

Vrije Universiteit Amsterdam



Universiteit van Amsterdam



Master Thesis

KM3NeT Neutrino Detection using Deep Learning

Author: Arumoy Shome (2636393)

1st supervisor: Dr. Adam Belloum

2nd supervisor: Ben van Werkhoven

(Netherlands eScience Center)

2nd reader: Dr. FINDME

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

October 11, 2020

Abstract

Here goes the abstract of this thesis.

Acknowledgements

I would like to thank Dr. Adam Belloum and Dr. Ben van Werkhoven for introducing me to the fascinating field of neutrino detection and giving me the opportunity to be a part of the KM3NeT research initiative in collaboration with Nikhef and The Netherlands eScience Center. Many thanks to Dr. Roel Aaij, Dr. Ronald Bruijn and Brian O Fearraigh for their technical support on the physics side of things. Finally my gratitude and best wishes to my colleague Shruti Rao who is also defending her thesis in the coming months.

Contents

List of Figures	v
List of Tables	ix
1 Introduction	1
1.1 Situation of Concern	2
1.2 User Requirements	3
1.3 Research Question	3
2 The Data	5
3 The Karas Pipeline	9
3.0.1 Limitations of the Karas Pipeline	10
4 Replacement for Hit Correlation Step	13
4.1 Data Preparation	13
4.1.1 Preparation of Training Data	14
4.1.2 Preparation of Testing Data	15
4.2 Model Description	15
4.3 Model Evaluation	16
4.4 Results	19
5 Replacement for Graph Community Detection Step	23
5.1 Primer on Graph Convolutional Neural Networks	23
5.2 Data Preparation	24
5.3 Model Description and Evaluation	25
5.4 Results	26
6 Recommendations	31

CONTENTS

Bibliography	35
---------------------	-----------

List of Figures

1.1	Artist's impression of the ARCA detector <i>source: https://www.km3net.org</i>	2
1.2	An optical detector (DOM) <i>source: https://www.km3net.org</i>	2
2.1	Correlation matrix of features	7
2.2	Distribution of <code>label</code> column	7
2.3	Verification of Bias	8
2.4	Distribution of Timeslice 615	8
2.5	Distribution of event hits per timeslice	8
3.1	Overview of Karas Pipeline	9
3.2	(1). The main dataset of the project, each row representing a hit consisting of the (x, y, z, t) vector. Here an example set containing 5 hits is shown. (2). The input to the Correlation Step is all unique pairs of hits. (3). The output of the Correlation Step is a vector containing the probability that the pairs of hits are related to each other. (4). (1) and (3) are used to construct a graph representation of the dataset where event (red, orange and blue) and noise (grey) hits are represented as nodes and related nodes are connected by an undirected edge carrying as weight the probability of being related. (5). The graph from (4) acts as the input to the Graph Community Detection step, the output of which is another graph where event and noise nodes are grouped together and thus are linearly separable from each other.	12

LIST OF FIGURES

4.1	Overview of MLP dataset creation procedure. (1) The main dataset where each row represents a hit, here an example set containing 5 rows is shown for simplicity. (2) The MLP dataset is generated from the main dataset consisting of all unique pairs of hits. Algorithmically, this is done by pairing each hit with the subsequent hits below it as demonstrated with the use of colors. (3) The difference between the hits is taken (dark orange) and a label is assigned to each row (gray).	14
4.2	Distribution of MLP training dataset.	14
4.3	Learning Curve of MLP Training and Validation Datasets.	19
4.4	ROC Curve for TS2.	20
4.5	ROC Curve for TS3.	20
4.6	ROC Curve for TS4.	20
4.7	ROC Curves for MLP Test Datasets.	20
4.8	PR Curve for TS2.	20
4.9	PR Curve for TS3.	20
4.10	PR Curve for TS4.	20
4.11	PR Curves for MLP Test Datasets.	20
4.12	Confusion Matrix for TS1.	21
4.13	Confusion Matrix for TS2.	21
4.14	Confusion Matrix for TS3.	21
4.15	Confusion Matrix for TS4.	21
4.16	Confusion Matrices of MLP Test Datasets.	21
5.1	Overview of GCN dataset creation procedure. (1) an example of the main dataset with 5 hits. (2) the corresponding modified MLP dataset created. Although the dataset will contain $n^2 - n = 25$ rows, only the unique rows (10 in this example) are shown in the illustration for simplicity. (3) The graph representation is created where rows of (1) become the node embeddings and the label column of (2) become the edge weights.	25
5.2	Learning Curve for GCN.	27
5.3	TSNE for training set before training.	28
5.4	TSNE for training set after training.	28
5.5	TSNE for GCN training dataset.	28
5.6	CM for TS1.	28
5.7	CM for TS2.	28

LIST OF FIGURES

5.8	CM for TS3.	28
5.9	CM for GCN Test Datasets.	28
5.10	TSNE for TS1 before.	29
5.11	TSNE for TS2 before.	29
5.12	TSNE for TS3 before.	29
5.13	TSNE for TS1 after.	29
5.14	TSNE for TS2 after.	29
5.15	TSNE for TS3 after.	29
5.16	TSNE for GCN Test Datasets with naive edge weights.	29
6.1	TSNE for TS1 with advanced edge weights.	32
6.2	TSNE for TS2 with advanced edge weights.	32
6.3	TSNE for TS3 advanced edge weights.	32
6.4	TSNE for GCN test datasets with advanced edge weights.	32
6.5	CM for TS1 (advanced edge weights).	33
6.6	CM for TS2 (advanced edge weights).	33
6.7	CM for TS3 (advanced edge weights).	33
6.8	CM for GCN test datasets with advanced edge weights.	33

LIST OF FIGURES

List of Tables

2.1	Description of columns	6
2.2	Descriptive statistics	7
4.1	Distribution of MLP Training and Validation Datasets.	15
4.2	Distribution of MLP Test Datasets.	15
4.3	GCN Model Parameter Summary.	16
4.4	MLP Model Architecture Summary.	16
4.5	Summary of MLP performance across test sets.	19
5.1	Distribution of GCN testing datasets.	26
5.2	GCN Model Parameter Summary.	26
5.3	GCN model architecture summary.	26
5.4	Summary of GCN performance across test sets.	27
6.1	Advanced edge weight scheme for GCN.	31
6.2	Summary of GCN performance across test sets with advanced edge weights.	33

LIST OF TABLES

1

Introduction

Perhaps the most elusive subatomic particle known to science is the Neutrino. With no electrical charge (thus aptly named) and mass smaller than any other elementary particles, neutrinos simply pass through other matter making them virtually undetectable. However, neutrinos are sought after by researchers especially the ones working in the fields of Astrophysics and Astronomy since they may help us gain insights into astronomical events such as the birth of a neutrino star or a supernova.

When neutrinos experience a change in the density of the matter they are passing through(such as going from air into water), they experience a change in velocity thus emitting an electron and a photon. This phenomenon is known as Cherenkov Radiation (1) and is an indirect method which can be used to detect neutrinos. In fact, this remains the premise for some of the worlds largest neutrino observatories built to date such as the Sudbury Neutrino Observatory (Ontario, Canada), The Super-Kamiokande (Gifu Prefecture, Japan) and The IceCube Neutrino Observatory (Antarctica).

The KM3NeT or the Cubic Kilometer Neutrino Telescope is the next generation neutrino telescope, currently being constructed at the bottom of the Mediterranean Sea. The goal of this research infrastructure is two fold. First, is to study high energy neutrinos originating from celestial events in the galaxy. And second, to study the properties of the neutrino particles produced in the Earth's atmosphere (2). 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) (2). In this paper, we talk exclusively about KM3NeT/ARCA.

