IRIS-HEP Summer Fellowship 2021

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

Shravan Chaudhari

Mentors: Prof. Sergei Gleyzer &
Dr. Davide DiCroce

CMS E2E

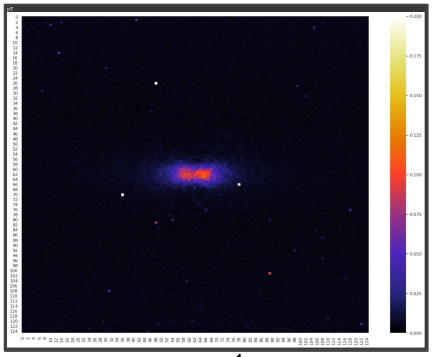
iris hep

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.

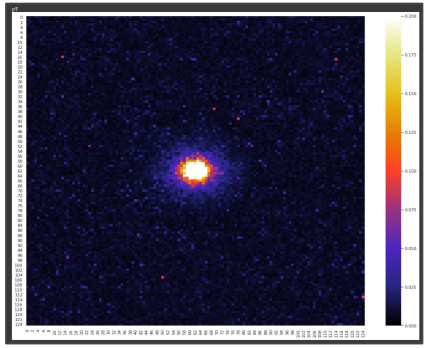
Shower plots

DYTauTau pT



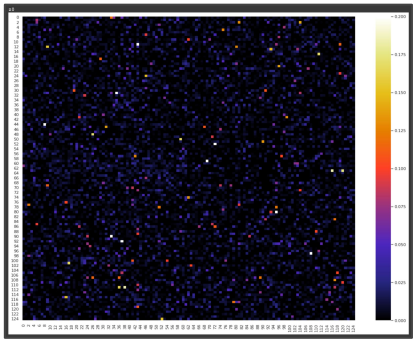
y=1

TTbar pT



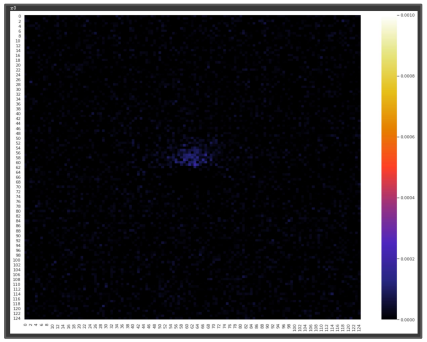
y=0

DYTauTau dz

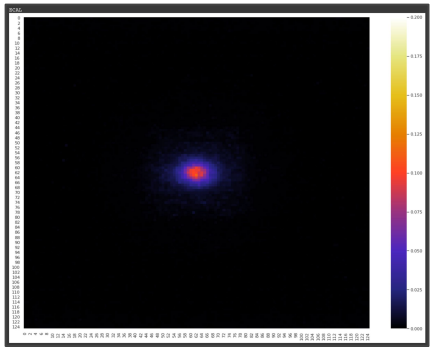


y=1

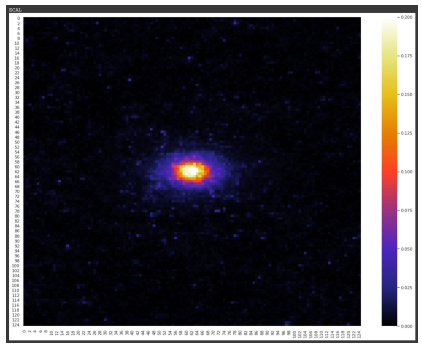
TTbar dz



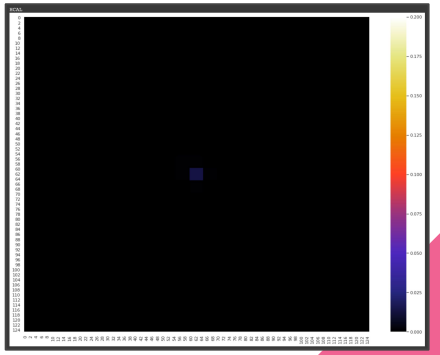
y=0



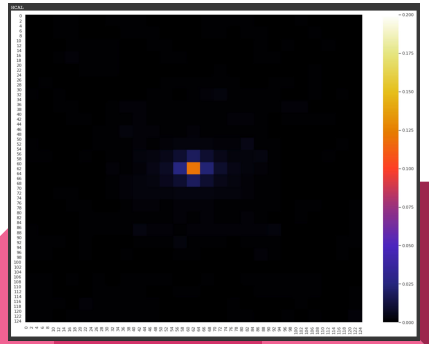
DYTauTau ECAL



TTbar ECAL

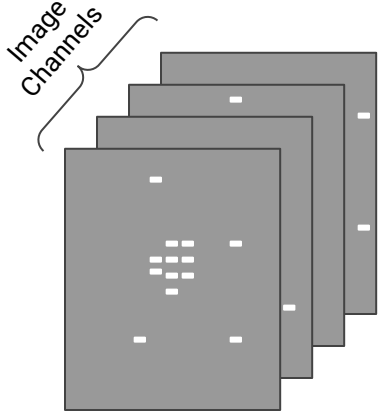


DYTauTau HCAL

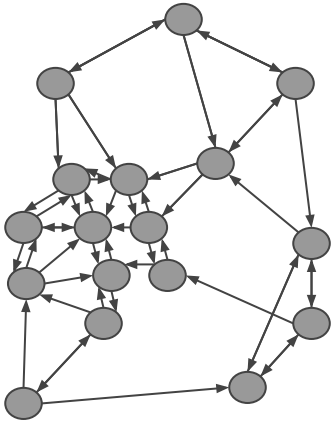


TTbar HCAL

