## Basic Information

**Name:** Vigneshwar Punna

**Email:** ajaypunna9342@gmail.com

**University:** Maturi Venkata Subba Rao Engineering College, Hyderabad, Telangana

**Major:** Computer Science Engineering (Bachelor of Engineering)

**GitHub:** Vignesh142

**Location:** Hyderabad, Telangana, India

**Time Zone:** IST( UTC + 5:30)

## Project

Graph Neural Network for E2E Particle Identification with the CMS Experiment

## Abstract

The project aims at Classifying the particles by using Deep Learning Models 125x125 matrices (three channel images) for two classes of particles quarks and gluons impinging on a calorimeter. Popular Deep Learning frameworks like Tensorflow/Keras and PyTorch are used to create a model. A Graph Neural Network Model to be prepared to classify the particles in the CMS Experiment. And compare the two model architectures- CNN and GNN with their accuracies.

## Project Proposal

I propose to implement a Deep Learning Graph Neural Network model that predicts the smallest particles quarks and gluons with their properties of them read on a calorimeter with more accuracy. I shall also compare the performance of the GNN architecture and CNN model.

Graphs can be used to represent a lot of real-world datasets such as social networks, molecular structures, geographical maps, weblink data, natural science, protein-protein interaction networks, knowledge graphs, etc. Also, non-structured data such as images and text can be modeled in the form of graphs. Graphs are data structures that model a set of objects(nodes) and their relationships (edges).

As a unique non-Euclidean data structure for machine learning, graph analysis focuses on tasks like node classification, graph classification, link prediction, graph clustering, and graph visualization. Due to its good performance in real-world problems involving non-Euclidean space, Graph Neural Networks have become a widely applied graph analysis method recently.

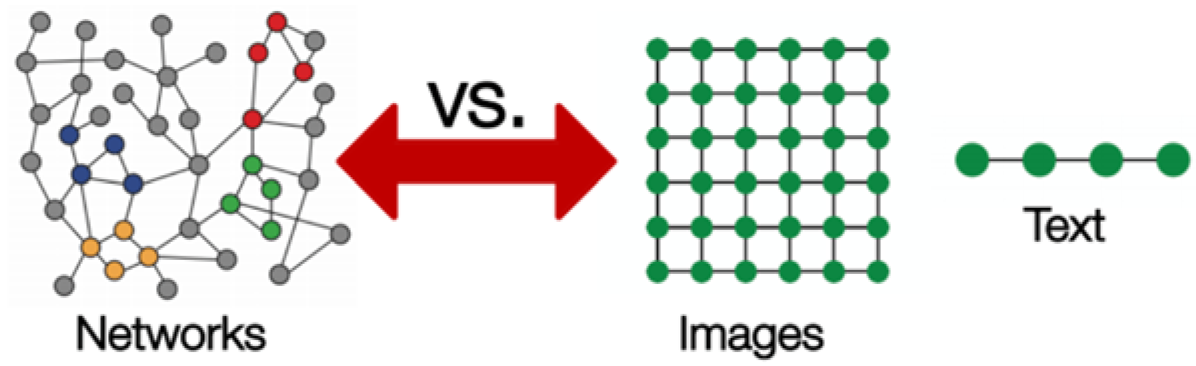


Fig.1.GNN vs Eucledian_space architecture

## Graph Neural Networks Algorithm-

A node can be represented by its features and the neighboring nodes in the graph. The target of GNN is to learn a state embedding, which encodes the information of the neighborhood, for each node. The state embedding is used to produce an output, such as the distribution of the predicted node label.

GNNs is a combination of an information diffusion mechanism and neural networks, representing a set of transition functions and a set of output functions. The information diffusion mechanism is represented by nodes in which their states are updated and information is exchanged by passing “messages” to their neighboring nodes until they reach a stable equilibrium. The transition function takes the features of each node, the edge features of each node, the neighboring nodes’ state, and the neighboring nodes’ features as input and the output is the nodes’ new state.

