The ‘subcaption’ package does not work correctlyin compatibility mode
subcaptionsubcaptionfont+=small,labelformat=parens,labelsep=space,skip=6pt,list=0,hypcap=0
*subcaption 1 subtypelist

Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Master Thesis

Title of the Thesis

Author: student name (student number)

1st supervisor: supervisor name
daily supervisor: supervisor name (company, if applicable)
2nd reader: supervisor name

A thesis submitted in fulfillment of the requirements for
the joint UvA-VU Master of Science degree in Computer Science

October 4, 2020

"I am the master of my fate, I am the captain of my soul"
from Invictus*, by William Ernest Henley*

Abstract

Here goes the abstract of this thesis.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

1

Introduction

The KM3NeT or the Cubic Kilometer Neutrino Telescope is currently being constructed at the bottom of the Mediterranean Sea. The goal of this telescope is two fold: first is to study high energy neutrinos originating from celestial events such as birth of a neutrino star or a supernova. And second, to study the properties of the neutrino particles produced in the Earth's atmosphere (1). The first goal will be realized with the KM3NeT/ARCA (Astroparticle Research with Cosmics in the Abyss) telescope and the second with KM3NeT/ORCA (Oscillation Research with Cosmics in the Abyss) (1). In this paper, we talk exclusively about KM3NeT/ARCA.

The ARCA telescope comprises of two "blocks" with a total volume of $1km^3$. Each block consists of 115 spherical detector units (DOMs) and each DOM consists of 31 Photo Multiplier Tubes (PMTs) in various spacial arrangement. Figure 1 shows an artist's impression of ARCA, figure 1 depicts a DOM along with the PMTs inside it.

The PMTs are sensitive to light or photons, the analog signal for all hits above a certain threshold are digitized. This datapoint consists of a timestamp and the spatial orientation of the DOM (ie. x,y,z coordinates). The digital signals from all PMTs are arranged in 100ms "timeslices" and sent to the on-shore facility for further processing (2).

1.1 Situation of Concern

When the high energy neutrino particles interact with surrounding matter, produce an electron and a photon, this phenomenon is known as the Cherenkov Radiation or Cherenkov Light (3). This phenomenon is utilized in the KM3NeT telescopes to detect high energy neutrinos ie. the Cherenkov Light is detected by the PMTs. Unfortunately, there are

several sources of noise (in this case, the noise is other light sources), bioluminescense and decay of Potassium 40 (^{40}K) and atmospheric Muons being the primary sources (4).

1.1.1 State of the Art

Due to the high level of noise, data is generated at an extremely high rate of 25GB/sec (1). Due to this high data rate, it must be filtered and selectively stored for further analysis. The state of the art for this task are known as "Event Trigger" algorithms (1, 2). The existing event trigger algorithms namely *L1* and *L2* have limitations and can be improved upon (5).

There is a tradeoff between performance and quality of the event trigger algorithms to be realized here. The state of the art *L1* and *L2* algorithms are very performant with the ability to filter data in real time however their quality of filtration can be improved. Thus new event trigger algorithms must consider this trade off.

Efforts have already been made to improve the existing event trigger algorithms. (5) proposed and implemented a GPU powered pipeline which utilizes correlation and graph community detection to identify time slices that may contain neutrino hits whilst (4) suggests an alternate using convolutional neural networks.

Next chapter discusses the existing body of work on the matter. Emphasis is put on the work done by **(author?)** since this project seeks to directly improve upon their work.

1.1.2 User Requirements

The primary users of the ARCA are researchers who want to study high energy particles from outer space. The stakeholders are all member institutes involved in the project and by extension all scientists from these institutes who will be working with the data collected.

The requirements of the primary users (and stakeholders) with respect to the data acquisition pipeline are as follows.

UR1. **The accuracy of filtration must be extremely high.**

Time slices which are deemed important by event trigger algorithms are stored for further analysis and research. Failure to store time slices containing information from neutrino events can lead to loss of important data and thus a poor quality of research. On the other hand, since majority of the data generated is noise, the pipeline must be able to eliminate majority of noise to prevent storage of unnecessary and potentially useless data.

UR2. **Filtration should occur in real time.**

The state of the art event trigger algorithms are able to process data in real time. The proposed alternative ideally should maintain or improve upon it's predecessor's performance else provide a good trade off with data quality.

1.2 Research Question

This report intends to improve upon the GPU pipeline proposed by **(author?)**, an overview for which is presented in Chapter 2. Specifically this project wishes to answer the following research questions.

RQ1. **Can the existing GPU pipeline be improved using neural networks (NNs)?**

Improvement may be achieved by reducing the processing time of the pipeline or improving the accuracy of identifying important timeslices. This project focuses on achieving improvement via accuracy. The task of validating the runtime performance of the methods proposed in this paper is left to a separate project.

In order to answer **RQ1**, the following sub questions are formulated.