1. INTRODUCTION

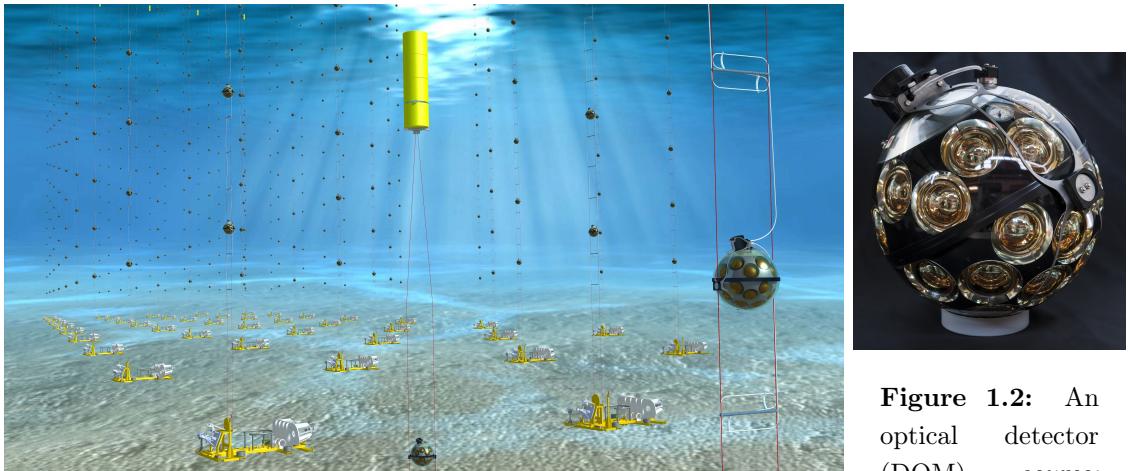


Figure 1.1: Artist’s impression of the ARCA detector *source:* <https://www.km3net.org>

Figure 1.2: An optical detector (DOM) *source:* <https://www.km3net.org>

1.1 Situation of Concern

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 1.1 shows an artist’s impression of ARCA, figure 1.2 depicts a DOM along with the PMTs inside it. The PMTs are highly sensitive to light (photons), and thus are used to detect the Cherenkov Radiation of neutrino particles. 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 (3).

Unfortunately, there are several sources of noise (in this case, the noise is other light sources), bioluminescence and decay of Potassium 40 (^{40}K) and atmospheric Muons being the primary sources (4). Due to the high level of noise, data is generated at an extremely high rate of 25GB/sec (2) and must be filtered and selectively stored for further analysis. The state of the art for this task are known as “Event Trigger” algorithms (2, 3) which can filter timeslices containing just noise thus only allowing important timeslices containing neutrino hits to pass through. The existing event trigger algorithms namely $L1$ and $L2$ although able to conduct the filtration in near real, lack the ability to do so with high accuracy thus often failing to save important timeslices (5). Efforts have already been made to improve the existing event trigger algorithms. Karas et al. proposed and implemented a GPU powered pipeline which utilizes correlation and graph community detection to

1.2 User Requirements

identify time slices that may contain neutrino hits whilst (4) suggests an alternate using convolutional neural networks.

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 timeslices containing information from neutrino events can lead to loss of important data and thus a poor quality of research. Since majority of the data generated is noise, the pipeline must be able to prevent storage of unnecessary timeslices containing only noise in the on-shore facility.

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.

1.3 Research Question

This report intends to improve upon *The Karas Pipeline*, a GPU pipeline proposed by Karas et al. to combat the limitations of the L1 and L2 event trigger algorithms. Specifically this project sought 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 and 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.

1. INTRODUCTION

RQ2. Can the *Hit Correlation Step* be replaced with a Multi Layer Perceptron?

The first step of The Karas Pipeline is the Hit Correlation Step, 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 Convolutional Neural Network?

The output of the Hit Correlation Step is used to create a graph structure where hits (of neutrino 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 neutrino and noise hits. The second phase of this project focuses on achieving a better accuracy for clustering neutrino and noise hits into separate communities in a given timeslice using Graph Convolutional Neural Networks (GCNs).

The following chapters of this report present the research efforts carried out to improve The Karas Pipeline using Artificial Neural Networks. In Chapter ?? the steps taken to prepare the dataset used in this project is presented followed by its statistical analysis and visual exploration in Chapter ???. Since this research initiative is a direct offshoot of the work previously conducted by Karas et al., an overview of The Karas Pipeline is presented in Chapter 3. Chapters 4 and 5 present detailed analysis of the neural networks created to replace segments of The Karas Pipeline. Drawing from the results of the replacement models, practical recommendations and directions for further research are laid out in Chapter 6. This report is intended for the primary users of ARCA with the hope to aid in the development of the successor to the state of the art event trigger algorithms. The report may also be used by deep learning practitioners working in the field of neutrino detection. This report assumes the reader posses a background in Computer Science or Artificial Intelligence and thus is familiar with concepts such as statistics, linear algebra and optimization. The reader is expected to have basic understanding of the guiding principals of Deep Learning such as Feed-Forward Neural Networks, backpropagation, binary classification and model evaluation metrics.

2

The Data

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

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 as rows *time (t)*, *dom_id*, *pmt_id* and *time over threshold (tot)*). The position coordinates (ie. *x*, *y* and *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.

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 were dropped along with columns which are

2. THE DATA

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.

The main dataset was explored using statistical analysis and visualizations to observe any patterns and "local trends" that may be present. 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.

Table 2.1: Description of columns

Column	Data type	Unit	Description
x, y, 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.
t	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.

Table 2.2 presents the descriptive statistics of the main dataset. The dataset consists of 7 columns (or features) and roughly 4.5 million rows, Table 2.1 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 the features are checked using the "Pearson" correlation and depicted by a correlation matrix in Figure 2.1. No significant correlations are observed between x , y , z and t 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 2.2. 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.



Figure 2.1: Correlation matrix of features

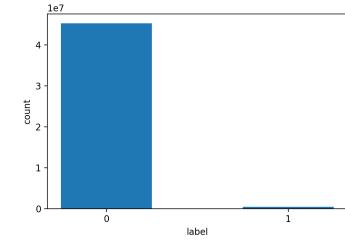


Figure 2.2: Distribution of label column

Table 2.2: Descriptive statistics

	x	y	z	t	label	event_id	timeslice
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

The main dataset is derived from synthetically generated data 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 as a scatter plot as depicted by Figure 2.3. A uniform distribution is noted with no visible patterns indicating no bias exists in the dataset and it is deemed suitable for further analysis.

The dataset is discretized into 6759 timeslices of which 2783 timeslices contain only noise hits. This is corroborated by Figure 2.5 which presents a skewed distribution where many

2. THE DATA

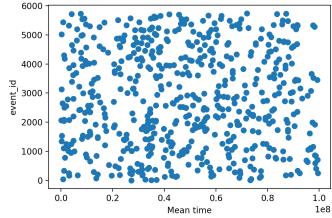


Figure 2.3: Verification of Bias

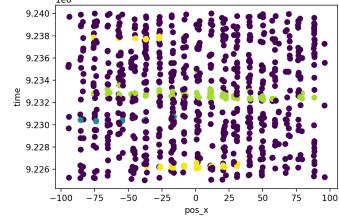


Figure 2.4: Distribution of Timeslice 615

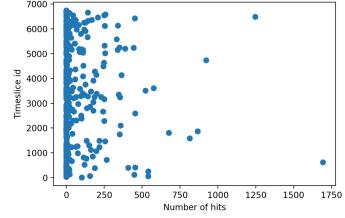


Figure 2.5: Distribution of event hits per timeslice

timeslices contain few to no event hits and few timeslices contain a high number of event hits. Figure 2.4 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).

3

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 ?? and ??).

Karas et al. 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, illustrated in Figure 3.1 and described in more detail below. Figure 3.2 provides a visual representation of the various shapes and arrangements of the data as it passes through the pipeline.

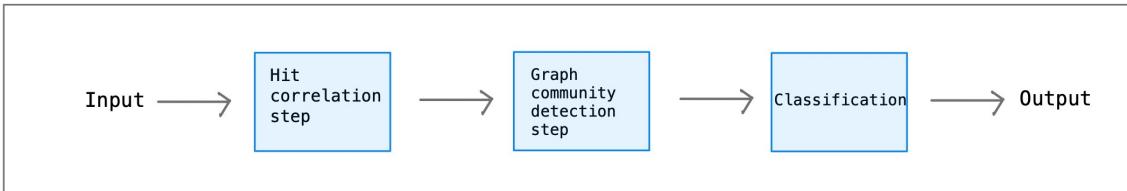


Figure 3.1: Overview of Karas Pipeline

The first step of The Karas pipeline is the Hit Correlation step which proposes *The Pattern Matrix Criterion (PMC)* to identify hits which may have originated from the same neutrino event, referred to as *causally related* hits. From domain knowledge, it is known that related hits occur close to each other both in space and time. Specifically, the space and time difference of related hits is found to be 100m and 300ns respectively. The PMC operates by creating a correlation criterion based on the probability that the aforementioned space and time difference occurs between two related hits. Pairs of hits (from both

3. THE KARAS PIPELINE

events and noise) are passed to the PMC as input, the output being an adjacency matrix where related hits are assigned a high score whilst unrelated hits are assigned a lower score. 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. Due to the stochastic nature of hits, unrelated hits such as noise-event or noise-noise pairs may occur within this predefined range thus being falsely identified as related. This is rectified in the next step of the pipeline.

