Graph Neural Networks for End-to-End Particle Identification with the CMS Experiment

Zixing Song

Personal Information

- Name: Zixing Song
- Email: songzixing98@gmail.com
- Location: Hong Kong SAR
- Time Zone: UTC+8
- University: The Chinese University of Hong Kong
- Major: Computer Science (Ph.D Student)

About Me

I am a PhD student from The Chinese University of Hong Kong and my current research interests include Graph Neural Networks(GNNs) and AI4Science.

I possess extensive hands-on research experience in utilizing Graph Neural Networks (GNNs) for a range of graph-related applications, like in the area of Neutrino Events Classification and Reconstruction in High Energy Physics (HEP). My research endeavors have resulted in several first-authored top-tier conference papers, including KDD, AAAI, NeurIPS, amongst others, that focus on GNNs.

Furthermore, I have had the privilege of interning at several distinguished technical companies, such as Microsoft and Alibaba, which have contributed significantly to honing my coding skills. My programming prowess is primarily centered around Python for research coding, with occasional use of C++. For deep learning, my preferred frameworks are PyTorch and PyG (PyTorch Geometric), which I use extensively on a daily basis.

Project Descriptions

This project will focus on the development of end-to-end graph neural networks for low-momentum tau identification and the testing and benchmarking of inference on GPUs. In recent years, the convolutional neural network has been

successfully applied in particle physics identification tasks [1]. Despite its great success, the existing image-based methods for particle suffers from data sparsity. Therefore, with the trend of Graph Neural Networks(GNNs) [2] in particle physics, we plan to represent the particles data or point cloud data as a graph to model the underlying iterations among particles so that the performance of the downstream task could be potentially improved.

The project will involve the development of advanced machine learning models for particle identification using graph neural networks (GNNs). Specifically, the project will focus on low-momentum tau identification, which is a challenging task that requires high accuracy and efficiency. The project will include two main tasks. The first task will be the development of end-to-end GNN models that can handle the entire tau identification process, from raw detector data to final identification decisions. The models will be trained on large datasets of simulated LHC data and will be optimized for accuracy and efficiency for low-momentum taus. The second task will be to test and benchmark the GNN inference on GPUs. GPUs have become a popular tool for accelerating machine learning models, and the use of GPUs can greatly improve the speed of inference. The benchmarking will be performed on a variety of GPUs to ensure that the models can be run efficiently on a wide range of hardware.

The expected results of the project will include a trained end-to-end graph neural network model for low-momentum tau identification that is optimized for accuracy and efficiency, as well as a benchmark of the end-to-end GNN inference on GPUs. The models and benchmarking results will be made publicly available to the scientific community, and will be integrated into the CMS experiment's reconstruction algorithms for use in offline and high-level trigger systems.

Technical Details

Dataset

Based on the Task 3 in the test and released codes from previous years of GSoC [3], we find that the Boosted Top Tau Particle dataset [4] contains `X-jets`, `y`, `m0`, and `pt` columns, where `y` represents the label for tau sample and the `X-jets` column is 125×125 jets image matrices of 8 channels (pt, d0, dz ECAL, HCAL, BPIX1, BPIX2, and BPIX3). More specifically, we have

- (pt, d0, dz) are tracker layers.
- ECAL and HCAL are electromagnetic and hadronic calorimeter deposits respectively.
- BPIX are the pixel layers which are integer values representing the number of collisions occurring in the pixel layers.

Therefore, the first step is represent the jet image as a graph so that the GNN model can be applied on top of this constructed graph for the downstream learning task.

Graph Construction

Constructing a graph from a jet image in the CMS experiment involves representing the complex relationships and interactions between particles in a jet using graph-structured data. Here is a step-by-step process to achieve this.

1. Preprocessing data: Begin by preprocessing the CMS experiment data, which includes information about particle trajectories, energy deposits, and other relevant features. Convert the raw data into jet images, where each image represents a single jet event. A jet is a collimated spray of particles originating from high-energy processes, such as quark or gluon interactions.

2. Define nodes: In the graph representation, nodes will represent the constituents of the jet, such as charged and neutral particles, as well as energy deposits in the calorimeters. Extract relevant features for each node, such as energy, momentum, position, and particle type.

3. Define edges: Edges in the graph will represent the relationships and interactions between the nodes (jet constituents). You can create edges based on several criteria, such as:

    - Spatial proximity: Connect nodes that are close to each other in the detector's coordinate system.
    - Topological features: Connect nodes based on their topological relationships, such as shared sub-detector elements or overlapping energy deposits.
    - K-nearest neighbors: For each node, connect it to the K nearest nodes based on a chosen distance metric (e.g., Euclidean distance).

