Accelerating End-to-End Deep Learning using Graph Neural Networks

Shravan Chaudhari

Mentors: Prof. Sergei Gleyzer & Dr. Davide DiCroce
Goals of the project

- Use low-level detector features for high energy particle reconstruction (Taus).
- Design Graph based deep learning approaches (Graph Neural Networks)
- Compare their performance with Convolutional Neural Networks.
- Integrate Tau Tagger with the E2E Framework built within CMSSW (with deep learning inference support using Tensorflow C++ Runtime)
- Develop a prototype for integration of Graph Neural Networks with E2E Framework.
Shower plots

DYTauTau pT

TTbar pT

DYTauTau dz

TTbar dz

DYTauTau ECAL

TTbar ECAL

DYTauTau HCAL

TTbar HCAL
Data preparation & preprocessing

- All the nonzero pixels of the images are used to construct the graph.
- The edges are constructed using the euclidean distance and knn graph clustering fashion based on the image coordinates.
- The node features are the nonzero pixels of the image channels.

Multichannel Image

Graph (k=3)
Graph Neural Network

Trained on Tesla P100 GPU and Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz

<table>
<thead>
<tr>
<th>Channels</th>
<th>Validation ROC AUC scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPIX * 4</td>
<td>0.697</td>
</tr>
<tr>
<td>pT + ECAL</td>
<td>0.856</td>
</tr>
<tr>
<td>pT + ECAL + HCAL</td>
<td>0.864</td>
</tr>
<tr>
<td>pT + ECAL + HCAL + dz + d0</td>
<td>0.870</td>
</tr>
<tr>
<td>pT + ECAL + HCAL + dz + d0 + BPIX * 4 + TIB<em>2 + TOB</em>2</td>
<td><strong>0.887</strong></td>
</tr>
</tbody>
</table>
Model Performance for low momentum jets (DYTauTau)

- **DYTauTau vs Wjets**
  
  \[(p_T + \text{ECAL} + \text{HCAL} + dz + d0)\]

- **DYTauTau vs QCD**
  
  \[(p_T + \text{ECAL} + \text{HCAL} + dz + d0)\]

- **DYTauTau vs TTbar**
  
  \[(p_T + \text{ECAL} + \text{HCAL} + dz + d0)\]
Training Data Distribution

Data Samples vs. Data Types

- HTauTau
- QCD
- DYTauTau
- Wjets
- TTbar
Comparison with CNN

<table>
<thead>
<tr>
<th>Channels</th>
<th>GNN</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pT + ECAL + HCAL + dz + d0)</td>
<td>0.87</td>
<td>0.798</td>
</tr>
<tr>
<td>(pT + ECAL + HCAL + dz + d0 + BPIX^4 + TIB^2 + TOB^2)</td>
<td>0.887</td>
<td>0.864</td>
</tr>
</tbody>
</table>
Integration of Tau Reconstruction with E2E Pipeline
Event Pipeline:

Pool source

- Detector arrays (1D)
- Deposit vectors (3D Frames)

- 3D Frames, Seeds
- Object Level Data Producers

- Cropped/Processed Frames (4D)
- Cropped Frames

- Predictions
- TauTagger

FrameProducer package:
- EGFrameProducer: Electrons & Photons
- JetFrameProducer: Quarks, Gluons, Top Jets & Taus

Taggers:
- EGTagger: Electrons & Photons
- QGTagger: Quarks & Gluons
- TopTagger: Top Jets
- TauTagger: Taus
GNN with CMSSW

Available tools:

- MXNet
- SonicTriton (using Nvidia Triton Server)
- ONNX
  - Open format built to represent Machine Learning (ML) models.
  - Consists of common set of operators and common file format to enable the use of ML/DL models with a variety of frameworks, tools, runtimes and compilers.
E2E GNN Framework

- ONNX exporter does not support PyG knn_graph function for graph definition.

<table>
<thead>
<tr>
<th>Batch1</th>
<th>Batch2</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>o</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>z</td>
<td>e</td>
</tr>
<tr>
<td>e</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>o</td>
</tr>
<tr>
<td>pix</td>
<td>pix</td>
</tr>
<tr>
<td>vals</td>
<td>vals</td>
</tr>
</tbody>
</table>
Issues with ONNX exporter & Solutions

- The PyTorch to ONNX exporter is currently not mature enough to convert graph neural networks defined in PyTorch to ONNX.
- Hence, the exporter fails when multiple inputs with different sizes are used in the graph neural network model.
- The core problem seems to be with the message passing class of the torch_geometric (PyG) library due to which the exporter is not able to parse multiple sized inputs correctly.
- To develop the prototype, we wrote the GNN code from scratch without using torch_geometric library. Using this strategy we were able to deploy small sized GNN layers. (Work in progress)
More on issue with PyG

Relevant Github Issues raised: https://github.com/pytorch/pytorch/issues/65138
https://github.com/pytorch/pytorch/issues/64769
Future Goals

- Continue the development of ONNX interface
- Implement several other architectures (attention based) to achieve even better than state of the art performance
THANK YOU