Classifying Charged Current Neutrino Events Using Machine Learning in the MINERνA Experiment

Nilotpal Kakati
Roll No: MS14074

A dissertation submitted for the partial fulfilment
of BS-MS dual degree in Physics

Under the guidance of
Dr. Satyajit Jena

April 2019

Indian Institute of Science Education and Research Mohali
Sector - 81, SAS Nagar, Mohali 140306, Punjab, India

Dedicated to my family and friends

Certificate of Examination

This is to certify that the dissertation titled **"Classifying charged current neutrino events using Machine Learning in the MINERvA experiment"** submitted by **Nilotpal Kakati** (Reg. No. MS14074) for the partial fulfillment of BS-MS dual degree program of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Jasjeet S. Bagla Dr. Anosh Joseph Dr. Satyajit Jena
(Supervisor)

Dated: 26.04.2019

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Satyajit Jena at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Nilotpal Kakati
(Candidate)

Dated: April 26, 2019

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. Satyajit Jena
(Supervisor)

Acknowledgement

It is my privilege to express my sincere gratitude to my supervisor, Dr. Satyajit Jena for his invaluable discussion, guidance and supervision throughout the work. The confidence and dynamism with which he guided the work require no elaboration. Without his guidance, compiling the project report in this form would have been impossible. I am thankful to my thesis committee members Prof. Jasjeet S. Bagla and and Dr. Anosh Joseph for spending their valuable time and offering useful suggestions during the assessment of my thesis work. Special thanks to Dr. Gabriel N. Perdue (Fermilab), Dr. Anushree Ghosh (UTFSM) and Dr. Jonathan Miller (UTFSM) for their constant guidance throughout the project.

I am thankful to all my lab members - Rohit bhaiya, Nishat di, Karthik bhaiya, Akhil bhaiya, Kriti di, Ojaswi, paras and Vassu for keeping me motivated and to have maintained a cheerful and comfortable atmosphere in the lab. I am thankful to all my friends in IISER Mohali especially Parth, Nishant, Pinku, Somak, Vishnu, Himanshu for their heartily support throughout my stay at IISER Mohali. I am thankful to DST-INSPIRE for the funding to carry out research work and Fermilab for providing the computational resources needed for the project. I would also like to thank Dr. Kirandeep for proofreading this thesis providing useful inputs.

Finally, I would like to express my heartfelt gratitude to my parents and family members for their unconditional love, support, and encouragements for accomplishing my dream.

Nilotpal Kakati

MS14074

IISER Mohali.

List of Figures

Contents

Abstract

Neutrinos are by far the second most abundant particles in the universe. About 100 trillion neutrinos pass through our body every second and we don't even realize it. The reason behind this ghostly presence is that they are chargeless and their mass is negligible. These unique features enable them to play an important role in the universe. Physicists believe that studying neutrinos may give us a better insight to still unanswered questions like the matter-anitimatter imbalance. But before answering such questions and understanding the role of neutrinos in the universe, we need to understand how they interact with matters; and MINERvA is one such attempt. It's an experiment in Fermilab which is being conducted to precisely characterize different types of neutrino interactions, and to study the physical processes that govern these interactions. Studying those interaction directly is not possible and hence we study the final state particles produced after such interaction instead, and try to understand the interactions from the information inferred from the particles. The experimental observations only give us information about the energy deposited by the particles while they travel through the detectors, but we need to know the type of particles in order to understand the interaction. In our approach, the gap between the two is bridged using Machine Learning (ML). We try some state of the art ML algorithms which have been proven to perform well in similar problems from other fields, and see how they perform in the problem at hand.

Chapter 1

Neutrinos

1.1 Why study neutrinos

The standard model of particle physics was proposed in 1970 which describes the fundamental forces and particles that make up all the matter. Despite being considered the most successful theory of particle physics, there are fundamental natural phenomena that the standard model fails to explain. This leads us to the physics beyond the standard model and neutrinos are excellent evidence for it.[SM16] In the standard model neutrinos have zero mass. But it has been experimentally confirmed that neutrinos oscillate, i.e. they change flavor while travelling from their sources.[Wil94] [F$^+$98] In order to oscillate, both the flavors involved cannot have same mass which implies that they cannot be both zero. In fact experiments have proven it that all the three types of neutrinos have mass. This challenges the completeness of the standard model.