Data preparation & preprocessing



Multichannel Image



Graph (k=3)

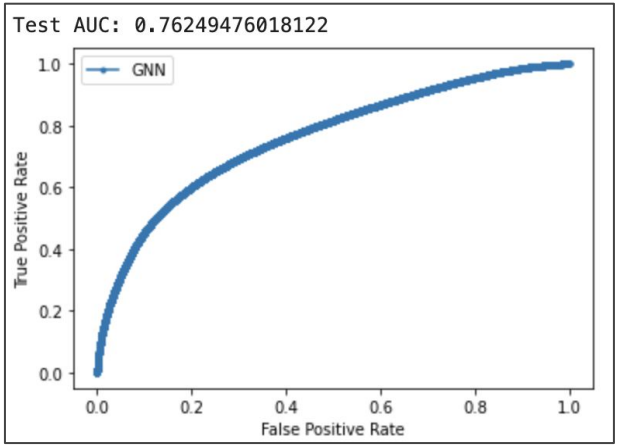
- All the nonzero pixels of the images are used to construct the graph.
- The edges are constructed using the euclidean distance and knn graph clustering fashion based on the image coordinates.
- The node features are the nonzero pixels of the image channels.

Graph Neural Network

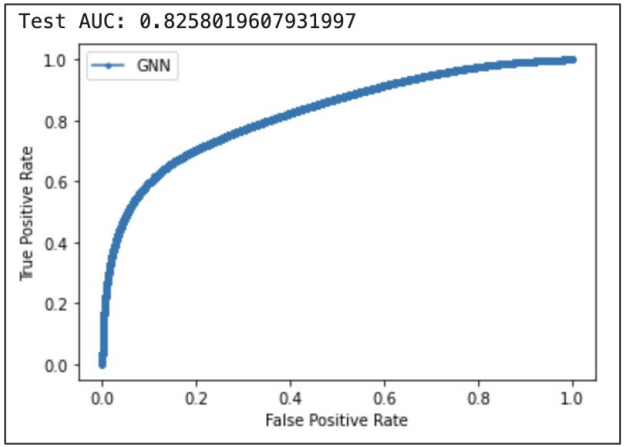
Trained on Tesla P100 GPU and Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz

Channels	Validation ROC AUC scores
BPIX * 4	0.697
pT + ECAL	0.856
pT + ECAL + HCAL	0.864
pT + ECAL + HCAL + dz + d0	0.870
pT + ECAL + HCAL + dz + d0 + BPIX *4 + TIB*2 + TOB*2	**0.887**

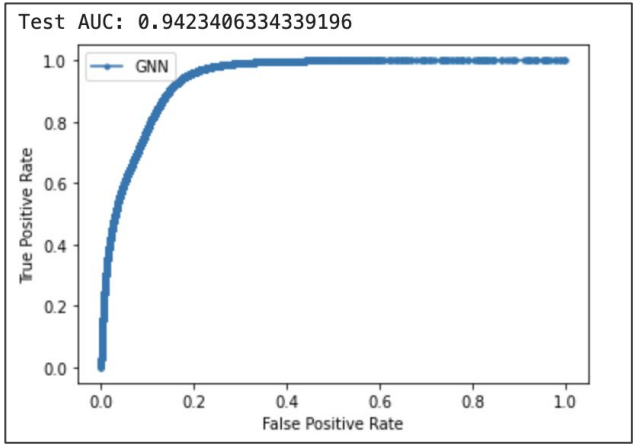
Model Performance for low momentum jets (DYTauTau)