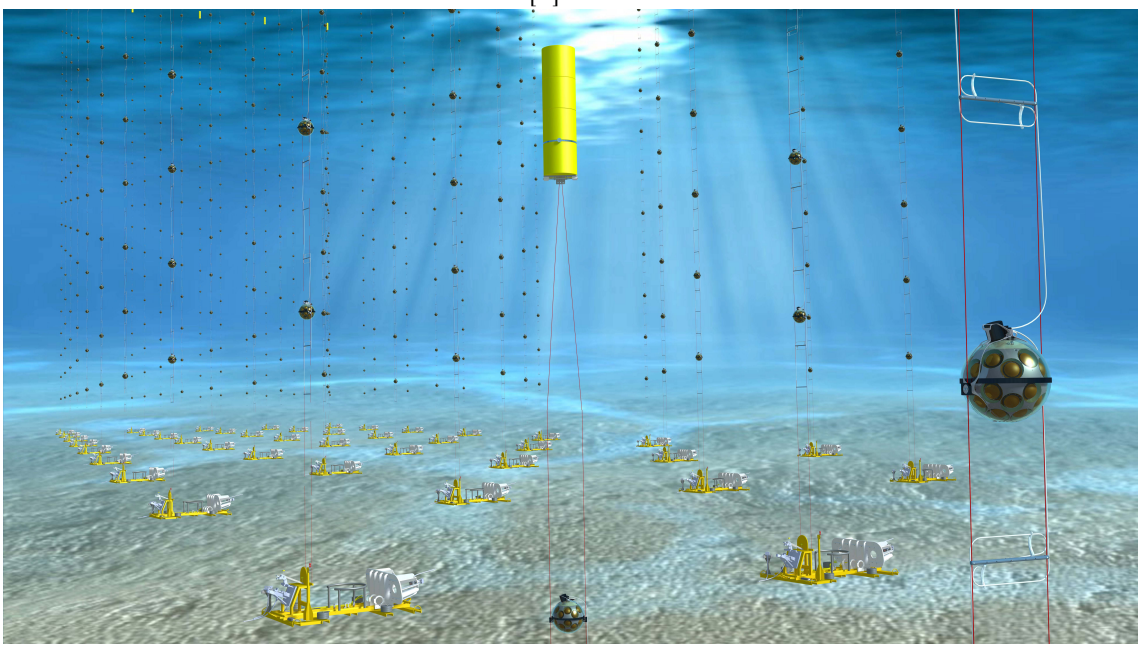
RQ2. Can the *Pattern Matrix Criterion* be replaced with a Multi Layer Perceptron?

(author?) proposed a novel trigger criterion which given a pair of points, can quantify the level of correlation amongst the points with an accuracy of 80%. The first phase of this project focuses on achieving better accuracy to identify "causally related" points using a Multi Layer Perceptron (MLP).

RQ3. Can the *Graph Community Detection* step be replaced with a Graph Convolusional Neural Network?

The output of the Pattern Matrix Criterion is used to create a graph structure where hits (of events and noise) are represented as nodes and causally related nodes are connected with an undirected edge carrying the probability of correlation as its weight. The Constant Pots clustering algorithm (**?**), which operates on the principles of Graph Community Detection, is used to separate the graph into communities of event and noise hits. The second phase of this project focuses on achieving a better accuracy for identifying event hits in a given timeslice using Graph Convolusional Neural Networks (GCNs).

[h]0.75



[h]0.25

2

The Karas Pipeline

This chapter presents the GPU pipeline proposed by (5) (henceforth referred to as the *Karas Pipeline* in more detail since the work presented in this report is directly based off of this seminal work. The chapter concludes by presenting the limitations of the Karas Pipeline and derives motivations for a better alternative as presented in the following chapters (see chapters 5 and 6).

(author?) presents a data processing pipeline to filter timeslices containing hits from neutrino events from those containing only noise. The pipeline is able to achieve this filtration by processing the data in 3 steps, described in more detail below.

The first step of The Karas pipeline is the Hit Correlation step. In this step, pairs of hits (from events and noise) are considered and the correlation along two axes namely space and time are considered. **(author?)** proposes *The Pattern Matrix Criterion (PMC)*. The PMC operates by creating a correlation criterion based on the probability that a given space and time difference occurs between two event hits. From domain knowledge, the evaluation is limited to 100m and 300ns for space and time differences respectively. The algorithm is evaluated with a dataset containing 130 event hits and 5000 noise hits and scores in the range of 0.3 - 0.375 is reported for the recall, precision and F1 metrics and an accuracy of 80% is achieved.

The second step of the Karas pipeline is the Graph Community Detection step. The Constant Potts Model (CPM) is used to group event and noise hits (represented as nodes of a graph) into separate communities (or clusters) based on the density of connections and the size of the communities. The output of the PMC is utilized to connect causally related nodes with an undirected edge and the probability of correlation is assigned as the weight of the edge. The model is tested using a dataset consisting of 130 event hits and 5000 noise

hits. The model is able to perform exceptionally well and groups most event hits into a single community and the noise in another. No performance metrics are reported however.