Neutrinos are chargeless and their mass is negligible. These unique and interesting features enable them to play an important role in the universe. They are produced in the nuclear fusion process that powers the sun and the stars, in the radioactive decays that provides a source of heat inside our planet, and also they are produced in the nuclear reactors. They can emerge from the deep inside of astrophysical objects, thus becoming a very promising carrier of information. Since they interact only through electroweak interaction, they are considered to be an excellent tool to study the structure of nucleons. [min]

1.2 Types of neutrino-nucleus interactions

There are three flavors of neutrinos, namely - electron neutrino (ν_e), muon neutrino (ν_μ) and tau neutrinos (ν_τ). The neutrino-nucleus interactions can be of two types -

- Neutral current interactions - Here the mediating particle is a neutral Z^0 boson.

$$\nu_\mu N \rightarrow \nu_\mu X$$

- Charged current interactions - Here the mediating particle is a charged $W^\pm$ boson.

$$\nu_\mu N \rightarrow \mu^- X$$

Fig 1.1 shows the Feynman diagram for a charged current event.

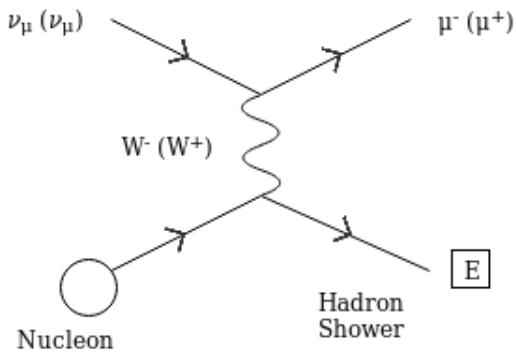

Figure 1.1: Charged current event

In the MINERvA experiment, the neutrino beam used contains only muon neutrinos/anti neutrinos (we will discus it later in the second chapter). Since we are only interested in the charged current events, in the final state particles we will always have a muon

1.2.1 How do we study the interactions

As discussed before neutrinos interact only via weak interactions and hence we cannot study them directly. So, in experiments like MINERvA, there are nuclear target layers. The neutrinos used in the experiments will collide with nuclei in various target materials, producing ionizing radiation and secondary particles. These secondary particles leave measurable energy deposits in the detector. [min]

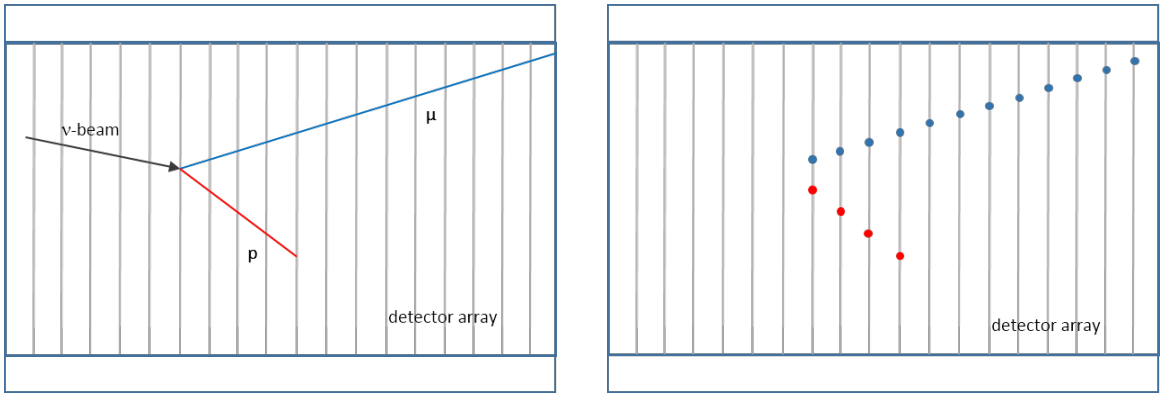

Figure 1.2: Event example

Fig 1.2 gives us a schematic of how an event is viewed in a detector array. In the first image, we can see an event happening where a muon interacts with a nucleus, and we get a proton and a muon as final state particles. The second image gives us an impression of what information we get from the detectors. The blue and red dots represent the energy hitpoints recorded by the detector. These energy deposits are later used for identifying the secondary particles produced, helping in the study of neutrino nucleus interactions.