GNNs started getting popular with the introduction of the Graph Convolutional Network (GCN) which borrowed some concepts from the CNNs to the graph world. The main idea of this kind of network is also known as the Message-Passing Framework. The Message-Passing framework states that, for every node in our graph, we will do two things:

- Aggregate the information from its neighbors
- Update the node information with the information from its previous layer and its neighbor aggregation

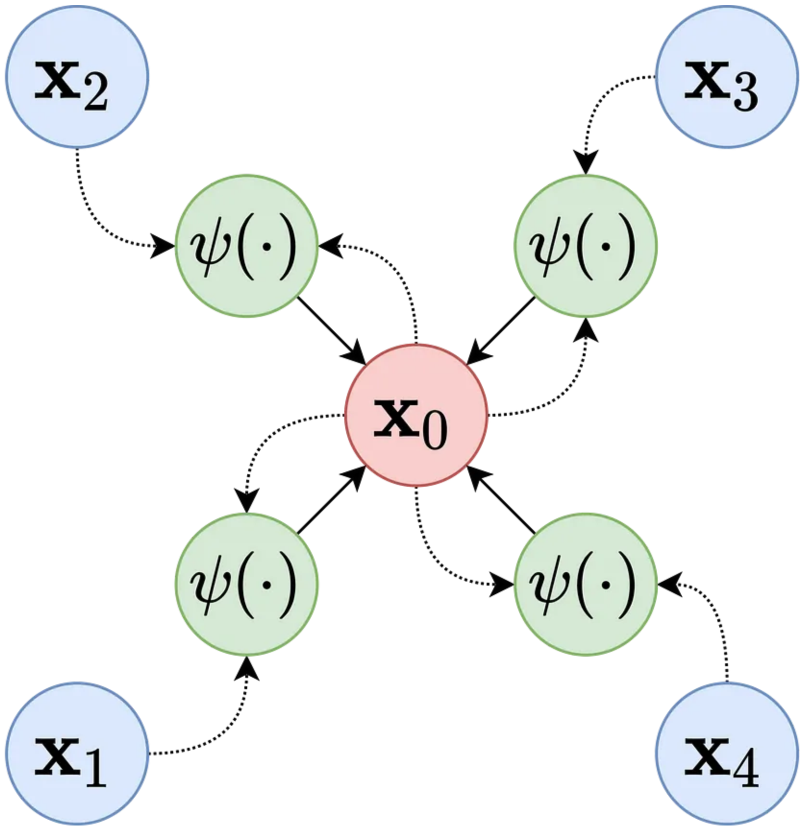


**Fig.2.Messaging passing Framework**

Figure 2 shows how this general framework works. Many architectures developed after GCN focused on defining the best way of aggregating and updating the data.

## Technical Background

I knows the programming languages C, C++, Python, Java, and Kotlin. I have done DSA in C++ and solved problems in LeetCode, and HackerRank. Besides, I learned AIML by taking a course. From then I have started learning AIML and Deep Learning. Initially, I learned Python and libraries like Numpy, pandas, and matplotlib.

Later on, I learned machine learning and deep learning algorithms and frameworks. I have done various projects on machine learning in  Supervised and Unsupervised Learning - Linear Regression, Logistic Regression, KMeans Clustering, KNN, Decision trees, Random Forest, etc also done Deep Learning projects.

I have participated in technical events, and hackathons conducted in our college to gain some experience to work as a group and knowledge.

## Why did I Choose this project?

I came to know about GSOC, suggested by my friend. I am new to open-source and pretty excited to work with organizations and contribute to open-source projects.

As soon as I came to know that GSOC 2023 was going on, I started to search for organizations. And finally, I have chosen your organization ML4SCI. I found it best to work more on deep learning and support your organization and build various deep learning models to predict the fundamental particles and deploy models in CMS Electromagnetic Machines.

I have previously worked on Deep Learning neural networks like CNN, but now in GNN. So I decided to prepare a GNN model for this project and aim to prepare the best model with high accuracy. I have no prior experience in working on this architecture. I would learn to implement it and prepare a GNN model at the time of this project. So, I thought that this would be the best organization chosen and the best idea to work on.