The third and final step of the pipeline is the Classification step. The two parameters namely the Probability Threshold (PT) of the PMC and the γ of the GCD step are experimented with to determine the optimal thresholds. All hits above the specified thresholds are classified as event hits the rest are classified as noise.

2.0.1 Limitations of the Karas Pipeline

The Karas Pipeline is able to identify timeslices with neutrino event hits more accurately compared to its predecessors such as the L1 and the L2 filtration. However, the pipeline still has certain limitations which hinders its performance, thus motivating a need for a better alternative.

To begin, the space and time difference based on which the PMC determines if two hits are causally related to one another, is static. This results in event hits which do not meet these thresholds to be incorrectly given a low probability of correlation to its sibling (related) hits.

The performance of the GCD step directly depends on the output of the PMC since the edges between causally related nodes and the weight it carries is determined from the PMC. Thus, any limitations of the PMC cascade down into the GCD and by extension, also into the classification step.

The CPM creates communities by observing the density of connections and the size of the communities. These thresholds are once again static which results in small communities being overlooked.

Finally, the classification step also operates on static values of the PT and γ and thus is unable to identify communities of size smaller than 20 hits.

3

Data Preparation

At the time of undertaking this project, the KM3NeT Neutrino Telescope was still under construction, thus simulated data provided by Nikhef was used for the project. The data itself was split onto two parts namely *events* and *noise*, both of which came from different sources and in different formats.

3.1 Preparation of *events* dataset

The *events* dataset was provided as a *HDF5* (Hierarchical Data Format) with a size of 42MB consisting of the `/data/mc_hits` and `/data/mc_info` tables. For the purposes of this project, the two tables were combined such that each row in the *mc_hits* table contains it's corresponding 'event_id' from the *mc_info* table. A `label` column was added containing a value of '1' and the resulting table (henceforth referred to as the *events* dataset) was saved as a CSV file for future use.

3.2 Preparation of *noise* dataset

The *noise* data was generated using a Python library written and maintained by Nikhef, `k40gen`. `k40gen.Generators(21341, 1245, [7000., 700., 70., 0.])` was used to create an instance of a generator where the first two arguments are random seeds followed by a list of rates at which single, double, triple and quadruple hits should be generated. The generator instance is then passed into `k40gen.generate_40()` method which returns a (4, n) array containing (time, dom_id, pmt_id, tot). The position coordinates (ie. `x`, `y`, `z` coordinates) for each datapoint was provided in a *positions.detx* file which was parsed using the Numpy Python package (6) and added to the *noise* array. The Python library

Pandas (7) was used to convert the array into a (n, 4) dataframe. A `label` column was added containing a value of '0' and the dataframe was saved as a 3.9GB CSV file.

3.3 Preparation of *main* dataset

To create the *main* dataset for the project, the *events* and *noise* datasets were combined. Both datasets were read into memory as Pandas dataframes and their columns were renamed consistently. The two dataframes were concatenated and sorted based on the `time` column. Rows with a negative `time` value were dropped along with columns which are not relevant to this project. The `time` column was discretized into 15000ns bins and the resulting values were added to the `timeslice` column. The resulting dataframe was saved as a 1.9GB CSV file.

4

Data Exploration

The *main* dataset (generated as per the steps outlined in Chapter 3) was explored using statistical analysis and visualizations to observe any patterns and "local trends" that may be present. The following chapter presents the analysis that were done and the observations made.

Note that a random sample of only 10% of the data was taken for the following visualizations. This is because it is difficult to draw reasonable conclusions from the plots due to the high number of data points when the entire dataset is used.

4.1 Descriptive Statistics

Table 4.2 presents the descriptive statistics of the *main* dataset. The dataset consists of 7 columns and roughly 4.5 million rows, Table 4.1 provides more information on the columns on the dataset. The dataset does not contain any `nan` or `null` values except for the `event_id` column where rows containing noise hits are not associated with any event.

Next, the correlations amongst features is checked, the "Pearson" correlation is used. Figure 4.1 represents the correlation matrix of all features. No significant correlations are observed between `pos_x`, `pos_y`, `pos_z` and `time` which indicates that ML models may not be able to learn anything from the dataset without the aid of feature engineering.

The distribution of the `label` column is presented in Figure 4.2. A severe class imbalance is noted between events and noise hits. To be precise, the dataset contains 489906 instances of events compared to over 4.5 million instances of noise. An effective strategy to handle the class imbalance will need to be devised during training of models to prevent the model from overfitting.

Table 4.1: Description of columns

Column	Data type	Unit	Description
pos_x, pos_y, pos_z	float	meters (m)	The position within the detector where the hit was detected, they represent the x,y,z coordinates of the hit respectively.
time	float	nano seconds (ns)	The time at which the hit was detected.
label	int	NA	The type of hit, '0' represents noise and '1' represents a neutrino hit
event_id	int	NA	The id of the event to which the hit is related to. The id itself does not have any meaning, it is simply used to identify hits that originated from the same event.
timeslice	int	NA	The id of the timeslice to which the hit belongs. The id itself does not have any meaning, it is simply used to group hits into discrete bins.