1.3 Defining the problem

For our project work, we are interested in classifying the charged current neutrino-nucleus interaction events based on the final state particles using the data recorded by the detector. To simplify the problem, we decided to divide the data into the following classes -

- Class 1 : (0 proton, 0 pion, 1 muon) events
- Class 2 : (0 proton, 1 pion, 1 muon) events
- Class 3 : (1 proton, 0 pion, 1 muon) events
- Class 4 : (1 proton, 1 pion, 1 muon) events
- Class 5 : everything else

Our goal is to come up with an artificial neural network which, after successful training, can predict the events into these classes.

This thesis is divided into the following parts - the first chapter introduces us to the problem, the second and third chapter give brief introduction to the experiment MINERvA, and machine learning respectively. Fourth and fifth chapters walk us through the methodology and results and the sixth chapter summarizes the work. *Since we are working in the classification domain, though not mentioned explicitly, we look at everything from a classification point of view for the rest of thesis.*

Chapter 2

The MINERνA Experiment

2.1 Introduction

The field of experimental neutrino physics is comparatively new. The unique properties, such as taking part only in weak interactions, unique flavor sensitivity make the neutrinos an ideal probe; but before studying their interaction one needs to tackle issues like small cross-section and difficulty in producing neutrino beams. In the recent years, a lot of effort has been made and as a result we currently have quite a few experiments trying to explore this intensity frontier.

It's been already proven that neutrinos oscillate, i.e., they change flavor while propagating from the source. This oscillation comes from the mixing of the mass eigenstate of neutrinos. Current state of neutrino physics is mainly centered around neutrino oscillation. In particular, the near future neutrino experiments like DUNE (Deep Underground Neutrino Experiment) will focus on determining the mass hierarchy and CP violation by comparing the oscillation probabilities of neutrinos and anti-neutrinos. [S$^+$16] But measurements with such high precision demands better understanding of neutrino-nucleus (and antineutrino-nucleus) interaction at the relevant energy range.

MINERvA aims to address this issue by studying neutrino reactions with five different nuclear targets using a very high intensity neutrino beam produced in the Neutrinos at the Main Injector (NuMI) beamline. It is the first experiment to have a self contained comparison of interactions with different elements with very high precision. The data obtained

from MINERvA significantly improve the models of neutrino-nucleus scattering, thereby reducing the uncertainty in the results from the oscillation experiments. [A+14]

2.2 The Experimental Arrangement

MINERvA's compact detector design, use of a very high intensity neutrino beam and the proximity of the detector to the beam make it unique in its domain.

2.2.1 The NuMI neutrino beam

The source of MINERvA's neutrino (and anti-neutrino) beam is the the Fermilab NuMI beamline which produce a very high intensity beam of muon neutrinos and anti-neutrinos.The NuMI beam is created by firing protons from Fermilab's Main Injector into a carbon target. The interaction of protons with the target produces a stream of positively and negatively charged particles. A pair of magnetic horn is used to focus the particles which can be pulsed in either polarity. If the horns are focusing positively charged mesons, then resulting beam is primarily neutrinos, and if it is focusing the negatively charged mesons it would result in an anti-neutrino beam. Then the focused particles go through a 675 meter long decay pipe. Inside the pipe, the particles decay to produce muons and muon neutrinos (or anti-muons and muon anti-neutrinos). Then the beam passes through 240 meters of rock and muon absorbers where all the particles except the neutrinos are absorbed, creating a clean beam of neutrinos. [A+14]

2.2.2 NuMI detector hall

The neutrino beam then passes through the NuMI detector hall which is located 105 meters undergrounds. It contains the MINOS (Main Injector Oscillation Search) near detector, the MINERvA detector and the NOvA (NuMI Off-axis ν_e Appearance) near detector. The propagation axis of the neutrino beam passes through the MINERvA first, then MINOS and then it just continues its journey through Earth towards the Soudan mine.