The second step of the Karas pipeline is the Graph Community Detection (GCD) step. The main dataset is represented as a graph where hits are represented as nodes. All nodes are connected with each other and carry as weight the probability of being related using the output of the PMC. The *Constant Potts Model (CPM)* is used to group the nodes into separate communities (or clusters) of related and unrelated hits. The model is tested using a dataset consisting of 130 event hits and 5000 noise hits and performs exceptionally well as it is able to group most event hits into a single community and the noise in another.

Since the GCD step directly operates on the data derived using the Hit Correlation step, communities may still contain noise nodes. Thus the third and final step of the pipeline classifies given communities as *event* or *noise* communities based on the exclusive presence of event nodes. This can be done by observing two properties of the graph namely the size of the communities and the density of edges within the communities. Communities consisting exclusively of event nodes will be of small sizes and have high edge density whilst communities consisting a mix of event and noise nodes or only noise nodes will be relatively larger and have lesser edge density. The two parameters which aid in the classification are the Probability Threshold (PT) of the PMC and the CPM resolution parameter (γ) of the GCD step. The PT and γ are grid searched to determine their optimal thresholds, all hits above the specified thresholds are classified as event communities and the rest as noise communities.

3.0.1 Limitations of the Karas Pipeline

Although The Karas Pipeline is able to identify timeslices with neutrino event hits more accurately compared to its predecessors (5), the pipeline still has certain limitations which hinders its performance. 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 may result in related hits which do not meet these thresholds to be incorrectly given a low score. Due to the stochastic nature of hits, event-noise and noise-noise pairs which fall within the thresholds may also occur and these will be incorrectly given a high score. The communities created

created by CPM is biased by the edge weights provided by the PMC. Furthermore, 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.

Instead of deriving the correlation thresholds manually, a better approach may be to use NNs to learn the optimal thresholds. This idea is further explored in Chapter 4 where a MLP is used to classify related and unrelated hits. Recently, the field of Geometric Deep Learning has gained popularity which some very exciting developments in models which are designed to operate upon graph structures. Once such development is the Graph Convolutional Neural (GCN) Network proposed by Kipf et al. The limitations of the GCD and classification steps may be alleviated by using a GCN to classify event and noise nodes and is explored in Chapter 5.

3. THE KARAS PIPELINE

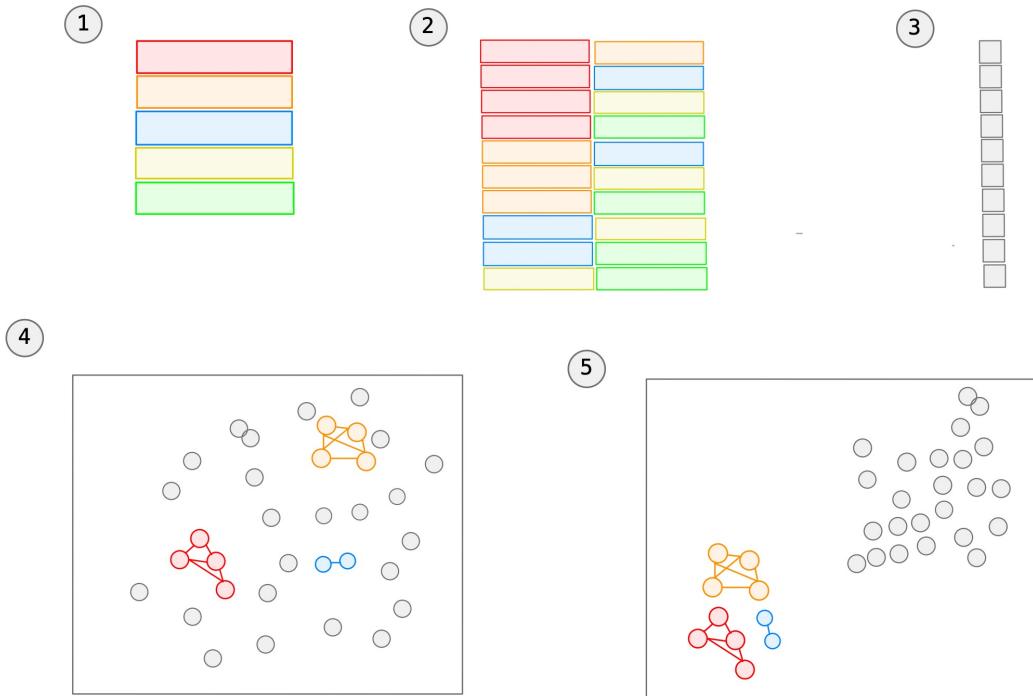


Figure 3.2: (1). The main dataset of the project, each row representing a hit consisting of the (x, y, z, t) vector. Here an example set containing 5 hits is shown. (2). The input to the Correlation Step is all unique pairs of hits. (3). The output of the Correlation Step is a vector containing the probability that the pairs of hits are related to each other. (4). (1) and (3) are used to construct a graph representation of the dataset where event (red, orange and blue) and noise (grey) hits are represented as nodes and related nodes are connected by an undirected edge carrying as weight the probability of being related. (5). The graph from (4) acts as the input to the Graph Community Detection step, the output of which is another graph where event and noise nodes are grouped together and thus are linearly separable from each other.

4

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 3). It is observed that a MLP is able to identify causally related hits with a higher accuracy, precision and recall compared to the PMC. 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.

4.1 Data Preparation

Figure 4.1 summarizes the MLP dataset creation process. With an input data of shape $(n, 4)$ (n rows and 4 columns representing $x, y, z, \text{ and } t$), 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 all rows that follow. 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 final dataset of shape $\sum_{k=n-1}^1 k, 5$ was obtained. The first 4 columns being the difference of x, y, z and t vectors of the paired hits and the last column being the label. The data is additionally scaled between [0, 1] as

4. REPLACEMENT FOR HIT CORRELATION STEP

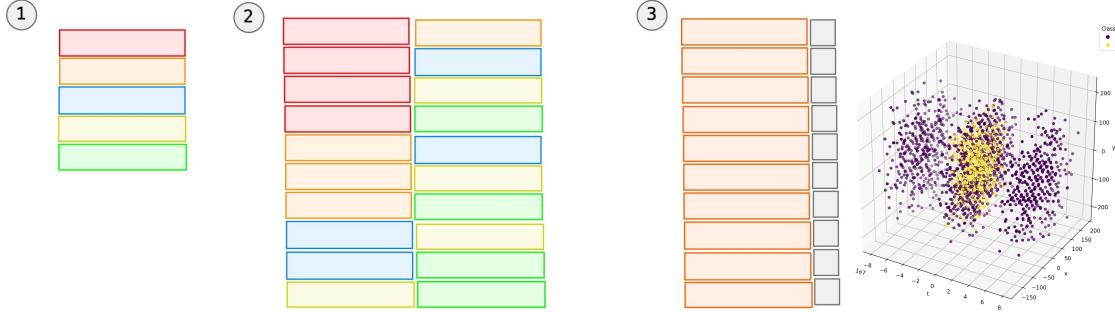


Figure 4.1: Overview of MLP dataset creation procedure. (1) The main dataset where each row represents a hit, here an example set containing 5 rows is shown for simplicity. (2) The MLP dataset is generated from the main dataset consisting of all unique pairs of hits. Algorithmically, this is done by pairing each hit with the subsequent hits below it as demonstrated with the use of colors. (3) The difference between the hits is taken (dark orange) and a label is assigned to each row (gray).

Figure 4.2: Distribution of MLP training dataset.

this is recommended and empirically proven to improve model performance and prevent vanishing gradients during training (8, 9).

4.1.1 Preparation of Training Data

The main dataset is highly skewed, with the **majority or negative class** being hits from background noise and the **minority or positive class** being hits from neutrino events. Thus, the training set created is also skewed with the minority class being related hits and majority class being unrelated hits. To maximize the number of positive examples in the training set, a random sample was taken from the top 5 timeslices of the main dataset with the most number of event hits. 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 main dataset, training using a sample does not introduce any bias into the model.

