Machine Learning for Science

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

Proposal - Google Summer of Code 2023

PERSONAL DETAILS

- Name: Sarith Imaduwage
- Telephone: +94 777041978
- CV: Click Here
- Email: imaduwagesarith932@gmail.com
- GitHub: https://www.github.com/SarithRavI
- Country of Residence: Sri Lanka
- Timezone: IST (Sri Lanka)
- Language: English

PROJECT PROPOSAL

Project Title

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

Prerequisite Tests

View **here**

Abstract

One of the important aspects of searches for new physics at the Large Hadron Collider (LHC) involves the identification and reconstruction of single particles, jets and event topologies of interest in collision events. The End-to-End Deep Learning (E2E) project in the CMS experiment

focuses on the development of these reconstruction and identification tasks with innovative deep learning approaches.

This project will focus on the development of end-to-end graph neural networks for particle (tau) identification and CMSSW inference engine for use in reconstruction algorithms in offline and high-level trigger systems of the CMS experiment.

Background

I am a fourth-year undergraduate at General Sir John Kotelawala Defence University (KDU). I am a FOSS (Free and Open Source) enthusiast involved in open-source projects that include production standard integrations. I am widely ***interested in the domain of Graph Representation Learning*** with skills in implementing state-of-the-art models and syncing with state-of-the-art research. I have a decent ***knowledge of topics like the expressivity and scalability of GNN*** which I assume will be very useful when carrying out this project.

I did my internship at WSO2 Inc., the number one open-source integration vendor, where I had the opportunity to integrate WSO2's main PaaS product Choreo with the analytics dashboard Moesif. I was the chairperson of the IEEE Student Branch of KDU where we voluntarily conducted many programs that broadened technical skills in a wide range of domains of students at my university, which led us to be recognized as an Exemplary Student Branch of the Region. I also mentor hundreds of Juniors in topics like Machine Learning, Deep Learning, and DevOps/MLOps.

Speaking of my Tech stack, I have a strong background in C++, Python, and Java. I also have experience with MLOps tools like Django REST, and FastAPI for implementing RESTful APIs and Docker, Kubernetes (K8), and CI/CD tools for deployments.

I have a ***deep passion for researching*** and coming up with novel ideas that extend the state-of-the-art. Currently, I am involved in contributing to a literature survey conducted by an international university on the topic of *GNNs' performance in Facial Landmark Detection*. At the same time, I am working on extending some of my previous works on *Graph Representation learning-based Fake News Detection*.

Some selected publications

1. Capturing Credibility of Users for an Efficient Propagation Network Based Fake News Detection
   2022 2nd International Conference on Computer, Control and Robotics (ICCCR)

2. The Mismatch between Graph Neural Networks' Expressivity and Propagation Graphs in Fake News Detection
2022 IEEE/ACIS 7th International Conference on Big Data, Cloud Computing, and Data Science (BCD)

Tasks

- My first task is determining feasible *images (raw data) to graph techniques*. Given the nature of prerequisite tests of the project it is safe to assume that a *pruning* of image data will be required before building graphs.

- I should explore these techniques restricting to consider methods that only preserve *relational information* of the image data points (pixels). Hence it is crucial to understand any existing underlying relational structure in images by having discussions with mentors.

- Each feasible image-to-graph technique may produce graphs with unique structural characteristics. For a motivating example, graphs produced by a particular technique could have graph motifs that are not present in graphs produced from a different technique. Hence high level analysis of graph structures should be added to the task list.

- Expressivity of GNNs i.e., capability of distinguishing two non-isomorphic graphs, is bounded mainly by two factors (1) node features and most important (2) graph structure. Based on insights on graph structures, I can categorize graphs produced by different techniques into basically two classes:

    A. Graphs with structures that are distinguishable by GNNs that have expressivity at most of the Weisfeiler-Lehman Isomorphism Test (1-WL). Let's denote such GNNs as type-A.

    B. Graphs with structures that need GNNs that have expressivity that exceeds the 1-WL test. Transformer based GNNs (Graphormers) have such higher expressivity. Let's denote such GNNs as type-B

- By this stage I can build suitable GNNs for different graphs yielded from different image-to-graph techniques.

- These experiments can be managed by GraphGym.

- Hyperparameter tuning. (I can use general suggestions for type-A from this paper and for type-B from this). GraphGym will come in handy when hyperparameter tuning.

- According to the previous discussion with mentors, this inference pipeline can be deployed. I assume that at the end this inference pipeline should be integrated into an existing inference engine or a production-level software (e.g., CMS Offline Software)

- Understand such integration points and discuss them with mentors.
  Currently, I have a rough idea of the integration points that I got from going through this [*documentation.*]

- Guaranteeing code readability by adding comments and documentation.

- Producing a comprehensive report that will state research outcomes and matters related to development.

Timeline

Please go through the **Tasks** section before this section.

★ Community Bonding Period (4th May - 28th May):

a. List all the tasks and subtasks precisely. As can be seen in the previous section, a task can have many subtasks. Hence it is important to scope these subtasks.
b. This scoping should be backed by a literature survey. (I started the literature survey before the beginning of the program)
c. Discuss with mentors and put together precisely components of my proposal, and include the necessary changes.

★ Week 1 & 2 (29th May - 11th June):

a. Analyze the raw data.
b. Converting images (raw data) to graph data such that underlying semantic and relational information between data points (pixels) is preserved.
c. Pruning raw data if necessary.
d. Discuss each decision regarding *a,b, and c* with mentors and make the changes.

★ Week 3 (12th June - 18th June):

a. By now, we would probably have several feasible raw image-to-graph conversion techniques. We can categorize each technique based on the structural features of yielding graphs.
b. Such a broad categorization is:
   - Graphs that are classifiable with GNNs with at most 1-WL expressivity.
   - Graphs that need expressivity beyond 1-WL.