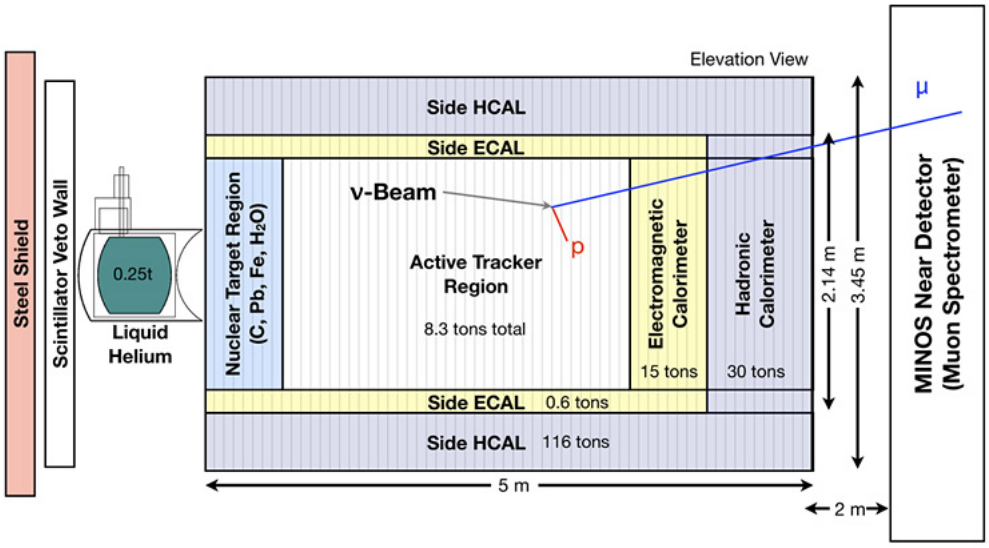

Figure 2.1: side-view of the MINERvA experimental arrangement. The neutrino beam travels from left to right. (Credit: Fermilab)

2.2.3 MINERvA

The MINERvA detector starts with a wall of steel, followed by another wall of the solid scintillator. This so-called "veto wall" is used to identify charged particles that were created in neutrino interactions with the rock surrounding the detector hall. Next to this veto wall is an aluminum cryostat containing liquid helium. After passing through the cryostat the neutrinos enter the main body of the detector. MINERvA has 200 hexagonal detector panels; each panel consists of 127 plastic scintillator strips with optical fiber in their centers. When a charged particle passes through a scintillator, it produces an amount of light which is proportional to the energy lost by the particle while travelling through the scintillator. The massive detector array record the energy loss of the particles and this information is used to create the three dimensional path, or track of the final state particles which ultimately can lead us to the origin of the neutrino interaction.

The front of the detector is called the nuclear target region as it contains the detector panels made of varying configurations of solid carbon, iron, and lead. These nuclear target panel separated by eight scintillator panels. It also has a liquid water target which sits between the 4^{th} and the 5^{th} nuclear targets. Thus, it enables researchers to have a self complete comparative study of neutrinos interaction with nuclei of different mass.

The next section of the detector is called the active tracker region. It consists solely of scintillator panels for tracking the final state particles. Both the nuclear target and active tracker regions are surrounded by electromagnetic and hadronic calorimeters which have alternating strips of scintillator and lead or steel sheets. The sheets help in slowing down and stopping the final state charged particles, allowing MINERvA to fully measure the energy of these particles. [A$^+$14]

MINERvA Coordinate System

The MINERvA coordinate system is defined such that the z-axis is horizontal and points downward along the central axis of the detector. The y-axis points upward and the x-axis is horizontal, pointing towards beam left, with x-y origin at the center of the inner detector. The z-axis is defined to place the front face of MINOS at z=1,200 cm. In this system, the neutrino beam central axis is in the y-z plane and points downward at 3.34 degree.

Module Assembly

The MINERvA detector is comprised of 120 modules suspended vertically and stacked along the beam line. There are four types of modules - tracking modules, electromagnetic calorimeter modules, hadronic calorimeter modules and passive nuclear targets. Tracking module is our main interest, since the data we use as input for our network is obtained from this module.