The training set still however contains a skewed distribution of examples, and training the model with such a 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. Figure 4.2 shows the distribution of a random sample of the training set. It is observed that the related hits occur close to one another in space and time whilst the noise hits are scattered throughout which should aid the model to learn.

4.2 Model Description

A fraction of the training set is kept as a hold out or validation set to evaluate the model's training. Table 4.1 presents the distribution of the training and validation sets.

Table 4.1: Distribution of MLP Training and Validation Datasets.

	Total examples	Positive examples	Negative examples
Training	48,434	24,217	24,217
Validation	23,856	11,928	11,928

4.1.2 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 in Table 4.2. In practice, the pipeline will observe timeslices which contain no to very few related hits, thus the performance of the model on TS1 and TS2 are of vital importance.

Table 4.2: Distribution of MLP Test Datasets.

	Total examples	Positive examples	Negative examples
TS1	774,390	—	774,390
TS2	5,829,405	10	5,829,395
TS3	5,880,735	1176	5,879,559
TS4	364,231	8,372	355,859

4.2 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 ??, 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.

The parameters of the model are summarized in Table ???. The Adam optimizer with a learning rate of 0.001 is used to optimize the loss function since it is considered a reasonable start for many optimization problems and has been empirically proven to outperform other algorithms such as *Stochastic Gradient Descent (SGD)*, *RMSProp* and *Adagrad* (10). Being

4. REPLACEMENT FOR HIT CORRELATION STEP

Loss	BCELoss
Optimizer	Adam
Learning rate	0.001
Hidden activation function	ReLU
Output activation function	Sigmoid
Training batch size	16
Testing batch size	32

Table 4.3: GCN Model Parameter Summary.

a binary classification task, the *Binary Cross Entropy Loss (BCELoss)* (also known as *Log Loss*) was selected as the loss function since it has been established as the standard loss function for binary classification tasks (11). 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 function was chosen for the hidden layers since it is generally considered a good start for most problems (12, 13). A batch size of 16 is used for the training and validation sets whilst a larger batch size of 32 is used for the testing set since these value are often recommended as good defaults by machine learning practitioners (8?).

Layer Position	Type	Activation	In features	Out features
1	Linear	ReLU	4	16
2	Linear	ReLU	16	8
3	Linear	Sigmoid	8	1

Table 4.4: MLP Model Architecture Summary.

The model architecture is summarized in Table 4.4. It consists of an input layer, two hidden layers and an output layer. 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 optimal value of all parameters stated above were identified either empirically or from recommendations presented in literature. The number of epochs used to train the model is not mentioned above since this parameter is largely determined by the dataset, batch size and the learning rate, thus its value varied per experiment.

4.3 Model Evaluation

The model is evaluated using several metrics which are regarded as the standard set of metrics to evaluate machine learning model. Additionally, metrics geared towards evaluating

4.3 Model Evaluation

a model performing binary classification on skewed datasets are considered as well.

1. **Accuracy.** The accuracy is the ability of a model to classify unseen data correctly and is mathematically defined by Equation 4.1.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of examples}} \quad (4.1)$$

2. **Learning Curve.** The Learning 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.
3. **Confusion Matrix (CM).** For highly skewed data, accuracy is not a good metric for evaluating the model performance (14) since it may achieve a high score by simply predicting the majority class. Thus the CM is used to visualize the number of *true positive (TP)*, *true negative (TN)*, *false positive (FP)* and *false negative (FN)* predictions of the model.
4. **Recall.** The 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. The recall is mathematically defined by Equation 4.2.

$$\frac{TP}{TP + FN} \quad (4.2)$$

5. **Precision.** The 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. The precision is mathematically defined by Equation 4.3.

$$\frac{TP}{TP + FP} \quad (4.3)$$

6. **F1 score.** The F1 score is the harmonic mean of the precision and recall, thus it is a single metric to summarize the model's performance based on its precision and recall. The F1 score is a value between [0, 1] with a value close to 1 indicating high precision and recall. The F1 score is mathematically defined by Equation 4.4.

4. REPLACEMENT FOR HIT CORRELATION STEP

$$\frac{2 * (Precision * Recall)}{Precision + Recall} \quad (4.4)$$

7. **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. The F2 score is mathematically defined by Equation 4.5. Thus the F2 score with $\beta = 1$ is equivalent to the F1 score.

$$\frac{(1 + \beta^2) * Precision * Recall}{(\beta^2) * (Precision + Recall)} \quad (4.5)$$

8. **Receiver Operating Characteristic (ROC) curve** is a plot of the *true positive rate (TPR)* and the *false positive rate (FPR)* (mathematically defined by Equations 4.6) across various discrimination probability thresholds of the model. The ROC curve can be interpreted as the fraction of correct predictions for the positive class (along the y-axis) versus the fraction of error in predictions for the negative class (along the x-axis). The area under the ROC curve (ROCAUC) can be used to summarize the ROC curve with a singular value. Thus, a highly skilled model has a ROC curve which arches from $(0, 0)$ to $(1, 1)$ with a ROCAUC between 0.5 and 1.0. Whilst the ROC curve of a model without any skill is a straight line from $(0, 0)$ to $(1, 1)$ with a ROCAUC of 0.5.

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN} \quad (4.6)$$

9. **Precision-Recall (PR) Curve** can be considered a better alternative to the ROC curve since the ROC curve can be overly optimistic of the model's skill when dealing with highly skewed data since it considers the model's performance for both the classes. In contrast, the PR curve is a diagnostic plot of the model's precision and recall across various discrimination probability thresholds of the model. Thus the PR curve only considers the model's performance with regards to the positive class. Similar to the ROC curve, the PR curve can also be summarized by the area under the curve or the PRAUC. A skilled model, thus has a PR curve which bows towards $(1, 1)$ whilst a model with no skill is a horizontal line.

4.4 Results

Figure 4.3 shows the learning curve of the model on the training and validation tests. The curves approach zero with time and remain in close proximity to each other indicating a good fit. Table 4.5 summarizes the model’s performance across the various test sets (see Section 4.1.2). In general, the model performs well with high accuracy and recall across all test sets. The model achieves almost perfect recall for TS2 and TS3 which have extremely low positive examples which speaks to its strength. The poor precision scores can be attributed to the skewed nature of the datasets. Since the FPs are relatively high compared to the TPs in the datasets, the precision falls drastically. This is also corroborated by the fact that the recall in TS4 increases due to a better FP:TP ratio (see Figure 4.16).

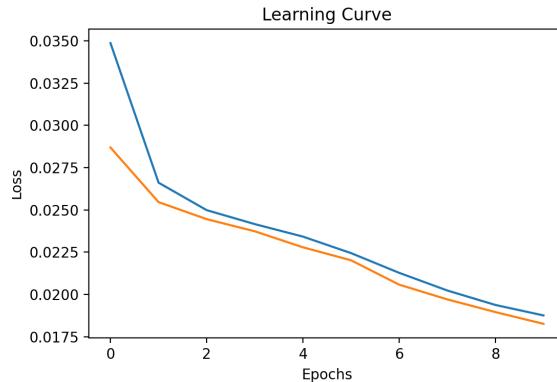


Figure 4.3: Learning Curve of MLP Training and Validation Datasets.

	Accuracy	Precision	Recall	F1	F2	ROCAUC	PRAUC
TS1	0.80	—	—	—	—	—	—
TS2	0.91	0.00	1.00	0.00	0.00	0.99	0.00
TS3	0.92	0.00	0.98	0.00	0.01	0.98	0.01
TS4	0.92	0.19	0.83	0.31	0.48	0.96	0.33

Table 4.5: Summary of MLP performance across test sets.

The ROC curves paint a different picture (see Figure 4.7) with exceedingly high areas. This however is misleading since ROC curve takes into account the performance for both the positive and negative classes and thus tend to be overly optimistic of the model’s performance on skewed datasets (14, 15). In such circumstances, The PR curves are considered

4. REPLACEMENT FOR HIT CORRELATION STEP

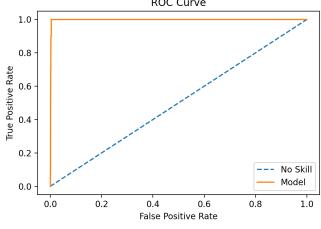


Figure 4.4: ROC Curve for TS2.