## Previous Work

- I am new to open source and did not contribute to open source before. But, I have done various projects in Machine Learning and Deep Learning working on various algorithms. I have participated in Hackathons from College, taken course training from Rinex Technologies on ML and  DL, and learned from free courses on Youtube.
- Some of my projects include Image Classification, Face Recognition, Emotion Recognition in text i.e. Spam Detector, Computer Vision-Editing text, Text Recognition from Images, and also on Machine Learning Algorithms.
- My previous work is available in my git repositories. My previous work on project repositories is attached below-

  https://github.com/Vignesh142

## Tasks

I have worked on Common Tasks 1 and 2 in the End-to-End Deep Learning Project.

- Common Task 1. Electron/photon classification
- Common Task 2. Deep Learning-based Quark-Gluon Classification

I have gone through the given datasets to prepare the Deep Learning model. I have prepared the model using the two frameworks as specified which are TensorFlow/Keras and PyTorch. I have used 80 percent of the data to train the model and 20 percent to test the model and I have the accuracy of the TensorFlow model above 80 percent.

In the second task, I prepared the model using the Convolutional Neural Network(CNN) architecture.

I have trained the whole model on my laptop with a CPU. I have also done the visualization of the model and the datasets. The link to the projects I have shared in my GitHub repositories are attached below-

github.com/Vignesh142/ML4SCI_Electron-photon-classification

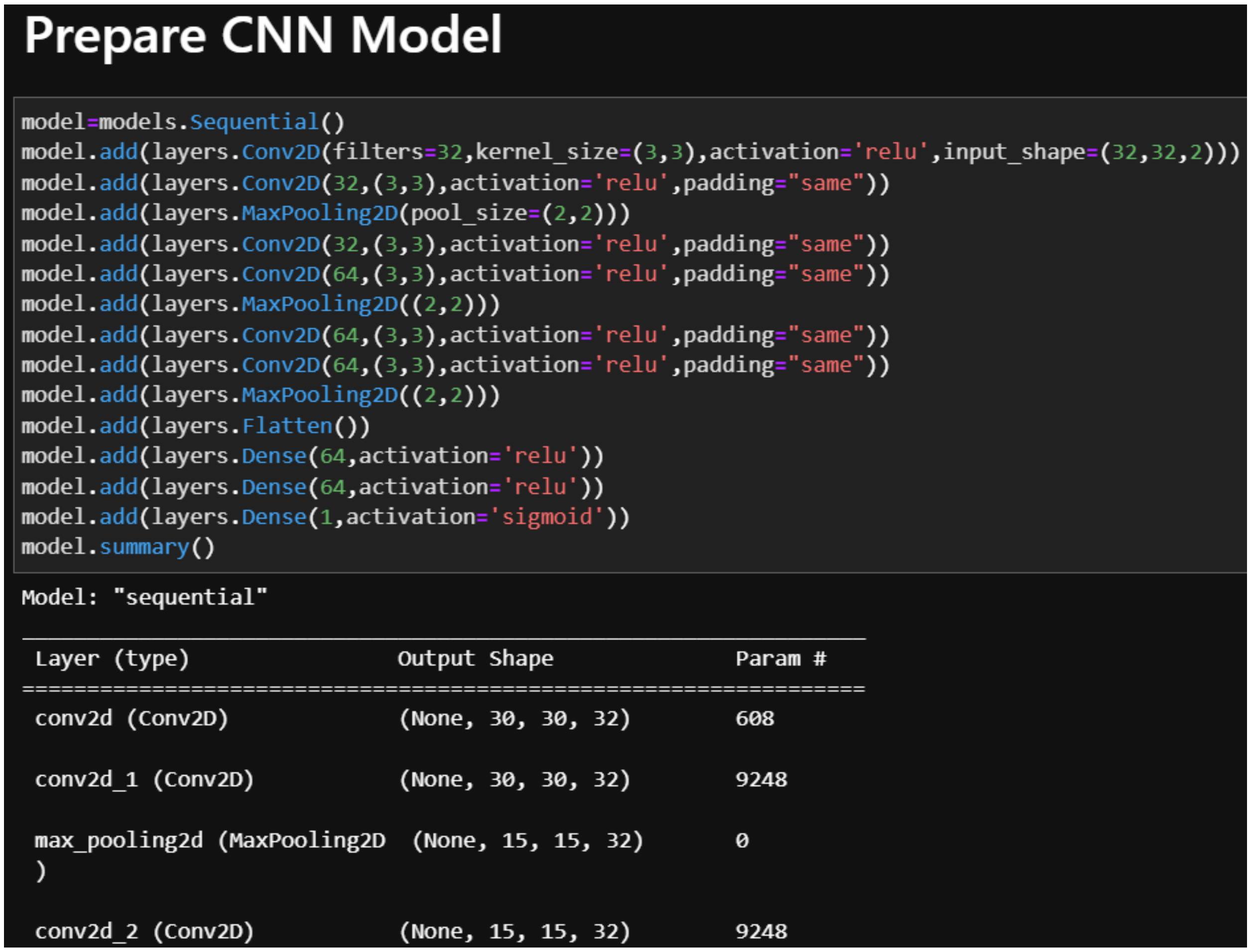
github.com/Vignesh142/ML4SCI_-Deep-Learning-based-Quark-Gluon-Classification