Tracking modules consist of two scintillator planes each composed of 127 triangular scintillator strips. A plane can have one of three different orientations, referred to as X-planes, U-planes or V-planes according to the coordinate in the MINERvA system in which each plane measures particle hit positions. X-planes have scintillator strips aligned vertically, hence hits in this view give position information in the horizontal or x-direction. The U- and V-planes are rotated 60 degrees clockwise and counterclockwise from the X-planes in the x-y plane, respectively. [A$^+$14]

Three different views are used in order to avoid ambiguities with reconstructed hit associations that can occur when multiple tracks traverse two orthogonal planes. Each tracking and electromagnetic calorimeter module has one X-plane, and either a U or V-plane,

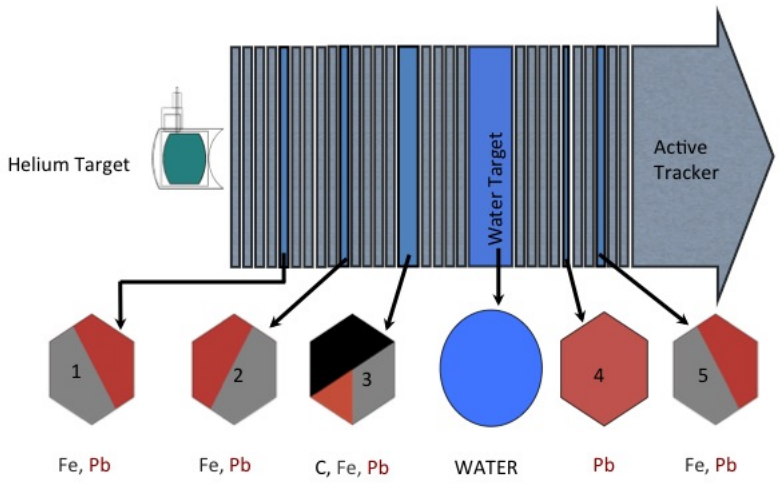

Figure 2.2: side-view of the arrangement of MINERvA's nuclear target and detector modules (Credit: Fermilab)

with modules alternating between a UX or VX structure with the X-planes always located downstream of the U or V-planes. Thus the number of X-planes is twice the number of U or V-planes, which is reflected in the data we use (the data for X-plane is of dimension 127x94 and the data for U or V-planes are of dimension 127x47).

Chapter 3

Machine Learning: An Introduction

Human brain excels in many things and its working has been a wonder for scientists for many years. One major difficulty in understanding how brain works is that *it doesn't follow any predefined rule*. We feel the world around us using the senses and using that information, the brain guides us in this world. But there are tasks in the world which are far complicated for human brains and are more suited for computers and sometimes it is difficult to *define* rules for such tasks. This let to the idea that it would be really helpful if we somehow can reflect the brains learning process in computers and implementation of such ideas is machine learning.

3.1 Machine Learning as a function approximator

As defined by Arthur Samuel, Machine Learning is a type of algorithm where a system can automatically learn to perform specific tasks without any explicit instructions. We can think of it as a search for a function $\mathbf{f} : \mathbf{X} \rightarrow \mathbf{Y}$, which maps a high dimensional space of observational data or *features* $\mathbf{X}$ to a low dimensional space of target *labels* $\mathbf{Y}$ of interest while optimizing some metric, called the loss function $\mathbf{L}(\mathbf{y}, \mathbf{f}(\mathbf{x}))$ [GCW18]

Ideally, we would like to optimize the loss over all the possible values of $(\mathbf{x}, \mathbf{y})$ from the underlying distribution $\mathbf{p}(\mathbf{x}, \mathbf{y})$, but this is not practically possible. So we usually stick to a dataset $\{\mathbf{x_i}, \mathbf{y_i}\}_{\mathbf{i=1}}^{\mathbf{N}}$, called the training set and a model. In this case, the algorithm will try to optimize the loss over the parameters of the defined model through training. The main objective we want to achieve through this process is generalization, i.e., we want our

algorithm to perform well on data that it hasn't already seen during training. Based on the problem, the data at hand and available computational resources, different types of models are used. Some commonly used examples are - Support Vector Machines (SVM), Decision Trees (DT) and Artificial Neural Network (ANN). ANNs are generally more complex models which require *large dataset* but they perform really well when the problem involves non-linear decision boundaries or absence of well-defined features. Since the classification problem we are working on involves detector data as features, we decided to use a specific type of ANN, called Convolutional Neural Network which we will talk about later.