   For instance, if the graph is sparse or tends to occur bottlenecks, then we put such graphs into the second category.
c. Studying the feasibility of managing experiments with GraphGym.

★ Week 4 & 5 (19th June - 2nd July):

a. Assimilate the changes and improvements suggested by mentors.
b. Building GNN models and starting evaluation.
c. Managing experiments with *GraphGym*.
(Note: Initial hyperparameters can be set to general design choices in the case of type-A GNNs as per mentioned [here]() and in case of type-B, design choices mentioned in [here]()can be used.)

★ Week 6,7 & 8 (3rd July - 23rd July):

a. Assimilate the changes and improvements suggested by mentors.
b. Continuation of hyperparameter tuning.
c. Continuation of benchmarking of end-to-end GNN inference on gpu.
(I dedicate approximately 4-5 weeks for model selection alone to obtain the best model and to report on my model selection procedure.)

★ Week 9 (24th July - 30th July):

a. By the beginning of 9th week, I have benchmarked feasible GNN models and am able to select the best *raw data-to-graph* method.
b. Upon agreement of mentors, begin deployment by studying integration points.
c. This integration could be a simple code-base migration to a complex task. *Depending on that I dedicate time to study the architecture/ design patterns of the inference engine / prod. software.*

★ Week 10 (31st July - 6th August):

a. If it is determined to be rather a simple integration, then I will additionally work on optimizing the *raw data-to-graph* pipeline. Such optimization can be having concurrency in execution.
b. Otherwise, I will only make necessary changes that will integrate the inference pipeline successfully.

★ Week 11 (7th August - 13th August):

a. Cleaning the code for better readability and adding documentation/ comments.
b. Continuation of integration if necessary.
c. Integration testing.

★ Week 12 (14th August - 20th August):

a. Finalizing a comprehensive report mentioning research, development and deployment outcomes/results.
b. Discussing future works such as producing research papers with mentors.

★ Final Evaluation (21st August - 28th August)

Note on Availability

Please go through the **Timeline** section before this section.

- Above timeline is made for 12 weeks. Depending on necessity or complexity of *deployment* this timeline can get reduced to 10 weeks or get extended to 22 weeks. Hereby **I confirm my availability for additional weeks** in addition to the aforementioned 12-week timeline.

- This statement is based on the requirement that the project timeline should span a minimum of 12 weeks and a maximum of 22 weeks according to these instructions.

Prior Experience with Machine Learning

I started diving into Machine Learning when I was in my sophomore year of graduation. From the beginning I tried to lay a good foundation in mathematical intuition behind concepts and practical implementation of Machine Learning Models. In the same year I trained a Support Vector Machines (SVM) based model to predict severity of a traffic accident when common environmental factors are provided. This model was deployed in a larger Traffic Accident Management System.

In the following year (2021), I drew my attention toward Deep Learning. Starting from the Artificial Neural Networks, I gradually learned Convolutional Neural Networks, Recurrent Neural Networks etc., both theoretical aspects and implementation. My first exposure to a deep learning framework/ library was Tensorflow. Along with this learning of fundamentals I cultivated a habit of reading state-of-the-art research and syncing up with research community.
By the beginning of 2022, I really started focusing on a single Deep Learning paradigm, and having to know that graphs are the most fundamental data structure fed into any Deep Learning model, I drew attention toward Graph Neural Networks. Simultaneously I improved my skills in Pytorch, because in my opinion, Pytorch Geometric is the most sophisticated library for ML on graphs. By the mid of last year, I was able to publish 2 research papers that cover topics in ML like feature selection, Variational Inference, Performance benchmarking of GNNs, expressivity and scalability of GNNs.

Selected courses I have taken

1. **Machine Learning (CS229) by Stanford University - Link**
2. **Machine Learning with Graphs (CS224W) by Stanford University - Link**
3. **Machine Learning Engineering for Production (MLOps) Specialization - Link**

Selected courses I am currently enrolled

1. **Probabilistic Deep Learning with TensorFlow 2 - Link**
2. **ETL and Data Pipelines with Shell, Airflow and Kafka - Link**

Reading groups, I am currently joined

1. **Learning on Graphs and Geometry - Link**

Previous Contributions to Open Source

- Pyg-face_landmarks - Here I benchmark performance of different GNN architectures such as GCN, GIN & GAT etc. on the task of facial landmark detection. This is an open source contribution to a literature survey conducted at Monash University (Malaysia).

- FakeNewsNet-extended - Benchmark Dataset for Propagation Based Fake News Detection.
    a. Preparing a complete benchmarking dataset using Twython and Python multiprocessing.
    b. This dataset is suitable for comprehensive analysis and comparison of existing Fake News Detection models. For more detail refer to my publication.

- Moesif-DataService - This is a microservice I wrote in ballerina language to integrate Moesif Analytics dashboard with WSO2 Choreo API manager during my internship at WSO2.

- During my internship at WSO2, I was involved in a production grade project to integrate one its main product, Choreo with an external API analytics dashboard called Moesif. During a span of 4 months I was able to:

    a. Make WSO2 API Manager cloud native such that it supports robust analytics generation.
    b. Make a publisher that is tolerant to heavy load with an inbound event queueing mechanism.

c. During this period, I made 40+ PRs (Pull Requests in Github) and 5000+ lines of code.

d. Here are the public repositories of WSO2 that I contributed:

    - apim-analytics-publisher
    - carbon-apimgt
    - Product-microgateway

Social Links

LinkedIn: Click Here
GitHub: Click Here
ResearchGate: Click Here
Facebook: Click Here