The Process flow:

- Firstly, I have imported libraries
- Loaded the data from the given dataset.
- Visualized data
- Splitting the dataset for Training, testing, and validation
- Finding and removing null and duplicated values
- Prepared the model architecture of the neural network
- Define loss function and optimizer
- Train the model and save
- Testing and finding analyze the accuracy

I have done with the model, but the problem I have faced was with the CPU and RAM. Mine was 32GB so it couldn't load data at a time so I have trained a part- saved and retrained it and looped.

Sample Code of CNN Model architecture:



Prepare CNN Model

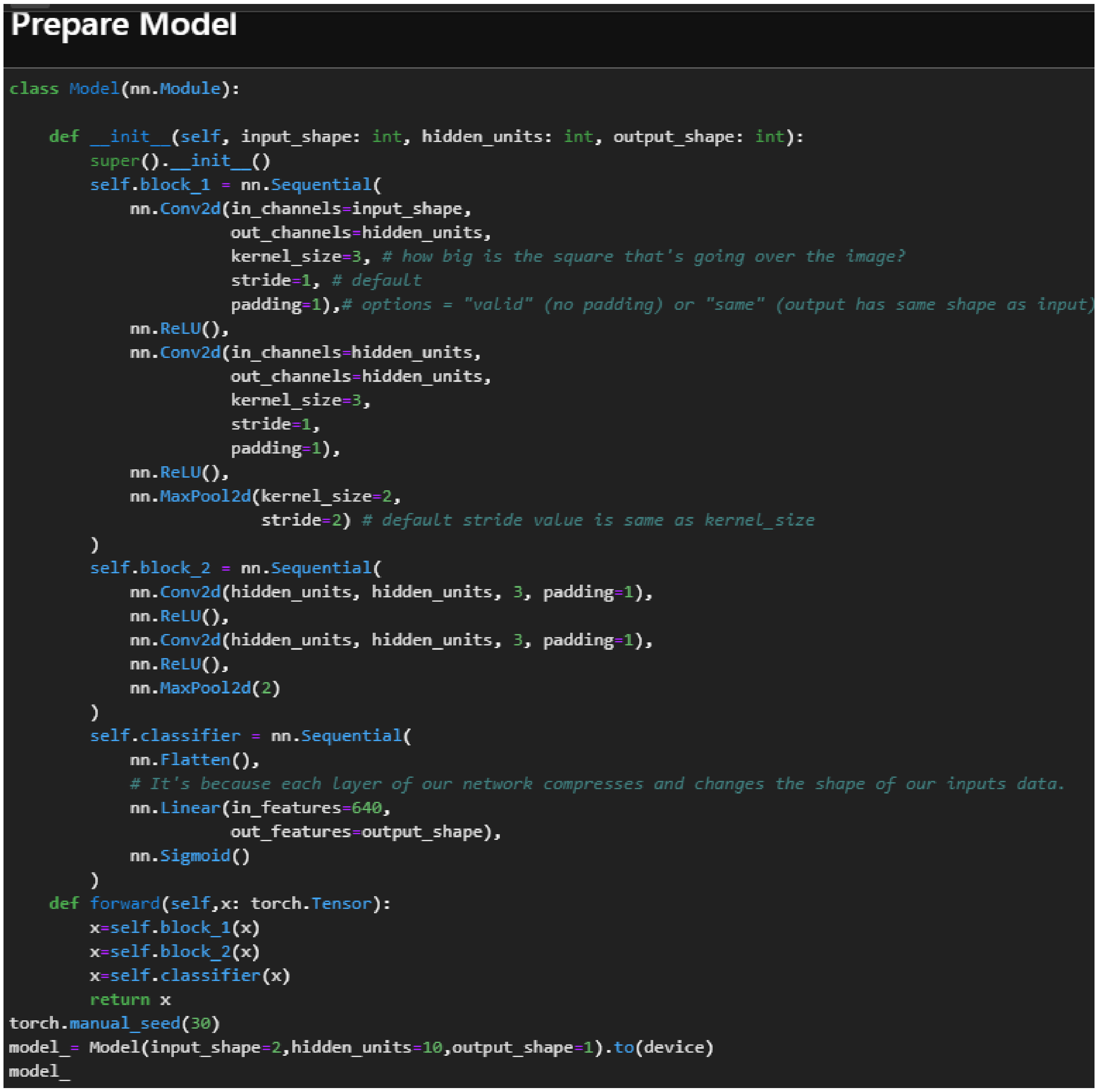
```
model=models.Sequential()
model.add(layers.Conv2D(filters=32,kernel_size=(3,3),activation='relu',input_shape=(32,32,2)))
model.add(layers.Conv2D(32,(3,3),activation='relu',padding="same"))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Conv2D(32,(3,3),activation='relu',padding="same"))
model.add(layers.Conv2D(64,(3,3),activation='relu',padding="same"))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation='relu',padding="same"))
model.add(layers.Conv2D(64,(3,3),activation='relu',padding="same"))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        608

 conv2d_1 (Conv2D)           (None, 30, 30, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 15, 15, 32)        9248
```