3.2 Artificial Neural Network

Though Artificial Neural Networks have gained popularity in the recent years, the idea of using algorithms, inspired by the biological neural networks, to solve complicated problem can be traced back to the early twentieth century. Unfortunately, implementing those ideas into practice was not possible at that time and hence this field had to wait for quite a few decades to gain an interest among the researchers. But the recent advancement made in the filed of computational architectures has made it possible to try out these network and since then it hasn't looked back. The community started growing rapidly when graphical processing units (GPUs) became affordable for everyone and implementing those networks and training them didn't require any more access to special computational resources.

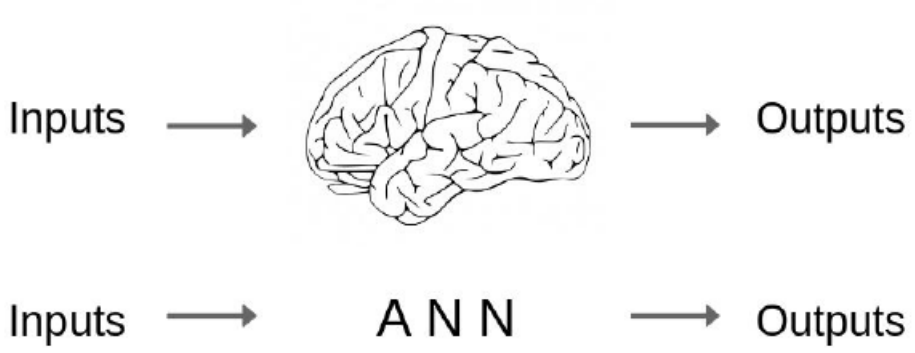

Figure 3.1: ANNs are inspired by biological neural networks

We can think of ANNs as function approximators with a very high number of parameters. [LS16] The network tries to learn these parameters via training. The function space of these

network is determined by the architecture which can be understand in terms of transformations. [GCW18] The input to a network is transformed into outputs through some hidden states. The transformations are defined as

$$h_{i+1} = g_i(W_i h_i + b_i) \tag{3.1}$$

where, h_i and h_{i+1} are the i^{th} and $i + 1^{st}$ transformations, the elements of matrix W are the weights, b is the bias vector, and g_i the activation function

The idea becomes more clear when we start talking in terms of the neurons. [neu] [Hay06] These neurons or perceptrons are computational units that make up the network. Again, the idea of neuron is inspired by the biological neurons which we still haven't been able to understand completely. Fig 3.1 shows an artificial neuron

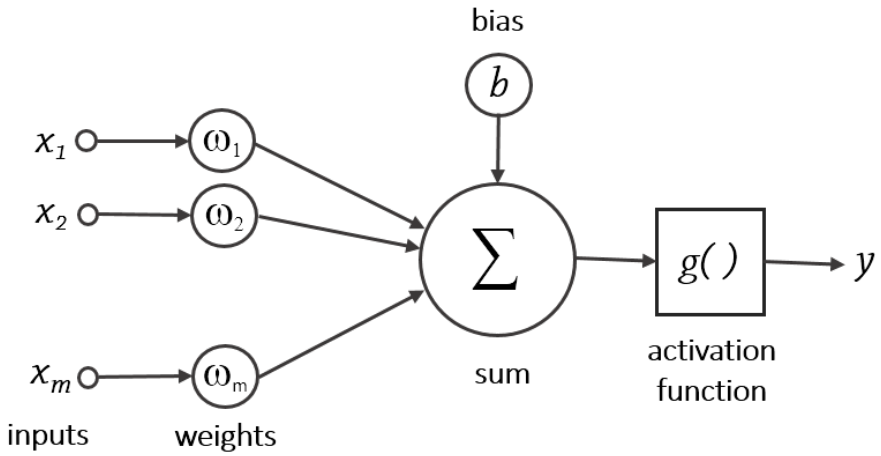

Figure 3.2: An artificial neuron

Here, h_i^j are the inputs to the $(i + 1)^{th}$ transformation. These inputs then get multiplied by the elements of the weight matrix. These weights determine how important the information from the previous neuron is. The higher the weight value, the more important that information is for the neuron. After multiplying the inputs with the weights, they get summed up; in addition we add a bias value to the sum.