4.2 Verification of Bias

The *events* dataset is synthetically generated using simulations. As such, it is likely that the event hits in each timeslice may occur at a specific time such as at the beginning, middle or end of the timeslice. Having such a pattern in the dataset may bias the model since it may learn this pattern and thus fail to generalize. If this pattern does exist in the dataset, corrective measures need to be taken such that the event hits in each timeslice are uniformly distributed.

To verify the existence of such patterns in the dataset, the mean `time` of event hits across all events was visualized. Figure 4.3 depicts a scatter plot of mean `time` across events. A uniform distribution is noted with no visible patterns. Thus no bias exists in the dataset and is deemed suitable for further analysis.

4.3 Exploration of Interesting Timeslices

Figure 4.4 represents to total number of event hits per timeslice. The dataset is discretized into 6759 timeslices of which 2783 timeslices contain only noise hits. This is corroborated by Figure 4.4 which presents a long tail distribution where many timeslices contain few to

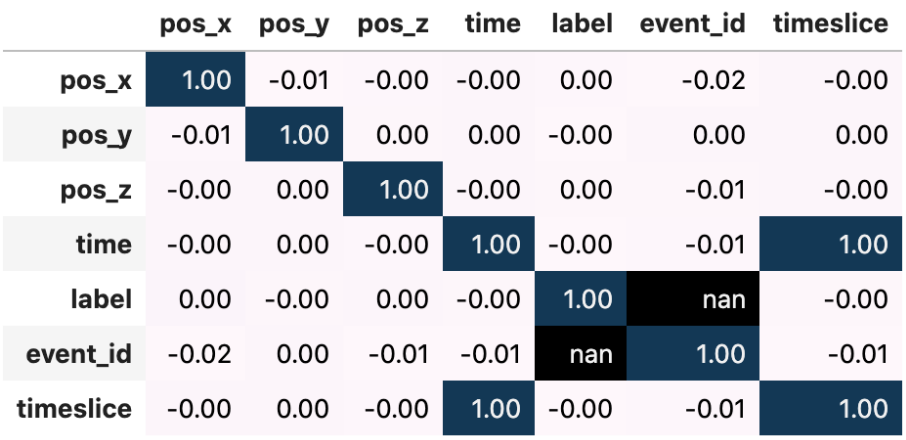


Figure 4.1: Correlation matrix of features

Table 4.2: Descriptive statistics

	pos_x	pos_y	pos_z	time	label	event_id	time slice
count	4.58e+7	4.58e+7	4.58e+7	4.58e+7	4.58e+7	489906	4.58e+7
mean	1.16e-02	-1.59e-02	1.17e+02	5.00e+07	1.06e-02	2862.00	3.33e+03
std	5.12e+01	6.22e+01	4.86e+01	2.89e+07	1.02e-01	1667.61	1.92e+03
min	-9.46e+01	-1.15e+02	3.77e+01	0.00e+00	0.00e+00	0.00	0.00e+00
25%	-4.50e+01	-5.79e+01	7.40e+01	2.50e+07	0.00e+00	1392.25	1.66e+03
50%	1.30e+00	-4.18e+00	1.21e+02	5.00e+07	0.00e+00	2887.00	3.33000e+03
75%	4.04e+01	4.85e+01	1.60e+02	7.50e+07	0.00e+00	4304.75	5.00000e+03
max	9.62e+01	1.05e+02	1.96e+02	1.01e+08	1.00e+00	5734.00	6.77e+03

no event hits and few timeslices contain a high number of event hits.

Figure 4.5 depicts a scatter plot of *timeslice 615* which contains the largest number of event hits. It is observed that event hits occur close to each other in space and time (represented by the yellow, blue and green points) whilst background hits are uniformly distributed in space and time (represented by the purple points).