4. Assign edge weights: Edge weights can help capture the relative strength or importance of the interactions between nodes. Weights can be assigned based on various criteria, such as:

    - Inverse distance: Assign higher weights to edges connecting nodes that are closer together.
    - Shared energy: Assign higher weights to edges connecting nodes with a higher degree of shared energy deposits.
    - Other domain-specific knowledge: Assign weights based on expert knowledge or specific particle interaction properties.

5. Create adjacency matrix A: Use the above-defined nodes, edges, and edge weights to create an adjacency matrix representing the graph. The adjacency matrix is a square matrix where the rows and columns are indexed by the nodes, and the (i, j) entry represents the weight of the edge connecting node i to node j.

6. Create node feature matrix X: Organize the node features into a feature matrix, where each row corresponds to a node and each column corresponds to a specific feature.

With the adjacency matrix A and the node feature matrix X, we can now use graph neural networks (GNNs) to analyze and process the graph representation of the jet images. GNNs can effectively capture the complex relationships between jet constituents, leading to improved performance in tasks such as particle identification.

Choices of GNN Architecture

There are several types of Graph Neural Network (GNN) models that can be considered for the task of tau identification in particle physics experiments. These models have different strengths and capabilities, and their suitability depends on the specific problem requirements and data characteristics. We plan to employ the following potential choices of GNN models.

- Graph Convolutional Networks (GCNs): GCNs [5] are one of the most widely-used GNN models, and they perform well on tasks that involve learning from local neighborhood information. They use a convolution operation to aggregate information from neighboring nodes and learn meaningful node representations.
- GraphSAGE: GraphSAGE [6] is designed to handle large graphs by sampling a fixed-size neighborhood for each node during training. It learns an aggregator function to combine the features of the target node with its neighbors, allowing the model to generalize to unseen nodes.
- Graph Attention Networks (GATs): GATs [7] use an attention mechanism to weigh the contributions of neighboring nodes when aggregating information. This enables the model to focus on the most relevant neighbors, leading to better performance on tasks where certain relationships are more important than others.
- Graph Isomorphism Network (GIN): GIN [8] is designed to be more expressive than other GNN models, making it capable of distinguishing between different graph structures. It uses a learnable aggregation function and is particularly suitable for tasks that require capturing complex graph patterns.
- Equivalent Graph Neural Network (EGNN): EGNN [9] is a more recent type of GNN model designed to handle relational reasoning tasks on graphs with continuous node features and pairwise relations. EGNNs are particularly suitable for problems involving dynamics in graphs, such as physical systems, as they can naturally represent and learn from pairwise interactions between entities.

Benchmark

Testing and benchmarking GNN inference involves evaluating the model's performance in terms of accuracy, speed, memory consumption, and scalability. Here's a step-by-step plan to test and benchmark GNN inference for particle identification.

1. Utility evaluation
    - Split the dataset into training, validation, and test sets.
    - Train various GNN models on the training set and fine-tune the hyperparameters using the validation set.
    - Evaluate the model's accuracy on the test set using standard metrics, such as precision, recall, F1-score, or area under the ROC curve (AUC-ROC).
2. Inference speed
    - Measure the time taken for the model to perform inference on a single graph or a batch of graphs.
    - Compare the GNN model's inference speed among each other used for the same task to determine its relative efficiency.
3. Memory consumption
    - Monitor the memory usage of the GNN model during inference, including GPU and CPU memory consumption.
    - Compare memory consumption to other models or methods used for the same task to determine if the GNN model is more resource-efficient.
4. Scalability
    - Test the GNN model's performance on increasingly larger graphs and datasets to assess its ability to scale with data size.
    - Identify potential bottlenecks or limitations in the model's design that may impact its scalability, such as memory constraints or computational complexity.

This benchmark information can be used to make informed decisions about deploying the GNN model in a real-world setting or identifying areas for further optimization and improvement. Besides, we can also benchmark the GNN model against other models or methods (CNNs) used for the same task, comparing their accuracy, speed, memory consumption, and scalability. We can hence identify the strengths and weaknesses of the GNN model compared to alternative approaches to determine its suitability for the specific task and potential areas for improvement.

Timeline Plan

According to the Google Summer of Code 2023 Timeline[1], the actual working period starts on May 29 and ends on August 28, which is approximately 12 weeks. Before the working period starts, I will get to know mentors, read documentation, and get up to speed to begin working on their projects from May 4 to 28. A detailed 12-week plan is presented as follows.