Fig.3.CNN Model

The above code is the architecture used to train the model. The above one is done in the TensorFlow/Keras framework. And also done in the Pytorch framework attached below.

20 percent of the data is used to test the model and in the remaining 80 percent of the model, 10 percent is used for validation and another to train the model.
The Binary Cross Entropy Loss function is used and the Adam optimizer is used.
Finally, I have trained the model and saved it. And after testing it, I have got the accuracy of the model above 80 percent.



## Prepare Model

```
class Model(nn.Module):

    def __init__(self, input_shape: int, hidden_units: int, output_shape: int):
        super().__init__()
        self.block_1 = nn.Sequential(
            nn.Conv2d(in_channels=input_shape,
                      out_channels=hidden_units,
                      kernel_size=3, # how big is the square that's going over the image?
                      stride=1, # default
                      padding=1),# options = "valid" (no padding) or "same" (output has same shape as input)
            nn.ReLU(),
            nn.Conv2d(in_channels=hidden_units,
                      out_channels=hidden_units,
                      kernel_size=3,
                      stride=1,
                      padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2,
                         stride=2) # default stride value is same as kernel_size
        )
        self.block_2 = nn.Sequential(
            nn.Conv2d(hidden_units, hidden_units, 3, padding=1),
            nn.ReLU(),
            nn.Conv2d(hidden_units, hidden_units, 3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.Flatten(),
            # It's because each layer of our network compresses and changes the shape of our inputs data.
            nn.Linear(in_features=640,
                      out_features=output_shape),
            nn.Sigmoid()
        )
    def forward(self,x: torch.Tensor):
        x=self.block_1(x)
        x=self.block_2(x)
        x=self.classifier(x)
        return x
torch.manual_seed(30)
model_ = Model(input_shape=2,hidden_units=10,output_shape=1).to(device)
model_
```

Fig.5. PyTorch CNN Model

In the Second task, I used the TensorFlow FrameWork to train the model. I have constructed the Convolutional Neural Network and passed the (125,125,3) shape array image as input to the model and single output layer that predicts the particle is Quark or Gulon. The data input given is “X_jets” which has the image 3D arrays.
And the label data “y” is passed through the model to classify the two particles. The roc accuracy of above 80 percent has got.