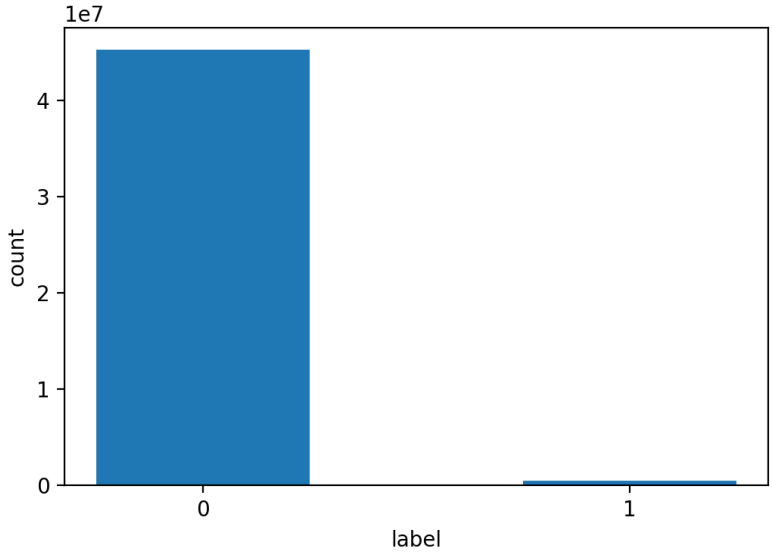


Figure 4.2: Distribution of `label` column

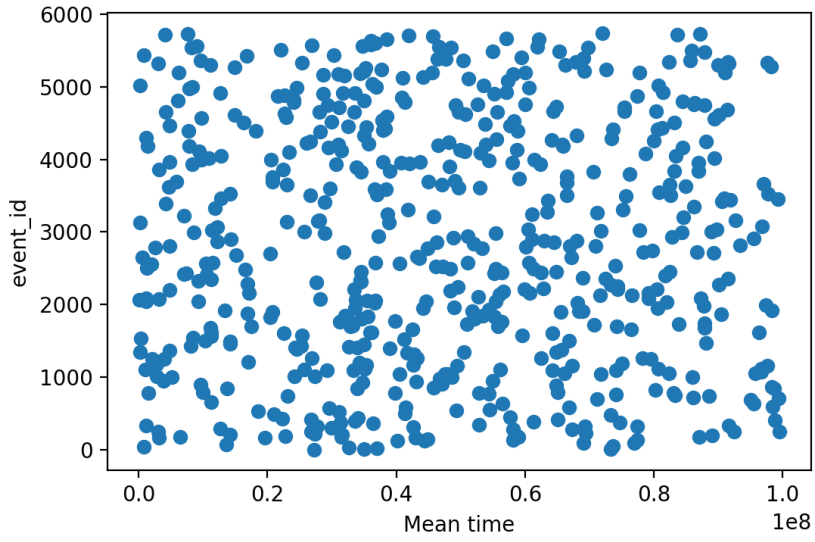


Figure 4.3: Verification of Bias

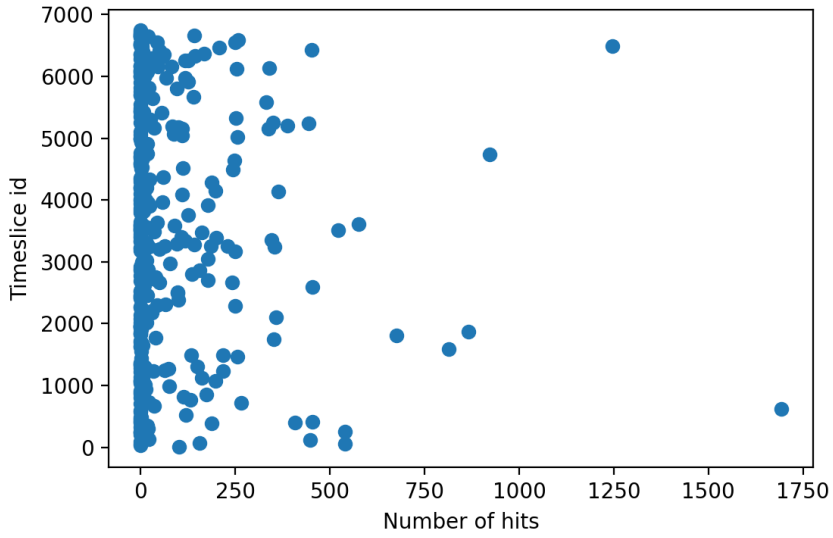


Figure 4.4: Distribution of event hits per timeslice

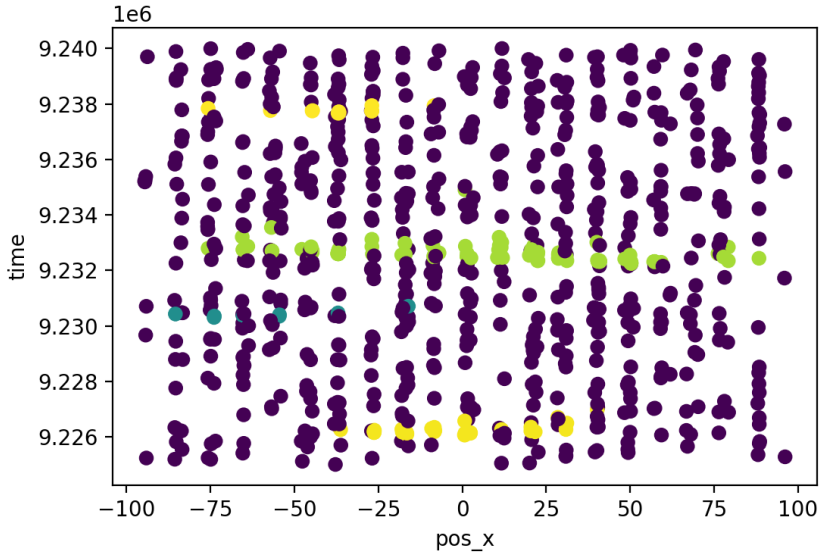


Figure 4.5: Distribution of Timeslice 615

5

Replacement for Hit Correlation Step

This chapter presents the replacement created using a Multi Layered Perceptron (MLP) for the *Hit Correlation Step* of the Karas pipeline (see **??**). It is observed that a MLP is able to identify causally related hits with a higher accuracy, precision and recall compared to the Pattern Matrix Criterion. The chapter begins by explaining how the data is created followed by its visual examination. The training and testing procedure for the model is explained next. The chapter concludes with discussions of the experiment results and next steps.