DYTauTau vs Wjets
(pT + ECAL + HCAL + dz + d0)

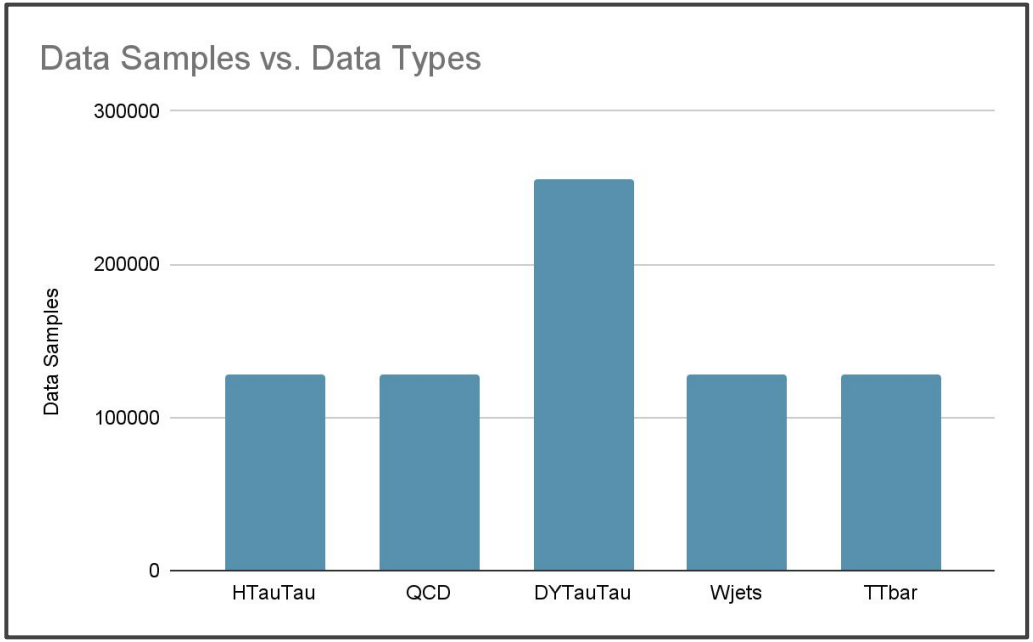


DYTauTau vs QCD
(pT + ECAL + HCAL + dz + d0)



DYTauTau vs TTbar
(pT + ECAL + HCAL + dz + d0)