The sample input data images passed through the model are shown below-

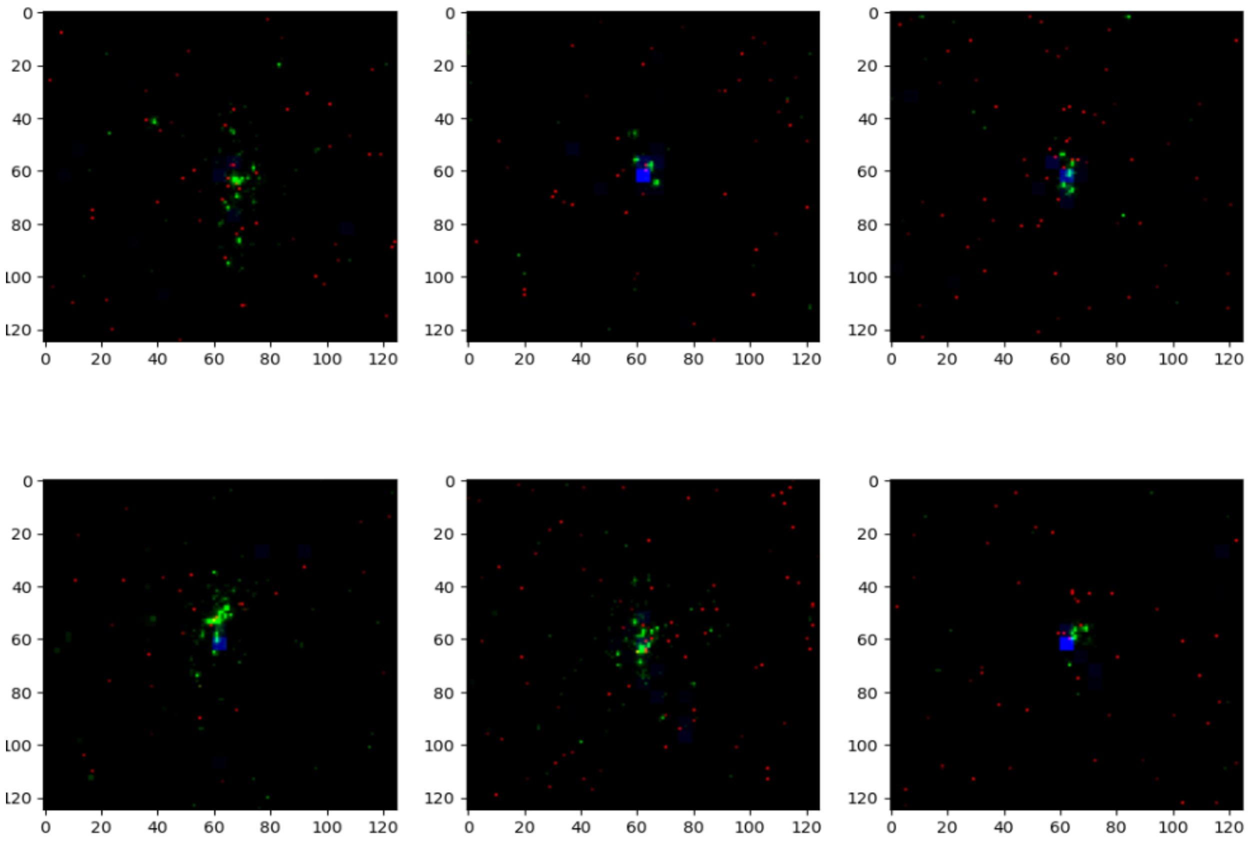


Fig.6. ‘X_jet’ Image Data Sample

The model is trained on CPU rather than GPU as it is not available, the model is trained efficiently but the time taken to train the model is more. There are also two other parameters ‘m0’ and ‘pt’ given in the dataset. They couldn’t be trained with the X_jet data, So I have prepared another model using the Machine Learning algorithm KNN Classification. I took both pt and m0 as input and y as the label and then trained the model. With this model, I have got an accuracy of above 65 percent but not more than the X_jets CNN Model. So finally, I took the CNN model and left the other parameters considering only the ‘X_jet’ parameter and predicted the model.