1. Week 1: Project initiation
    - Familiarize me with the project scope and detailed objectives after the discussion period with mentors from May 4 to 28.
    - Set up the development environment and required tools, including both the software packages and hardware configurations.
    - Review relevant literature on GNN models and their applications in particle physics if necessary.
2. Week 2: Data preprocessing and graph construction
    - Preprocess experimental data and convert it into jet images.
    - Develop several possible methodologies for constructing graphs from jet images, including defining nodes, edges, edge weights, and feature matrices.
3. Week 3: A baseline CNN model implementation
    - Implement a simple CNN model first to make the entire learning pipeline work.
    - Resolve unexpected and emergent issues, if any.
4. Week 4: GNN models implementation
    - Choose a suitable GNN model for low-momentum tau identification based on the literature review.
    - Implement the selected GNN architecture using a deep learning framework such as PyG (PyTorch Geometric).
5. Week 5: GNN models implementation (Cont.)
    - Implement other GNN architectures using a deep learning framework such as PyG (PyTorch Geometric).
    - Resolve unexpected and emergent issues, if any.
6. Week 6: Model training and evaluation

[1] `https://developers.google.com/open-source/gsoc/timeline?hl=en`

- Train the end-to-end GNN model for low-momentum tau identification using the constructed graphs and available training data.
- Evaluate the performance of the trained GNN models using standard metrics and compare them to existing methods.

7. Week 7: Model optimization
   - Migrate the model training from single GPU centralized training to distributed training across multiple GPUs.
   - Re-train and re-evaluate the optimized model.
8. Week 8: Buffer week
   - Resolve unexpected and emergent issues, if any.
9. Week 9: Testing and benchmarking
   - Test the trained GNN model's inference on GPUs, ensuring compatibility and proper functioning.
   - Conduct benchmarking to assess the GNN model's inference speed, memory consumption, and overall performance on GPUs.
10. Week 10: Model comparison and analysis
    - Compare the performance of the GNN model to other relevant models or methods used for the same task.
    - Analyze the strengths and weaknesses of the GNN model and identify potential areas for improvement.
11. Week 11: Documentation and code cleanup
    - Write detailed documentation for the project, including methodology, model architecture, training, evaluation, and benchmarking results.
    - Clean up and organize the codebase, ensuring it is well-commented and follows best practices.
12. Week 12: Final presentation and submission
    - Prepare a final presentation summarizing the project's objectives, methodology, results, and potential future work.
    - Submit the final project deliverables, including code, documentation, and presentation.

Deliverables

Upon completion of this project, the following results are expected.

- A trained end-to-end GNN model tailored for low-momentum tau identification, capable of capturing complex relationships between jet constituents and outperforming existing methods. (A Python project with Jupyter Notebook for illustration)
- A comprehensive evaluation and optimization of the GNN model, demonstrating its effectiveness and potential for integration with existing systems. (A detailed report or potential paper publication)

Why Me

Research Interests Matching

My current research interests are Graph Neural Networks and AI4Science, which are precisely the main area that this project focuses on. I have published several first-authored papers on GNNs, and one of my works on using GNNs for neutrino events classification and reconstruction in High Energy Physics(HEP) has been accepted to the Machine Learning and the Physical Sciences Workshop at the 35th Conference on Neural Information Processing Systems, 2021. We focus on the graph construction step via the score-based generative model to enhance the quality of the constructed graph for GNNs to operate on for downstream neutrino events classification in HEP.

Skills Matching

In my daily research, I use Python for the implementation of deep learning models a lot, with the help of PyG, and PyTorch packages. I have a decent experience in programming and Python in the industry as well. Therefore, I believe I have strong programming skills, particularly in Python, to handle this project. My Experience with deep learning frameworks, such as PyTorch and PyG, and my knowledge of graph neural networks with their applications in particle physics definitely help me finish this project.

Time Commitment

My summer vacation starts in the third week of May and ends in early September. I don't have any other commitments during that time. Therefore, I can spend around 350 hours on this GSOC task. I will work full-time on it throughout the summer for at least three months and will communicate with mentors regularly.

I choose to apply for this single project only and put all my efforts into the proposal and the test, even though I am suggested to submit more than one

proposal for ML4SCI organization. I am confident in myself, my skills, and above all, my enthusiasm. If given a chance, I will leave no stone unturned to excel in this project within the given period.

Link to my resume: Resume

Link to my solutions to the test: Test solutions

References

[1] M. Andrews, M. Paulini, S. Gleyzer, and B. Poczos, "End-to-end physics event classification with cms open data: Applying image-based deep learning to detector data for the direct classification of collision events at the lhc," *Computing and Software for Big Science*, vol. 4, pp. 1–14, 2020.

[2] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 021001, 2020.

[3] Allyyi, "Ml4sci-gsoc-gnn/dataset-explore.ipynb at main · allyyi/ml4sci-gsoc-gnn." [Online]. Available: https://github.com/Allyyi/ML4SCI-GSOC-GNN/blob/main/dataset-explore.ipynb

[4] S. Schätzel, "Boosted top quarks and jet structure," *The European Physical Journal C*, vol. 75, pp. 1–68, 2015.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[7] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

[8] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[9] V. G. Satorras, E. Hoogeboom, and M. Welling, "E (n) equivariant graph neural networks," in *International conference on machine learning*. PMLR, 2021, pp. 9323–9332.