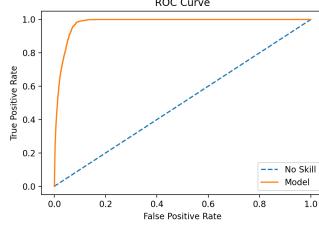


Figure 4.5: ROC Curve for TS3.

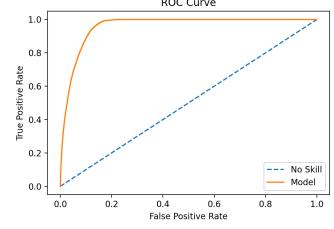


Figure 4.6: ROC Curve for TS4.

Figure 4.7: ROC Curves for MLP Test Datasets.

better at gauging a model's skill when dealing with skewed datasets (14). However, in this case it too is deemed misleading as it seriously undermines the model's performance (see Figure 4.11). This again is attributed to the precision approaching zero due to the extremely skewed FP:TP ratio which results in the PRAUC to also be zero.

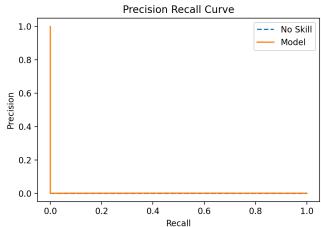


Figure 4.8: PR Curve for TS2.

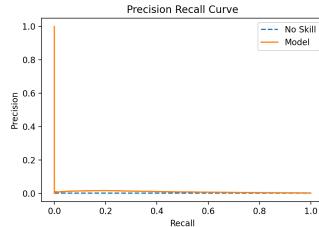


Figure 4.9: PR Curve for TS3.

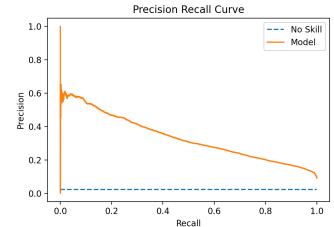


Figure 4.10: PR Curve for TS4.

Figure 4.11: PR Curves for MLP Test Datasets.

Figure 4.16 depict the confusion matrices for the various test datasets. The confusion matrices highlight the strengths and weaknesses of the model better than the ROC and PR curves. The model has a very low number of FNs and FPs which is extremely valuable as this ensures that the model does not miss hits from neutrino events and also does not classify hits from noise sources incorrectly as hits from neutrino events.

4.4 Results

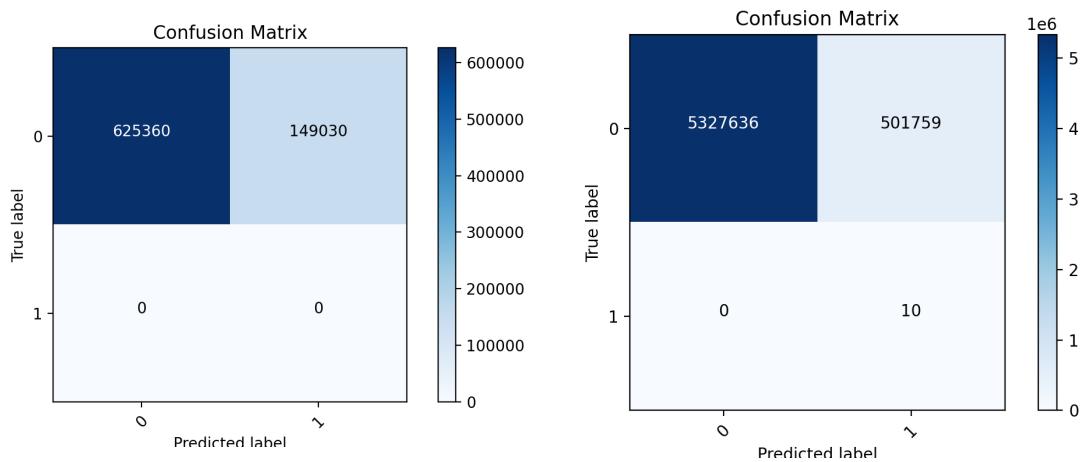


Figure 4.12: Confusion Matrix for TS1.

Figure 4.13: Confusion Matrix for TS2.

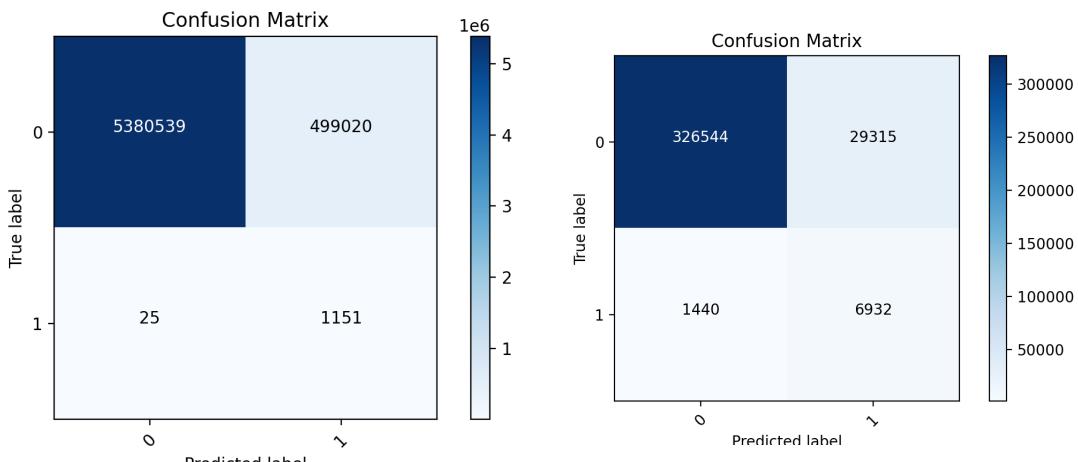


Figure 4.14: Confusion Matrix for TS3.

Figure 4.15: Confusion Matrix for TS4.

Figure 4.16: Confusion Matrices of MLP Test Datasets.

4. REPLACEMENT FOR HIT CORRELATION STEP

5

Replacement for Graph Community Detection Step

This chapter presents the replacement created using a Graph Convolutional Neural Network (GCN) as proposed by Kipf et al. for the *Graph Community Detection Step* of the Karas pipeline (see Chapter 3). It is observed that a GCN is able to identify event nodes very well however is severely biased to them thus unable to distinguish between event from noise nodes. The chapter begins with an overview of GCNs and how they have been applied to this problem. The data preparation and model evaluation are touched upon next. The chapter concludes with discussion of the results.

5.1 Primer on Graph Convolutional Neural Networks

GCNs are designed to operate on data consisting of entities and their relations, commonly referred to as *graphs*. A graph $G = (V, E)$ consists of a set *nodes* (V) and a set of *edges* (E). 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 have attributes associated with it, the two most common attributes being *weight* and *direction*. An edge $(u, v) \in E$ between two nodes u and v may be *directed* which denotes a sense of hierarchy amongst the nodes, an example being a graph which models how Twitter users follow one another. An edge may also be *undirected* such as a graph which models the friendship amongst the users of a social network (since friendship is mutual). An edge may also have a weight to signify a stronger or weaker connection amongst nodes. Nodes may also possess attributes associated with themselves, commonly known as *node embeddings*. The complexity of the node embedding may range

5. REPLACEMENT FOR GRAPH COMMUNITY DETECTION STEP

from a simple scalar quantity to a multi-dimensional tensor, and depends on the dataset and how the practitioner wishes to sculpt the graph.

Graphs are primarily classified into two variants namely *homogeneous* and *heterogeneous* graphs. Homogeneous graphs have the same type of entities and relations represented as nodes and edges respectively. For example, a graph representing the social network consisting of people and their connections is a homogeneous graph. In contrast, Heterogeneous graph consist of different types of nodes and edges. 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'.

GCNs learn by utilizing a message passing paradigm which is summarized in Figure ???. During each training epoch, all nodes propagate their embedding to their neighbors. The collected embeddings are then aggregated (for example using a sum, difference or mean) which becomes the new embedding for 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 aggregates the embeddings from their immediate neighbors. With 2 layers, the node also aggregates embeddings from the neighbors of it's immediate neighbors and so forth.