5.1 Data Preparation

A random sample was taken from the top 5 timeslices with the most number of event hits for the creation of the pattern matrix dataset, since these timeslices contain the highest number of event hits compared to other timeslices. This is useful since the dataset is highly skewed and using a timeslice with lesser number of event hits will proliferate the skewed nature of the data in the pattern matrix dataset once created. The model is required to classify related and unrelated hits which can be done by observing the space and time difference between the given points. Since this phenomenon is consistent across the entire dataset, training using a sample does not introduce any bias into the model.

Figure **??** summarizes the pattern matrix dataset creation process. With an input data of shape `(n, 4)` (`n` rows and 4 columns representing the `x, y, z, and time`), an output data of shape $(\sum_{k=n-1}^{1} k,\ 9)$ is obtained. A significant rise in the number of rows is observed since each row (representing a single hit) is paired with the remaining unique rows. The output dataset consists of 9 columns due to the presence of `x, y, z and t` columns of two hits plus the label column.

The label column is populated based on the values of the `event_id` column of the two hits. The row is assigned a label of 1 if the two hits have the same event id, which signifies that they originated from the same neutrino event and hence are causally related to each other. If the event id of the two hits are not the same then they are assigned a label of 0.

Better model performance was observed when the model was trained with the difference between hits in time and space. As a result, the pattern matrix was modified such that a dataset of shape $\sum_{k=n-1}^{1} k$, `5` was obtained. The fist 4 column being the difference of the `x, y, z and t` columns and the last column being the label.

5.1.1 Preparation of Training Data

The main dataset is highly skewed, with the **majority class** being hits from background noise and the **minority class** being hits from neutrino events. Thus, the *pattern matrix* dataset is also skewed with the minority class being related hits and majority class being unrelated hits. In binary classification, the majority class is also referred to as the **negative class** (since it usually has a label of 0) and the minority class is referred to as the **positive class** (since usually has a label of 1). Henceforth this alternative naming convention is used in this report.

Figure **??** shows the distribution of the two classes in the pattern matrix dataset. As observed, a several imbalance between the two classes exist. Training the model with such a skewed dataset will result in a model that is biased to the majority class. To combat this problem, the majority class is undersampled (**?**) such that the number of examples for each class is the same.

5.1.2 Preparation of Testing Data

Whilst the training dataset contains equal number of examples for each class, the testing dataset maintains it's skewed distribution since this represents realistic data which the model will be required to classify. Four variants of the testing dataset with varying level of examples of related hits were created as listed below.

In practise, the pipeline will observe timeslices which contain no to very few related hits, thus the performance of the model on test set 1 and 2 are of vital importance.

1. Test set containing no related hits
2. Test set containing less than 25 related hits
3. Test set containing less than 500 related hits

4. Test set containing less than 1500 related hits

5.2 Data Exploration and Visualization

5.3 Model Description

The expectation of the model is to identify if two given points are causally related to each other or not. As revealed through data exploration in Chapter 4, hits originating from neutrino events occur close to each other in space and time. Thus, The expectation from the model is to learn this phenomenon by training over pairs of points and classify unseen data as related or unrelated.

Being a binary classification task, the *Binary Cross Entropy Loss (BCELoss)* was selected as the loss function since it has been established as the standard loss function for binary classification tasks (**?**). As the BCELoss function expects an input in the range of $[0, 1]$, the *Sigmoid* activation function was chosen for the output layer. The *ReLu* activation was chosen for the hidden layers due to its XYZ properties as supported by many literate (**?**). The model architecture consists of an input layer, two hidden layers and an output layer. Figure **??** summarizes the model parameters and architecture. The network is fully connected with 4 neurons in the input layer, 16 neurons in the first hidden layer, 8 in the second hidden layer and finally 1 neuron in the output layer. The Adam optimizer with a learning rate of 0.001 is used to optimize the loss function.

The optimal value of all parameters stated above were identified empirically. The number of epochs used to train the model varied per experiment. This is because, this parameter is largely determined by the dataset itself and the learning rate of the optimizer.

5.4 Model Evaluation

The model is evaluated using several metrics which are regarded as the standard set of metrics used by deep learning practitioners to evaluate any machine learning model.

1. **Accuracy**. The accuracy is the ability of a model to classify unseen data correctly. Mathematically it can be defined as the number of correct predictions divided by the total number of examples in the test set.

2. **Loss Curve**. The loss curve is a line plot of the loss over the training epochs. A model with a good fit results in a loss curve which approaches 0 with time.

5.4.1 Additional Evaluation Metrics for Highly Skewed Data

For highly skewed data, accuracy is not a good metric for evaluating the model performance (8) thus the following alternatives are also considered for the evaluation of the model.