These neurons are then stacked to form a neural network which can tackle complicated problems. [fun] But as complexity of problem increases, we can't rely on linear solutions anymore. And even though adding more neurons increases the size of the parameter space,

it is still a linear function (with a very large number of parameters). To address this issue, we introduce a non-linearity into the system and the output of a neuron is first passed through an activation function, [RZL17] [NIGM18] before it becomes an input for the next neuron. Some of the commonly used activation functions are - binary steps, sigmoid, hyperbolic tangent, rectified linear units or ReLU and leaky ReLU.

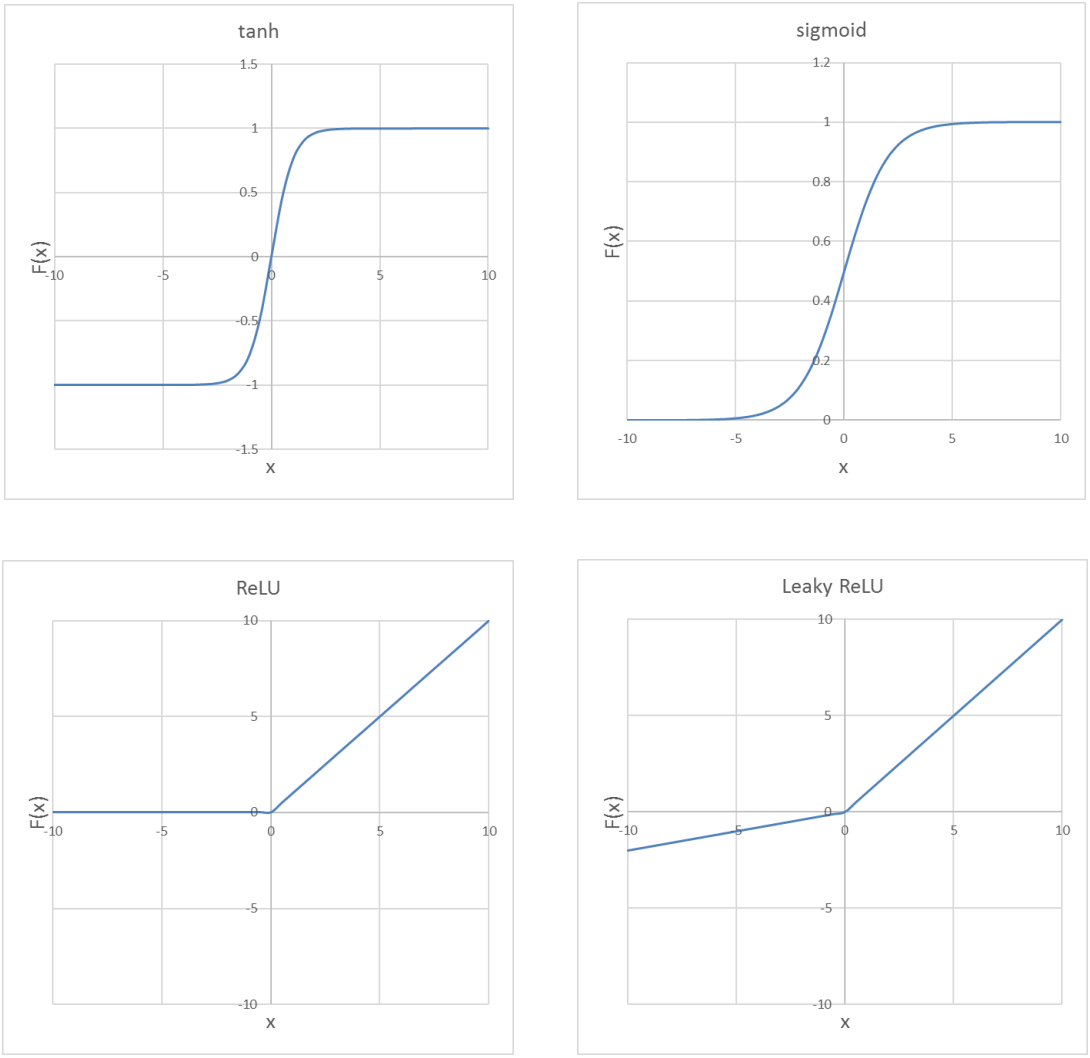

Figure 3.3: Some commonly used activation functions

After adding the activation functions, we stack the neurons again. But because of non-linearity, now the function space is much larger with the same number of parameters. While making a network, we stack neurons along both the axis. The number of neurons stacked in a layer is called the *width* of the network, and the no of layer stacked in a network is called the network *depth*.