5.2 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 MLP dataset, Figure 5.1 illustrates this procedure. A fully connected graph is constructed, and its node embeddings and node 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 to denote noise.

A modified MLP dataset (see 4.1) 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.

Since the main dataset and the MLP dataset are highly skewed, naturally the GCN 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 4.1) is used. The training set is a graph with approximately 1000 nodes equally distributed amongst the classes. The skewed

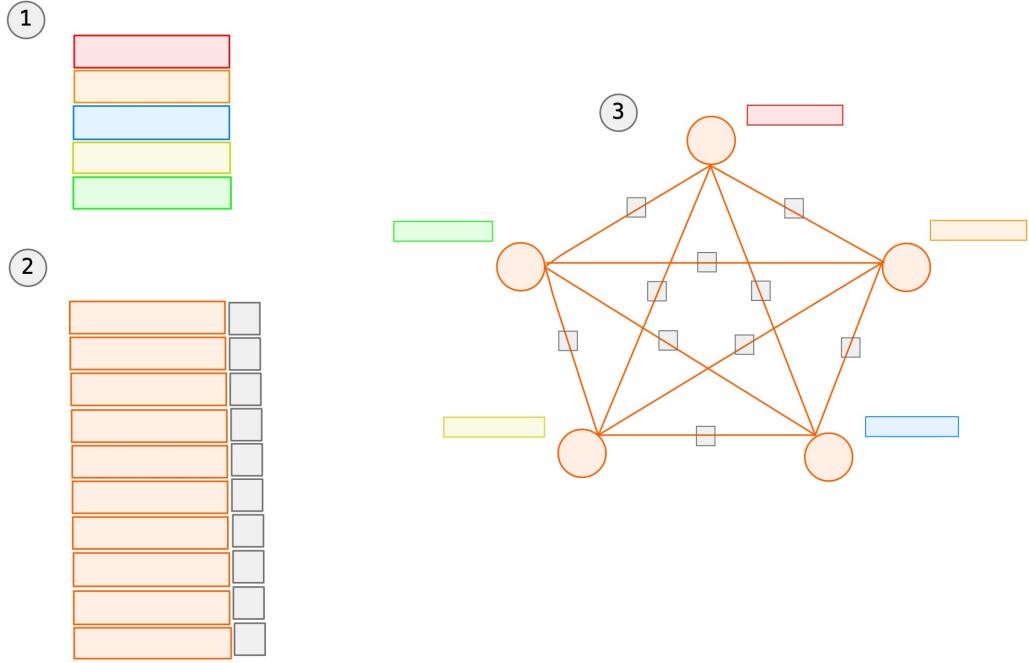


Figure 5.1: Overview of GCN dataset creation procedure. (1) an example of the main dataset with 5 hits. (2) the corresponding modified MLP dataset created. Although the dataset will contain $n^2 - n = 25$ rows, only the unique rows (10 in this example) are shown in the illustration for simplicity. (3) The graph representation is created where rows of (1) become the node embeddings and the label column of (2) become the edge weights.

nature of the data is maintained in the testing set. The model is evaluated with 3 test sets each with varying levels 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. The various test sets and their distribution are summarized in Table ??.

5.3 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 separate clusters of causally related nodes and noise nodes. The parameters of the model are summarized in Table 5.2, the rational for selecting the parameters being the same as that of the MLP

5. REPLACEMENT FOR GRAPH COMMUNITY DETECTION STEP

Table 5.1: Distribution of GCN testing datasets.

	Total examples	Positive examples	Negative examples
TS1	1000-1500	—	1000-1500
TS2	1000-1500	10-20	990-1480
TS3	1000-1500	200-250	800-1250
Loss			BCELoss
Optimizer	Adam with learning rate of 0.001		
Hidden Activation	ReLU		
Output Activation	Sigmoid		

Table 5.2: GCN Model Parameter Summary.

model (see 4.2) since both models perform binary classification. The difference comes from the model architecture which is summarized in Table 5.3. The GCN model comprises of an input layer, two graph convolutional layers and an output layer. The network is fully connected with 4 neurons in the input layer, 16 in both graph convolutional layers and 1 neuron in the output layer. A dropout layer is added between the two Gconv layers to prevent overfitting (16). Evaluation metrics are kept the same (see Section 4.3) since the GCN dataset is also highly skewed in nature.

5.4 Results

A good fit is achieved by the model during training as seen in Figure 5.2. In addition to the model’s performance on the various test sets as summarized by Table 5.4, the node embedding of the training and testing graphs are also inspected using t-SNE (17). Figure 5.5 shows the node embedding of the training before and after training. It is interesting to note that the model is indeed able to learn as clusters of similar nodes are noticed after training.

Table 5.3: GCN model architecture summary.

Layer position	Type	Activation	In features	Out features
1	GConv	ReLU	4	16
2	dropout	—	—	—
3	GConv	ReLU	16	2
4	Linear	Sigmoid	2	1

5.4 Results

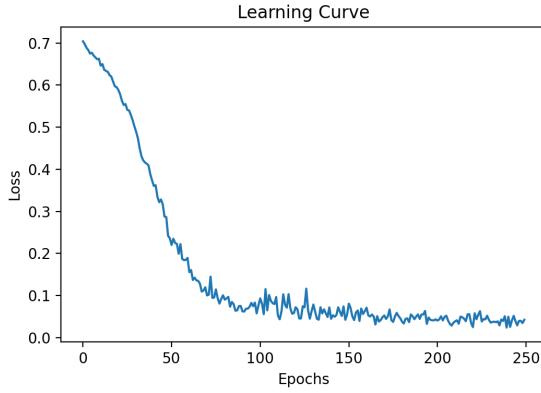


Figure 5.2: Learning Curve for GCN.

Table 5.4: Summary of GCN performance across test sets.

	Accuracy	Precision	Recall	F1	F2	ROCAUC	PRAUC
TS1	TODO	—	—	—	—	—	—
TS2	0.58	0.04	1.00	0.08	0.18	0.87	0.06
TS3	0.67	0.40	1.00	0.57	0.77	0.81	0.36

Inspecting Figure 5.16 one can see the model's inability to cluster similar nodes in the testing sets. Very scarce communities are observed in TS1 and TS2 with the model only being able to cluster the event nodes in TS3. In fact, the model is biased to the minority class as is seen in the Confusion Matrices depicted in Figure ???. The model is able to identify all event nodes in TS2 and TS3 perfectly however has a high number of FPs in all 3 test sets. This indicates that the presence of the high edge weights assigned to causally related nodes greatly aids the model in identifying the event nodes. However, since all other edges are assigned a weight of 0, the model does not quite learn to identify the noise nodes.

5. REPLACEMENT FOR GRAPH COMMUNITY DETECTION STEP

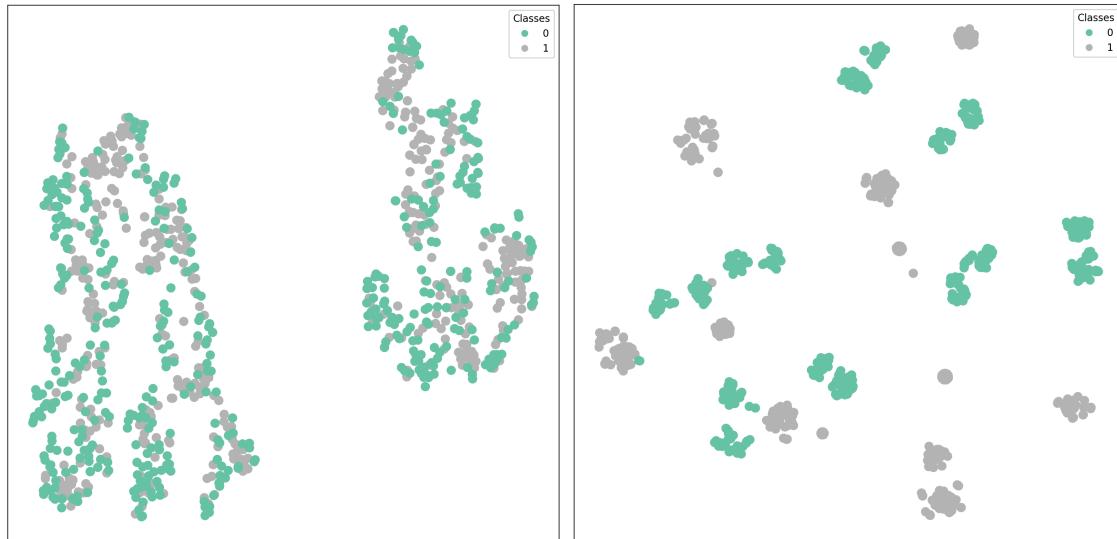


Figure 5.3: TSNE for training set before training.