1. **Recall** is the ability of the model to correctly identify the minority class. For this problem, the recall of the model is given precedence over its precision. This is because the model should be able to identify all instances of the positive class since this determines if the timeslice will ultimately be saved or not.

2. **Precision** is the ability of the model to not misclassify an instance of the negative class (ie. classify it as the positive class). Although this should also be high, it is often inversely proportional to recall.

3. **F1 score** is the harmonic mean of the precision and recall. The F1 score is a value between [0, 1] with a value close to 1 indicating high precision and recall.

4. **F2 score** since recall is given precedence for this problem, the F2 score can be considered a better alternative to the F1 score as it gives higher importance to the recall through the β parameter.

5. **Receiver Operating Characteristic (ROC) curve** is a plot of the false positive rate and the false negative rate across various probability thresholds. The area under the ROC curve (ROC ARC) is also considered.

6. **Precision-Recall (CPR) Curve** can be considered a better alternative to the ROC curve since the ROC curve can be overly optimistic of the model's skill for skewed data. The Area under the PR curve (PR AUC) is also considered.

7. **Confusion Matrix** is used to visualize the true positive, true negative, false positive and false negative predictions of the model.

5.5 Discussion

6

Replacement for Graph Community Detection Step

This chapter presents the replacement created using a Graph Convolusional Neural Network (GCN) for the *Graph Community Detection Step* of the Karas pipeline (see **??**). It is observed that a GCN is able to classify noise and event nodes very well, even in extremely skewed datasets with less than 10 event nodes. The chapter begins with an overview of GCNs and how they have been applied to this problem. The data preparation, visualization and model evaluation are touched upon next. The chapter concludes with discussion of the results and next steps.

6.1 Primer on Graph Convolusional Neural Networks

6.1.1 GNN and graphs

GNNs are designed to operate on data which can be represented as graphs. A graph consists of nodes and edges. Each node may or may not be connected to one or many nodes, these are referred to as the neighbors of the node. A graph with all nodes connected to one another is called a fully connected graph.

An edge may attributes associated with it, the two most common attributes being weight and direction. An edge may be directed which denotes a sense of hierarchy amongst the nodes, or undirected. An edge may also have a weight to signify a stronger or weaker connection amongst nodes.

Graphs are primarily classified into two variants.

1. **Homogeneous graphs**. Graphs with the same type of nodes and edges are referred to as homogeneous graphs. For example, a graph representing who follows who on

Twitter is a homogeneous graph. Here, people are represented as nodes and an edge indicates that person A follows person B.

2. **Heterogeneous graphs**. Graphs with different types of nodes and edges are referred to as heterogeneous graphs. For example, a graph representing a person's likes and dislikes in regards to food items. Here, two entities, namely people and food are represented as nodes. The edges also come in two variants ie. a 'like' and a 'dislike'.

6.1.2 The Message Passing Paradigm

During each training epoch, a node propagates it's embedding to it's neighbors and in return receives their embedding. All collected embeddings are aggregated (for example using a sum, difference or mean) which becomes the new embedding of the node. This procedure is done for all nodes of the graph, for each training epoch. The number of layers in the network determine how far the messages are sent. For example, for a network with a single layer, each node sends a message to it's immediate neighbors. With 2 layers, the node also sends a message to the neighbors of it's immediate neighbors and so forth.

6.1.3 GNN and it's Variants

GNNs have two primary applications.

1. **Node classification**. This is a semi-supervised learning setting (although it can also be used in a supervised setting). Given a graph with nodes associated with a label, the network can be used to predict the label of unseen nodes.

2. **Graph classification**. In this approach, the network is trained using several graphs each associated with a label. The network can then be used to predict the label of an unseen graph.

6.2 Data Preparation

The graphs for the testing and training of the network are constructed from a combination of the main dataset and a modified version of the pattern matrix dataset.

The node embeddings and it's labels are derived from the main dataset. Each node is thus assigned a `(x,y,z,t)` vector as it's node embedding. The node is assigned a label of 1 if it is an event hit, else a label of 0.

A modified pattern matrix dataset (see 5.1) with a shape of $(n^2 - n, 5)$ is created such that each hit is paired with all other hits except itself. The label column from this dataset is then used as the edge weights of the graph. Edges between event nodes from the same event thus are assigned a weight of 1 and all other edges are assigned a weight of 0.

6.2.1 Preparation of Training Data

Since the main dataset and the pattern matrix dataset are highly skewed, naturally the gcd dataset is also skewed with majority of the nodes being noise. Similar strategy as used in the creation of the pattern matrix training set (see 5.1.1) is used. The training set is a graph with 1000 nodes equally distributed amongst the classes.

6.2.2 Preparation of Testing Data

The skewed nature of the data is maintained in the testing set. The model is evaluated with 3 test sets each with varying levels of examples of event nodes. In practise, the pipeline will observe timeslices with no to very few events thus the performance of the model on test set 1 and 2 should be given importance.