## Official Overview

- May 4**:** Accepted GSoC contributor projects announced
- May 4 - 28: Community Bonding Period | GSoC contributors get to know mentors, read documentation, and get up to speed to begin working on their projects
- May 29: Coding officially begins!
- July 10: Mentors and GSoC contributors can begin submitting midterm evaluations
- July 14: Midterm evaluation deadline (standard coding period)
- July 14 - August 2:  Work Period | GSoC contributors work on their project with guidance from Mentors
- August 21- 28Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)

## Details

This is merely a modest sketch. I have tried to be as lenient as possible in assigning the weekly tasks. The last 1-2 days of a Week will usually be reserved for code review and documentation or buffers to complete work that has not been completed in the allotted time. Though I intend to keep in touch with the mentor throughout the week and ensure that I am going in the right direction, I have tried to maintain a feasible feedback mechanism in the schedule to ensure that my mentor gives feedback on a reviewable quantity of code at appropriate intervals.

Documentation: The documentation for each week's work shall be done by the end of that week only. In any case, any code shall not stay undocumented for more than 1-2 weeks.

Tests:  The given data will be tested in the model created using vision transformers. The prediction is made with the model and also compared with the other models like neural networks CNN architecture in different frameworks. The accuracy score is predicted and my best try is to have the required accuracy score for trained models.

The models prepared may be of TensorFlow or PyTorch which I would be following. The testing results will be computed and the metrics will be shown briefly.

Model Training: The models are trained on the GPU and CPU and find the benchmark score of the transformer model and find the best-predicted model.

Duration:

Total project length:  175 Hours

Community Bonding period:

- Read and brush up on the relevant theory needed to implement the GNN model.
- Get a working knowledge of the preparing GNN model.

Week 1 (29th May - 4th June):

- Set up the kernel in the Jupyter Notebook to use the required versions of the libraries.
- Download the given datasets and import them into Jupyter Lab.
- Writing the code for the data extraction from the dataset and visualizing the datasets. Removing the redundant data from the datasets.
- Analyze the data.

Week 2 (5th June - 11th June):

- I shall learn about the GNN. A deep study on GNNs like how they work, how they implement models, and how to create a transformer model.

Week 3 (12th June - 18th June):

- Continue to learn the basic implementation of the GNN model.
- Learning the mechanism of GNNs- directed and undirected graphs, nodes- Graph data

Week 4 (19th June - 25th June):

- Begin implementation of the GNN model.
- Implement the pipelines

Week 5 (25th June -  2nd July):

- Understand the libraries in PyTorch like PyG
- Begin training the transformer model.

Week 6 (3rdth July - 9th July):

- Use existing pipelined datapath from a design and identify key modules.
- Arrange the testing data and start testing the implemented model.

Week 7 (10th July -  16th July):

- Predict the data with the model and find the accuracy of the model.
- Visualize the prediction accuracy using different metrics.

Week 8 (17th July - 23rd July):

- Exhaustive chat with mentors regarding mid-term evaluation and some parts of the code that I might need to improve upon.

Week 9 (24th July - 30th July):

- Recreating the best model updating with improved code implementation and testing.
- Identification of utilities to be implemented and discussion with the mentor to finalize a list of them.

Week 10 (31st July - 6th Aug):

- Comparing the models prepared with the GNN architecture and the previously made Convolutional Neural network model and analysis.
- Writing the tests for the utilities.

Week 11 (7th Aug  - 13th Aug):

- Exhaustive documentation writing begins.
- Documentation and Buffer to complete any remaining parts.

Week 12 (14th July - 20th Aug):

- Ideally, a week to complete any remaining documentation of the code.
- Buffer week for analyzing the suggestions by the community and talking to the mentor regarding implementing the same.
- Implementing the suggestions.
- Final evaluation exhaustive chat with mentors and feedback for the change in the code needed.

## Future Deliverables

I would like to work further with your organization and help improve the predictions with new models working on different methods and algorithms. I am very happy working with open source, interacting and discussing with new community members. I want to do more projects and contribute to your organization.

## Concurrent Commitments in Summer

I have no other commitments this Summer. I regularly attend college lectures and college training. Apart from that I currently do not have any commitments. I would spend around 30 to 35 hours a week on the project according to what I have planned. I think it would be quite enough to work on the project and complete the project in time.