Figure 5.4: TSNE for training set after training.

Figure 5.5: TSNE for GCN training dataset.

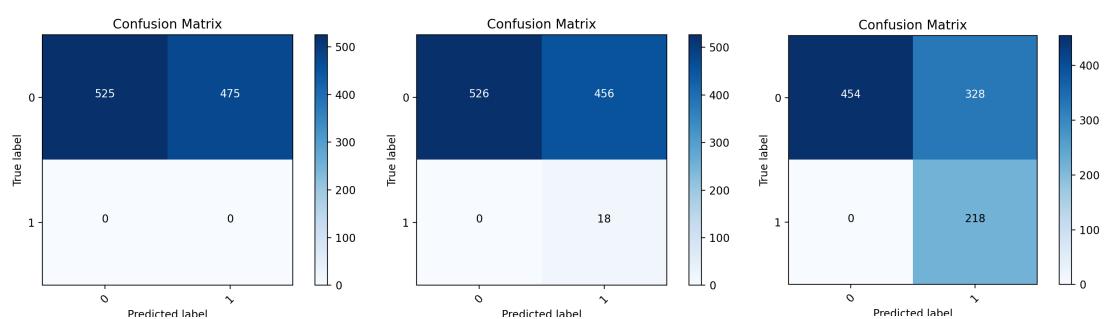


Figure 5.6: CM for TS1.

Figure 5.7: CM for TS2.

Figure 5.8: CM for TS3.

Figure 5.9: CM for GCN Test Datasets.

5.4 Results

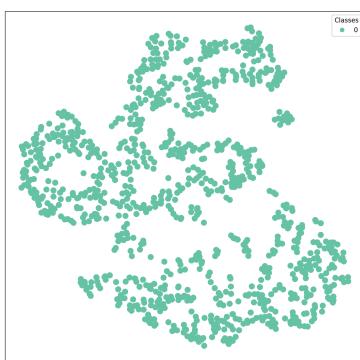


Figure 5.10: TSNE for TS1 before.

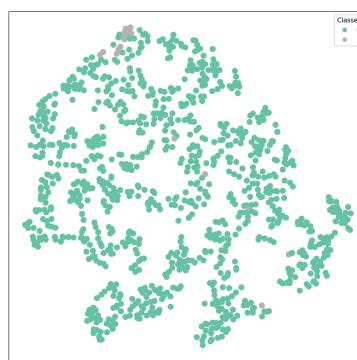


Figure 5.11: TSNE for TS2 before.

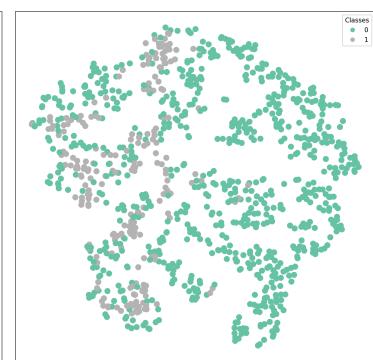


Figure 5.12: TSNE for TS3 before.

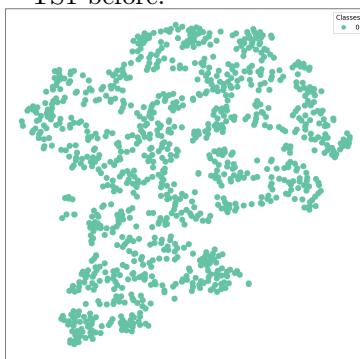


Figure 5.13: TSNE for TS1 after.

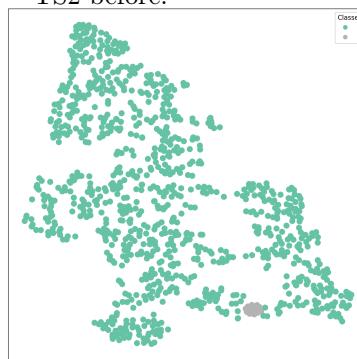


Figure 5.14: TSNE for TS2 after.

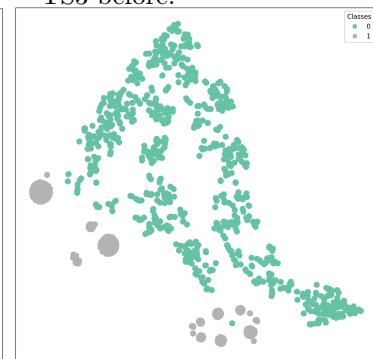


Figure 5.15: TSNE for TS3 after.

Figure 5.16: TSNE for GCN Test Datasets with naive edge weights.

5. REPLACEMENT FOR GRAPH COMMUNITY DETECTION STEP

6

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.

The Multi Layered Perceptron presented in Chapter 4 capable of identifying causally related hits with a higher accuracy, precision and recall compared to the Pattern Matrix Criterion presented by Karas et al. and thus is considered a viable successor to the PMC. Although experiments were done to identify the optimal parameters such as the batch size, optimization function, learning rate, height and depth of the network, further experimentation is recommended before it is integrated into the Data Acquisition Pipeline. The model also showed an increase in the number of FPs when the number of positive class examples are increased. It may be possible to correct this bias by adding some regularization into the model (?).

Table 6.1: Advanced edge weight scheme for GCN.

Node type(s)	Edge weight
noise-noise	1.0
event-event (causally related)	1.0
event-event (causally unrelated)	0.5
event-noise	0.1

The GCN obtained in Chapter 5 has a perfect recall however is also significantly biased to the positive class (event nodes) and thus unable to identify the negative class (noise nodes). The origin of the problem can be traced back to the edge weights being applied to the graph. The weights supplied by the MLP only take into account edges between

6. RECOMMENDATIONS

causally related and unrelated hits. In reality however, unrelated hits consists of various sub categories: 1. noise-noise hits 2. noise-event hits and 3. causally unrelated event-event hits. Although from a physics point of view the pairs of hits listed above are in fact causally unrelated, this *naive* edge weight scheme does not allow the GCN to learn adequately. The root cause of this conflict is based in the fact that the two models have contradictory goals. Whilst the MLP is trained to identify causally related and unrelated hits, the GCN is required to identify hits originating from neutrino events and noise. The solution is simple and requires assigning edge weights based on the nodes that it connects as summarized in Table 6.1. The GCN model is trained using the *advanced* edge weight scheme and the results are promising as observed in Figure 6.8 and 6.4. The model now has perfect discriminatory skills for both classes even in the extremely skewed datasets like TS2.

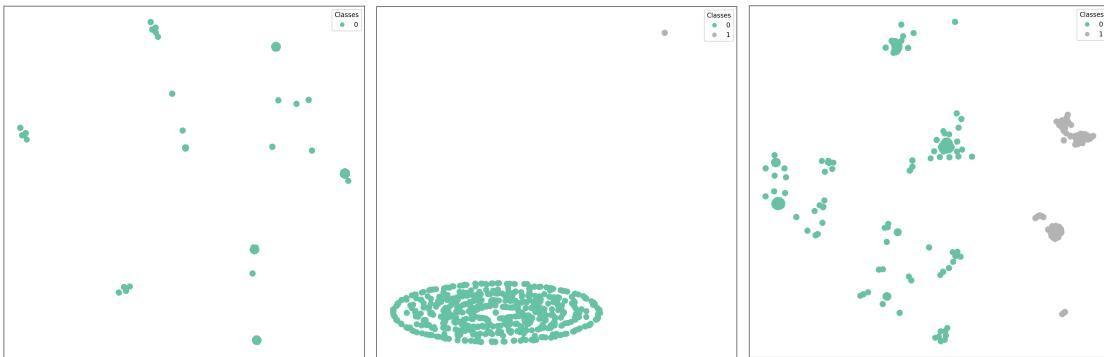


Figure 6.1: TSNE for TS1 with advanced edge weights.

Figure 6.2: TSNE for TS2 with advanced edge weights.

Figure 6.3: TSNE for TS3 advanced edge weights.

Figure 6.4: TSNE for GCN test datasets with advanced edge weights.