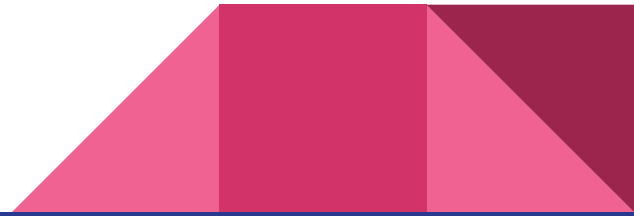
Training Data Distribution



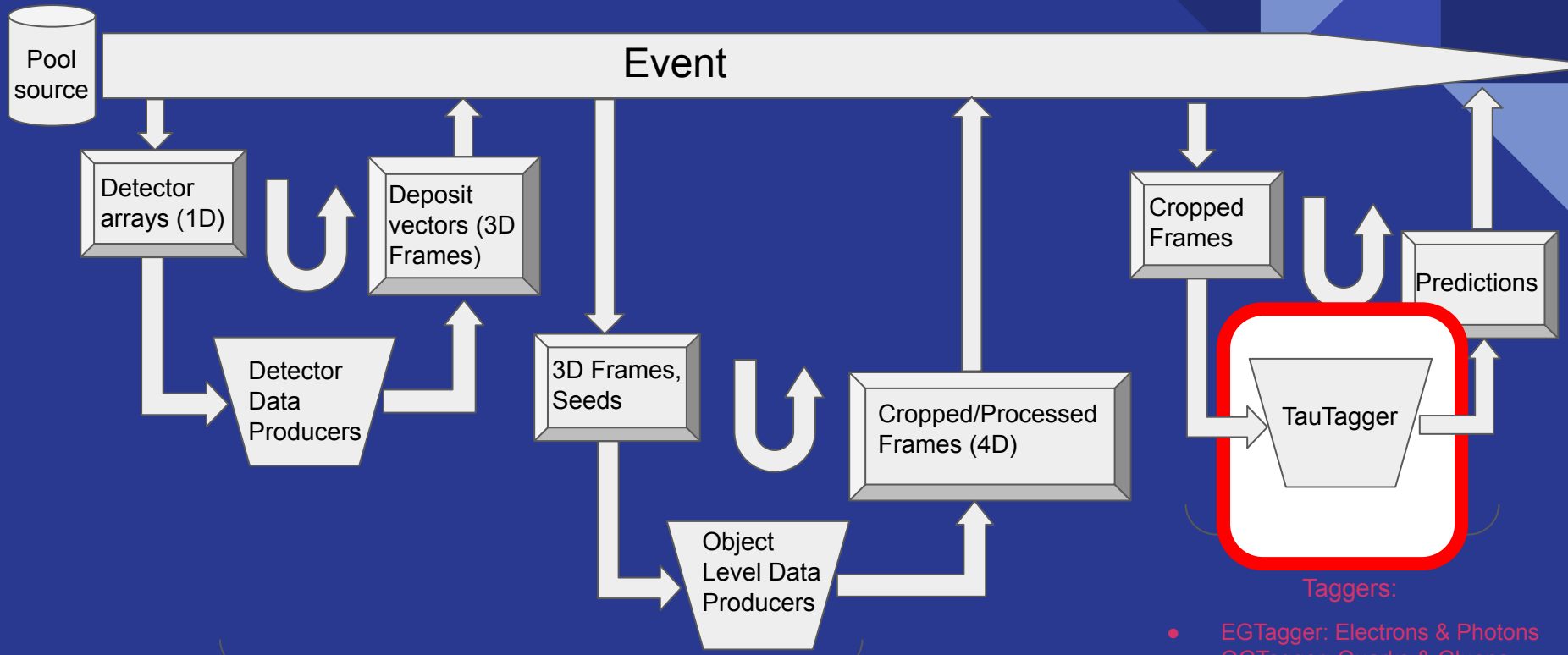
Comparison with CNN

Channels	GNN	CNN
pT + ECAL + HCAL + dz + d0	0.87	0.798
pT + ECAL + HCAL + dz + d0 + BPIX*4 + TIB*2 + TOB*2	0.887	0.864

Integration of Tau Reconstruction with E2E Pipeline



Current Pipeline:



FrameProducer package:

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

Taggers:

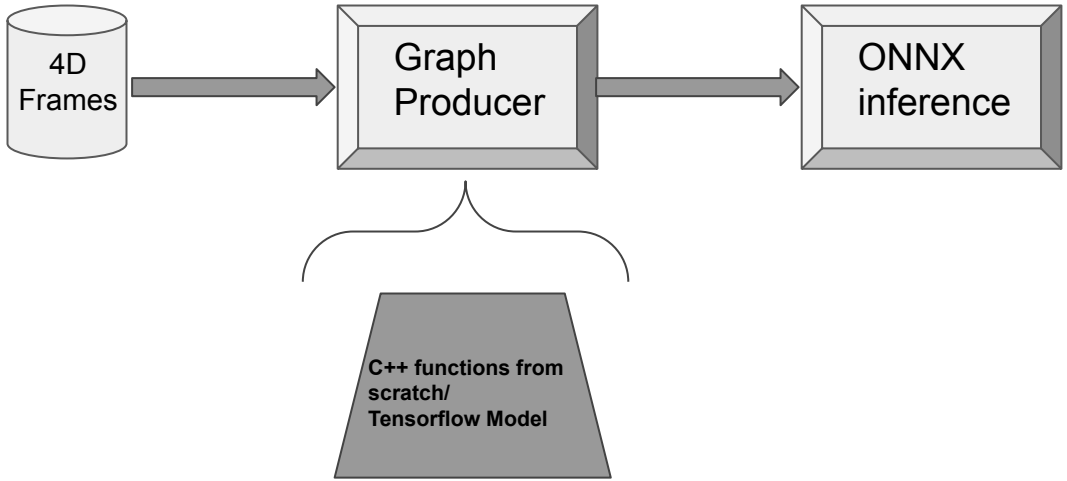
- EGTagger: Electrons & Photons
- QGTagger: Quarks & Gluons
- TopTagger: Top Jets
- TauTagger: Taus

