- Track Construction
- Track Set
- Trigger Event?
Overview

1. Problem Overview
2. Pixels $\mapsto$ Hits
3. Hits $\mapsto$ Tracks
4. Tracks $\mapsto$ Label
5. Remaining Challenges
6. Conclusion
Pixels $\leftrightarrow$ Hits
Clustering

- Clustering is done by solving a spanning forest problem
- There is an edge between pixels that are adjacent to each other
- Mean of all pixels in a cluster is taken as the hit location
- Most time-consuming portion, we are developing a sparse CNN to perform faster clustering
Hits $\mapsto$ Tracks
Problem Definition

- Once we have hits, we want to group hits that came from the same particle into a track.
- This will be solved by treating the problem as an edge classification problem.
- Out of the $N^2$ possible edges between the hits, we want to know the true edges.
Edge Candidate Selection

- Not all of the $N^2$ possible edges are plausible - we can eliminate a lot of edges from the get-go
- We can use some basic geometric constraints on the cylindrical coordinates of the hits
  - $|\Delta \phi / \Delta r| \leq \text{PHI\_SLOPE\_MAX}$
  - $|z_0| \leq \text{Z\_ORIGIN\_MAX}$
  - $z_0 = z_1 - r \cdot (\Delta z / \Delta r)$
- The geometric constraints determine much of the latency and will play a vital role in further reducing the FPGA latency.
Neural Network Architecture

- Message passing architecture.
- Initialization:
  - $h_v^{(0)} = x_v \quad \forall v \in V$
  - $x_v = (r/3, \varphi, z/3, n_{\text{pixels in hit}}, \text{layer})$
- Message Creation:
  - $m_{u,v}^{(t)} = f_{\text{Message}}(h_u^{(t-1)}, h_v^{(t-1)})$
- Message Aggregation:
  - $a_v^{(t)} = (f_{\text{Agg}}(\{m_{u,v}^{(t)} : v \in N(u)\}), f_{\text{Agg}}(\{m_{v,u}^{(t)} : v \in N^{-1}(u)\}))$
- Node Update:
  - $h_v^{(t)} = f_{\text{Update}}(h_v^{(t-1)}, a_v^{(t)})$
Message Network Details (This slide is very busy)

- \( m_{u,v}^{(t)} = f_{\text{Message}}(h_u^{(t-1)}, h_v^{(t-1)}) \)
- \( a_v^{(t)} = (f_{\text{Agg}}(\{m_{u,v}^{(t)} : v \in N(u)\}), f_{\text{Agg}}(\{m_{v,u}^{(t)} : v \in N^{-1}(u)\})) \)
- \( f_{\text{Edge}} : (\mathbb{R}^f \times \mathbb{R}^f) \mapsto [0, 1] \)
  \( f_{\text{Edge}}(h_u, h_v) = \text{MLP}(h_u, h_v) \)
- \( f_{\text{Message}} : (\mathbb{R}^f \times \mathbb{R}^f) \mapsto \mathbb{R}^{2f} \)
  \( f_{\text{Message}}(h_u, h_v) = f_{\text{Edge}}(h_u, h_v) \cdot h_u \)
- \( f_{\text{Agg}}(M) = \sum_m m \)
Node Network Details

- \( h_v^{(t)} = f_{\text{Update}}(h_v^{(t-1)}, a_v^{(t)}) \)
- \( f_{\text{Update}} : (\mathbb{R}^f \times \mathbb{R}^{2f}) \mapsto \mathbb{R}^f \)
- \( f_{\text{Update}}(h_u, a_u) = \text{MLP}(h_u, a_u) + h_u \)
Track Construction

- Once edge classification is performed, a track is constructed by finding the connected components
- Track is constructed by finding the mean of the hits on each layer
# Performance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Year</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>2023</td>
<td>92.07%</td>
</tr>
<tr>
<td>Precision</td>
<td>2023</td>
<td>92.54%</td>
</tr>
<tr>
<td>Recall</td>
<td>2023</td>
<td>97.97%</td>
</tr>
<tr>
<td>F1</td>
<td>2023</td>
<td>95.18%</td>
</tr>
<tr>
<td>Accuracy</td>
<td>2022</td>
<td>96.30%</td>
</tr>
<tr>
<td>Precision</td>
<td>2022</td>
<td>84.55%</td>
</tr>
<tr>
<td>Recall</td>
<td>2022</td>
<td>83.25%</td>
</tr>
<tr>
<td>F1</td>
<td>2022</td>
<td>83.89%</td>
</tr>
</tbody>
</table>
Tracks $\mapsto$ Label
Problem Definition

- After creating the tracks, we have a set of tracks
- We want to know whether the event that created these tracks was a trigger event
- A trigger event is an event in which we had a $D_0 \rightarrow (\pi^+, K^-)$ or $D_0 \rightarrow (\pi^-, K^+)$ decay
What needs to be modeled?