Although the GCN is now capable of identifying event and noise nodes with immaculate accuracy, precision and recall, it however classifies all examples as false positives in TS1 which contains no event nodes. The model is thrown off by the high weights on edges between causally related event nodes and noise nodes, which may be possible correct by framing the data as a multi-edge heterogeneous graph. By creating different types of edges corresponding to the various types of connections that two nodes may posses, each carrying the corresponding weights, the network may be able to correct it's bias to the positive class. In order to obtain the advanced edge weight, 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

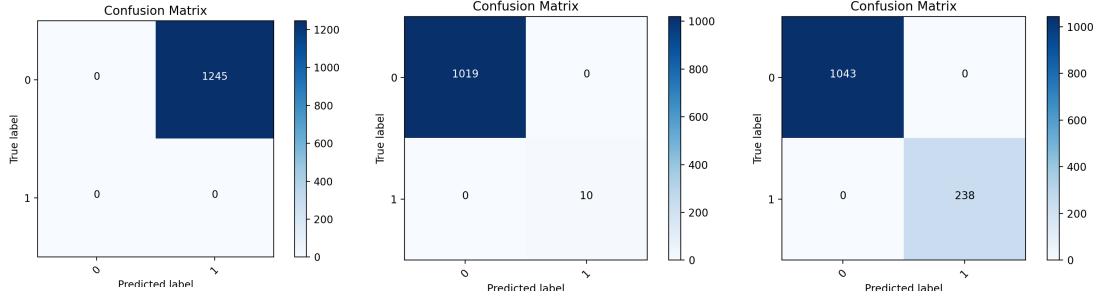


Figure 6.5: CM for TS1
(advanced edge weights).

Figure 6.6: CM for TS2
(advanced edge weights).

Figure 6.7: CM for TS3
(advanced edge weights).

Figure 6.8: CM for GCN test datasets with advanced edge weights.

probability for each class. These probabilities can then be used as the edge weights as illustrated in Figure ???. The MLP may not be able to discern causally unrelated hits since they are spread evenly in space and time, thus research to identify an appropriate model for the task is required. One approach may be to use a GCN to perform classification of the edge types or regression on the edge weights (18) however this remains to be validated.

Table 6.2: Summary of GCN performance across test sets with advanced edge weights.

	Accuracy	Precision	Recall	F1	F2	ROCAUC	PRAUC
TS1	0.00	—	—	—	—	—	—
TS2	1.00	1.00	1.00	1.00	1.00	1.00	1.00
TS3	1.00	1.00	1.00	1.00	1.00	1.00	1.00

We observed a high coupling of the Hit Correlation Step with the subsequent steps of the Karas Pipeline. The output of the PMC is used by the steps that follow so any shortcomings of the PMC cascade down the pipeline and affect the performance of the latter steps and the pipeline as a whole. This coupling is also present in the new pipeline as the performance of the MLP directly dictates the performance of the GCN. This coupling is not ideal and an effective solution to decouple the two models is required. One such solution which remains open as a path of research is to explore the possibility of replacing the entire pipeline with a single GCN. The data can be framed such that the (x, y, z) vectors are used as the node embedding. If the t is scaled between $[0, 1]$ and the complement of δt is assigned as the edge weights, it should result in a graph where edges between causally related nodes carry a high weight. After convolution, a reasonable expectation is the presence of small and tightly connected communities of causally related nodes and

6. RECOMMENDATIONS

large, weakly connected communities of noise nodes. The GCN model can now be modified to perform graph classification (19) instead of node classification, the premise being that presence of small, densely connected communities indicate the timeslice is important and thus should be saved for further analysis. Alternatively, the Line Graph Neural Network proposed by Chen et al. (2017) can also be used to perform community detection.

The models presented in this report were tested in isolation. The intended usage however is to use them in tandem in order to identify timeslices containing neutrino event hits. Thus it is recommended that the models be tested as an integrated pipeline so that the results may be compared to that of The Karas Pipeline. Additionally, performance of the pipeline should be observed using various permutations of the individual parts of the new and the old pipeline to determine the best performing combination overall. As noted in Section 1.2, the runtime performance of the pipeline is crucial and should be able to perform filtration in near real time. As neural networks can be parallelized using the compute power of GPUs, the models should be capable of meeting the requirements. Several existing work also indicate the feasibility of scaling GCNs to parallel computation over large graphs (20, 21, 22). Of course at this point there are merely speculations and empirical proof still requires to be gathered.

Bibliography

- [1] 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
- [2] 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
- [3] 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. 2
- [4] MAARTEN POST. *"KM3NNeT" A neural network for triggering and classifying raw KM3NeT data.* PhD thesis, Universiteit van Amsterdam, 2019. 2, 3
- [5] KONRAD KARAŚ. *Data processing pipeline for the KM3NeT neutrino telescope.* PhD thesis, Universiteit van Amsterdam, 2019. 2, 9, 10
- [6] NumPy - The fundamental package for scientific computing with Python. 5
- [7] Pandas. 5
- [8] YOSHUA BENGIO. **Practical recommendations for gradient-based training of deep architectures.** In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012. 14, 16

BIBLIOGRAPHY

- [9] IAN GOODFELLOW, YOSHUA BENGIO, AND AARON COURVILLE. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [10] SEBASTIAN RUDER. **An overview of gradient descent optimization algorithms.** *arXiv preprint arXiv:1609.04747*, 2016. 15
- [11] AMICHAI PAINSKY AND GREGORY WORNELL. **On the universality of the logistic loss function.** pages 936–940, 2018. 16
- [12] CHIGOZIE NWANKPA, WINIFRED IJOMAH, ANTHONY GACHAGAN, AND STEPHEN MARSHALL. **Activation functions: Comparison of trends in practice and research for deep learning.** *arXiv preprint arXiv:1811.03378*, 2018. 16
- [13] GANG WANG, GEORGIOS B GIANNAKIS, AND JIE CHEN. **Learning ReLU networks on linearly separable data: Algorithm, optimality, and generalization.** *IEEE Transactions on Signal Processing*, **67**(9):2357–2370, 2019. 16
- [14] PAULA BRANCO, LUIS TORGÓ, AND RITA RIBEIRO. **A survey of predictive modelling under imbalanced distributions.** *arXiv preprint arXiv:1505.01658*, 2015. 17, 19, 20
- [15] ALBERTO FERNÁNDEZ, SALVADOR GARCÍA, MIKEL GALAR, RONALDO C PRATI, BARTOSZ KRAWCZYK, AND FRANCISCO HERRERA. *Learning from imbalanced data sets*. Springer, 2018. 19
- [16] NITISH SRIVASTAVA, GEOFFREY HINTON, ALEX KRIZHEVSKY, ILYA SUTSKEVER, AND RUSLAN SALAKHUTDINOV. **Dropout: A Simple Way to Prevent Neural Networks from Overfitting.** *Journal of Machine Learning Research*, **15**(56):1929–1958, 2014.
- [17] LAURENS VAN DER MAATEN AND GEOFFREY HINTON. **Visualizing data using t-SNE.** *Journal of machine learning research*, **9**(Nov):2579–2605, 2008.
- [18] LIYU GONG AND QIANG CHENG. **Exploiting edge features for graph neural networks.** pages 9211–9219, 2019. 33
- [19] MUHAN ZHANG, ZHICHENG CUI, MARION NEUMANN, AND YIXIN CHEN. **An end-to-end deep learning architecture for graph classification.** 2018. 34

BIBLIOGRAPHY

- [20] YUZHE MA, HAOXING REN, BRUCEK KHAILANY, HARBINDER SIKKA, LIJUAN LUO, KARTHIKEYAN NATARAJAN, AND BEI YU. **High performance graph convolutional networks with applications in testability analysis**. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.
- [21] LINGXIAO MA, ZHI YANG, YOUSMAN MIAO, JILONG XUE, MING WU, LIDONG ZHOU, AND YAFEI DAI. **Neugraph: parallel deep neural network computation on large graphs**. In *2019 {USENIX} Annual Technical Conference ({USENIX}){ATC} 19*, pages 443–458, 2019.
- [22] HANQING ZENG, HONGKUAN ZHOU, AJITESH SRIVASTAVA, RAJGOPAL KANNAN, AND VIKTOR PRASANNA. **Accurate, efficient and scalable graph embedding**. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 462–471. IEEE, 2019.