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 Runtime Input Space

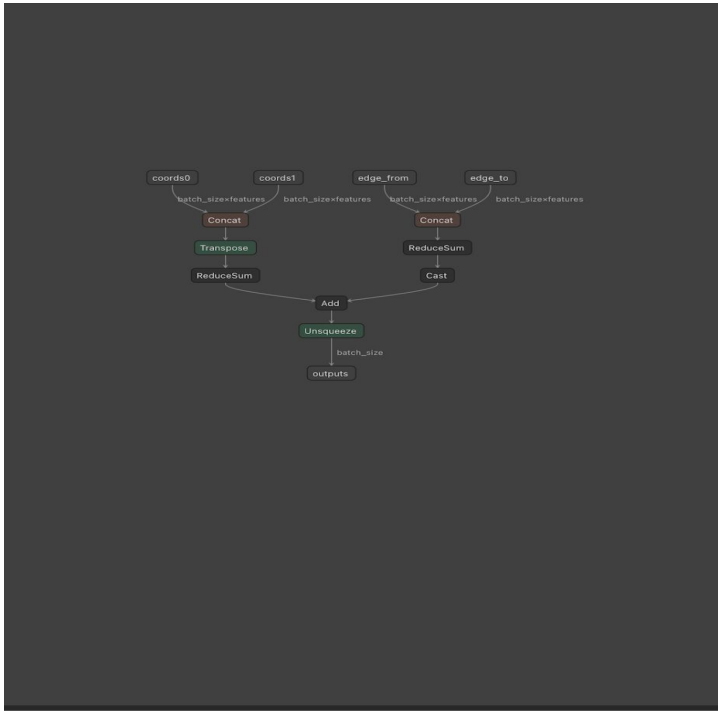
	eta	phi	pT	ecal	hcal	pix
Batch1	N					
	o					
	n					
	z					
	e					
Batch2	r					
	o					
	pix					
	vals					

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

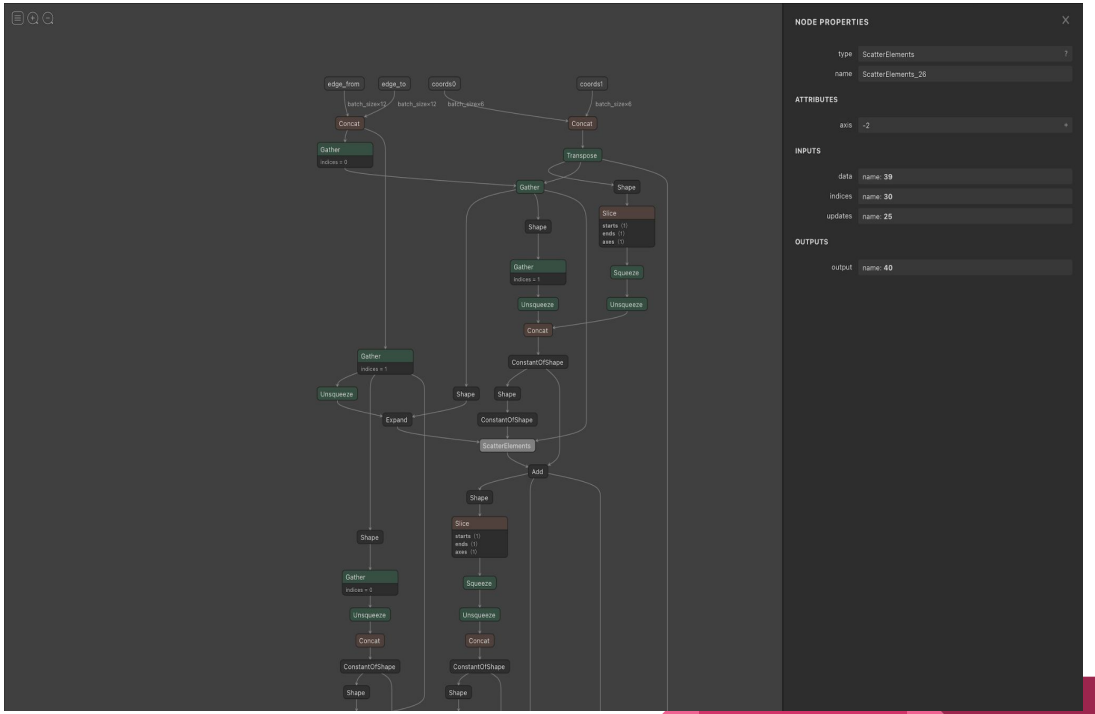
Issues with ONNX exporter & Solutions

- 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



PyTorch Model Without PyG

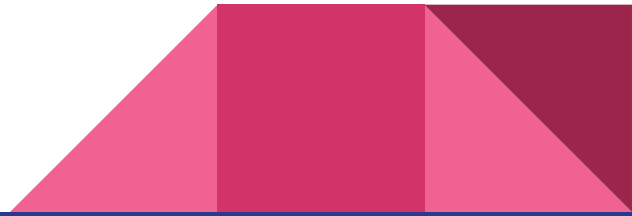


PyTorch Model With PyG

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