- $D_0 \mapsto (\pi^+, K^-)$ or $D_0 \mapsto (\pi^-, K^+)$
- Considering the problem from a high level perspective, we need to consider:
  - Track-to-track Interactions: Do these pair of tracks form a $(\pi^+, K^-)$ or $(\pi^+, K^-)$ pair?
  - Track-to-global Interactions: Where is the origin of this track?
  - Global-to-Track Interactions: Incorporate information about the origin of this track into the track embeddings
Architecture

- Previous considerations motivate the following block.
  - Set Encoder: Track-to-Track interactions
  - Bipartite Aggregation: Track-to-Global and Global-to-Track interactions
Set Encoder

- Create Query, Key, Value embeddings from track embeddings using an MLP
- Find attention between every track $i$ and track $j$ by calculating $Q_i \cdot K_j$ and using the softmax to normalize the sum of attention scores to 1
- Weigh value embeddings by the attention score and aggregate to create new track embeddings

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

$$SAB(X) = \text{LayerNorm}(H + rFF(H)), \text{where } H = \text{LayerNorm}(X + \text{Multihead}(X, X, X))$$
Bipartite Aggregators

- Use an MLP: $\mathbb{R}^f \rightarrow \mathbb{R}^n$, followed by a softmax to determine how much each track contributes to each aggregator
  - $f$ is size of track embedding
  - $n$ is number of aggregators
- For each aggregator:
  - Scale each track by its contribution score to that aggregator
  - Perform max and mean pooling over scaled tracks to calculate aggregator embedding
- Concatenate aggregators to track embeddings, and use an MLP to update track embeddings
Architecture

- Stack multiple SEBA Blocks
- Use Bipartite Aggregation with single aggregator to generate event embedding
- MLP on event embedding to predict Trigger Event
Track Features

- Track given to trigger classifier has the following features:
  - (x, y, z) location of hit on each layer
  - Length segment between each layer
  - Angle formed by segments
  - Estimated radius of circle fit to hits
  - Estimated center of circle fit to hits
  - Estimated transverse momentum of track

- Estimated radius and center provided ~10pp increase in performance
Multi-Task Learning to Improve model performance

- Several modifications to standard training process in order to improve the performance and robustness of our trigger algorithm
  - Track embeddings used predict whether two tracks come from the same parent
  - We perturb hits off the detector layers while keeping it on the particle path
- \[ \mathcal{L} = L_{CE}(\text{trigger}_{\text{pred}}, \text{trigger}_{\text{true}}) + L_{CE}(A_{\text{pred}}, A_{\text{true}}) \]
## Performance

<table>
<thead>
<tr>
<th>Data</th>
<th>Year</th>
<th>Metric</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT Tracks</td>
<td>2023</td>
<td>Accuracy</td>
<td>90.22%</td>
</tr>
<tr>
<td>GT Tracks</td>
<td>2023</td>
<td>Precision</td>
<td>86.35%</td>
</tr>
<tr>
<td>GT Tracks</td>
<td>2023</td>
<td>Recall</td>
<td>95.41%</td>
</tr>
<tr>
<td>Predicted Tracks</td>
<td>2022</td>
<td>Accuracy</td>
<td>84.01%</td>
</tr>
<tr>
<td>GT Tracks</td>
<td>2022</td>
<td>Accuracy</td>
<td>87.5%</td>
</tr>
</tbody>
</table>
Remaining Challenges

- Modifying algorithms to deal with pile-up
- Work on simplifying algorithms and reducing data quantity to meet latency challenges
  - Initial study of latency-accuracy tradeoff showed we could reduce edge quantity at the tracking stage by 60% with minimal loss in final trigger accuracy
- Ensure trigger algorithm works in explainable and robust way
  - Initial study has shown model prefers to drop non-trigger tracks without affecting event label and prefers to perturb hits as to not affect the track radius

<table>
<thead>
<tr>
<th>$d\phi_{\text{max}}$</th>
<th>$dz_{\text{max}}$</th>
<th>accuracy</th>
<th>Maximum Edge Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.025005</td>
<td>102.000000</td>
<td>0.885895</td>
</tr>
<tr>
<td>1</td>
<td>0.014881</td>
<td>15.000000</td>
<td>0.885360</td>
</tr>
<tr>
<td>2</td>
<td>0.011599</td>
<td>155.000000</td>
<td>0.884555</td>
</tr>
<tr>
<td>3</td>
<td>0.026555</td>
<td>113.000000</td>
<td>0.884320</td>
</tr>
<tr>
<td>4</td>
<td>0.024582</td>
<td>173.000000</td>
<td>0.883860</td>
</tr>
<tr>
<td>5</td>
<td>0.010320</td>
<td>48.000000</td>
<td>0.882630</td>
</tr>
<tr>
<td>6</td>
<td>0.012193</td>
<td>14.220353</td>
<td>0.881850</td>
</tr>
<tr>
<td>7</td>
<td>0.030000</td>
<td>200.000000</td>
<td>NaN</td>
</tr>
</tbody>
</table>
Conclusion

- ML models have shown steady increases in performance on the triggering problem
- Incorporating physics knowledge been responsible for large gains in performance in trigger prediction
- Challenges remain in adapting the ML algorithm to the real-world latency and data availability constraints