1. Test set with no event nodes
2. Test set with less than 25 event nodes
3. Test set with less than 250 event nodes

6.3 Data Exploration and Visualization

6.4 Model Description and Evaluation

The model is expected to classify nodes of an unseen graph as event or noise nodes. Since causally related nodes are connected with edges carrying a high weight, the model is expected to group them together thus resulting in a final graph with small clusters of causally related nodes and a large number of unclustered noise nodes.

The parameters of the model are summarized in Table **??**, the rational for selecting the parameters being the same as that of the pattern matrix model (see **??**) since both models perform binary classification. The difference comes from the model architecture. The GCD model comprises of an input layer, two hidden layers and an output layer. The network is

fully connected with 4 neurons in the input layer, 16 in both hidden layers and 1 neuron in the output layer.

Evaluation metrics used for evaluating the pattern matrix model are used to evaluate the GCD model as well (see Section 5.4) since the GCD dataset is also highly skewed in nature.

6.5 Discussion

7

Recommendations

This chapter presents some practical recommendations for the readers who wish to use the new data processing pipeline presented in this report. The chapter also presents alternative paths of research which remain unexplored and general improvements that can be made to the pipeline in the future.

Chapter 5 presented a Multi Layered Perceptron capable of identifying causally related hits with a higher accuracy, precision and recall compared to the Pattern Matrix Criterion presented by **(author?)**. The PM model is seem as a viable successor to the PM Criterion. The model can be further improved using larger training sets and the model performance can be improved by utilizing the plethora of techniques.

The output of the MLP is observed to be primitive (see Section **??**). As proved empirically, the GCD model is able to perform better when the edge weights are assigned based on the type of connection between the nodes (see Section **??**). In order to obtain the advanced edge weight scheme, the MLP must be modified such that it performs multi-class classification on the edge types. For a classification problem of n edge types, the output of the MLP will thus become a `(n,)` vector containing the expected probability for each class. These probabilities can then be used as the edge weights.

Chapter 6 presented a Graph Convolusional Neural Network capable of identifying event nodes with immaculate accuracy, precision and recall in extremely skewed datasets. The network however have a very high false positive rate when tested with a set containing no event nodes. This bias may be removed by framing the data as a multi-edge, heterogeneous graph. The model is thrown off by the high weights on edges between causally related event nodes and noise nodes. Thus by creating different types of edges corresponding to the various types of connections that two nodes may posses (see **??**), each carrying the corresponding weights, the network may be able to correct it's bias to the positive class.

7. RECOMMENDATIONS

The models presented in this report were testing in isolation. However the intended use is to combine them in order to identify timeslices containing neutrino event hits (see Section **??**). Thus the models should be tested as an integrated pipeline in order to determine if feasible as improvement over the Karas pipeline.

Results from the GCD experiments (see Section **??**) indicate that the model is unable to learn anything from the node embeddings thus identifying better node embeddings to help the enhance the model's learning abilities remains open to be examined. The network currently aggregates the node embeddings by adding them. This however may not be an apt aggregation function and the model's performance needs to be compared with alternatives such as the difference and mean.

An alternative path of research exists to explore the possibility of replacing the entire pipeline with a single GNN. The data can be framed such that the `(x, y, z)` vector is used as the node embedding and δt between the nodes is used as the edge feature.

Finally, the problem can also be framed as a graph classification problem. The expectation being that the network is able to identify clusters of causally related nodes from noise.

Bibliography

[1] Silvia Adrian-Martinez, M Ageron, F Aharonian, S Aiello, A Albert, F Ameli, E Anassontzis, M Andre, G Androulakis, M Anghinolfi, et al. **Letter of intent for KM3NeT 2.0**. *Journal of Physics G: Nuclear and Particle Physics*, **43**(8):084001, 2016. 1, 2

[2] Sebastiano Aiello, Fabrizio Ameli, Annarita Margiotta, Michel Andre, Giorgos Androulakis, Marco Anghinolfi, Antonio Marinelli, Gisela Anton, Miquel Ardid, Christos Markou, et al. **KM3NeT front-end and readout electronics system: hardware, firmware, and software**. *Journal of Astronomical Telescopes, Instruments, and Systems*, **5**(4):046001, 2019. 1, 2

[3] Annarita Margiotta, KM3NeT Collaboration, et al. **The KM3NeT deep-sea neutrino telescope**. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **766**:83–87, 2014. 1

[4] Maarten Post. *"KM3NNeT" A neural network for triggering and classifying raw KM3NeT data*. PhD thesis, Universiteit van Amsterdam, 2019. 2

[5] Konrad Karaś. *Data processing pipeline for the KM3NeT neutrino telescope*. PhD thesis, Universiteit van Amsterdam, 2019. 2, 3, 5, 23

[6] **NumPy - The fundamental package for scientific computing with Python**. 7

[7] **Pandas**. 8

[8] Paula Branco, Luis Torgo, and Rita Ribeiro. **A survey of predictive modelling under imbalanced distributions**. *arXiv preprint arXiv:1505.01658*, 2015. 18