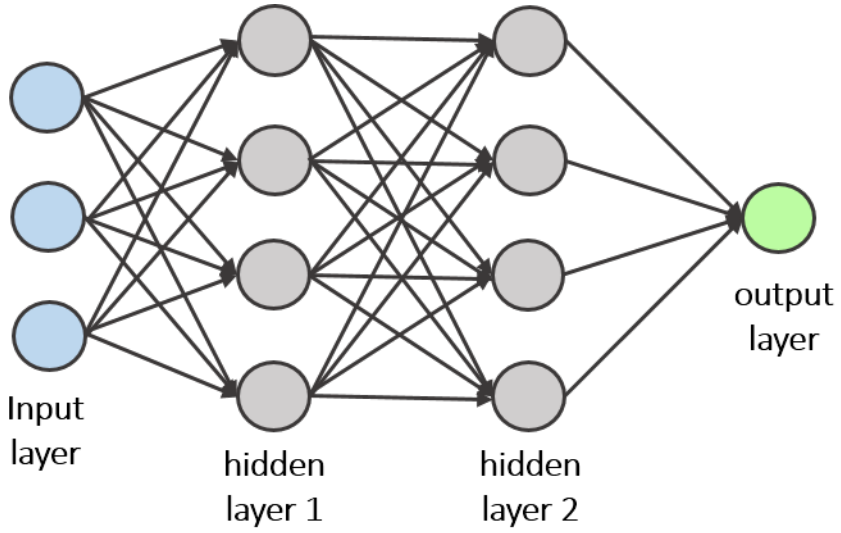

Figure 3.4: A complete network

3.2.1 Learning from flaws

When a network is created, the weights and the biases are initialized randomly. Through an iterative training process, the network learns these weights and biases. In theory, this training process involves all the training examples and the network calculates the gradient of the loss function with respect to each of the model parameter. [Rud16] Based on the gradients, the loss and bias values are updated. This process is repeated until the loss value saturates and the network can't learn anymore from training. In practice, the calculation is done based on a method called *backpropagation* and the weights and the biases are updated after each small *batches* of training sample. [GBC16] [LZCS14]

3.2.2 Evaluating the network performance

The goal of training a network is to make it understand the underlying distribution based on the training data so that it can predict on the data it has never seen during training. Thus, generalization is a key concept here, and in order to keep track of it, we usually divide the data available into a *training set*, a *validation set* and a *test set*. As discussed before, the training set is used for training and updating the model parameters based on the gradient of the loss. In order to inspect how well the network is generalizing, we use another dataset, and this dataset is the validation set. The model parameters are never updated based on how it performs on the validation set. And to compare how different networks are performing, we use the third dataset, the test set.

In order to keep track of the network performance, mainly the following metrics are monitored -

- the training and the validation loss profiles
- the validation accuracy profile
- the confusion matrix

After training every batch, the training loss is calculated since it's needed for updating the parameters. These values give rise to the training loss profile. Along with this training loss, the loss with the validation set is also calculated after a definite interval and this gives rise to the validation loss profile. The training loss profile tells us how well the network is learning, and the validation loss profile tells us how well the network is able to generalize.

The validation accuracy profile is another way of monitoring performance. It tells us what fraction of time the network is predicting the correct output. Mathematically,

$$accuracy = \frac{no\ of\ correct\ predictions}{total\ no\ of\ predictions} \tag{3.2}$$

The same can be calculated for training as well, but in general, training is done in small batches and accuracy is a statistical quantity. Hence it requires a large sample and a common practice is to avoid calculating it during training.

The Confusion matrix, also called the error matrix, is a tabular way of describing the performance of a network used for classification. For a binary classifier, we can define it as the following -
For a well-performing network, we would expect the matrix to be diagonal. It is considered as a powerful tool, as it not only gives us an insight into the errors, but also tells us what type of errors are being made. We can also normalize this matrix along the rows and the columns. The row normalized matrices, also called the precision, can be interpreted as efficiency which gives us what fraction of examples from a class is predicted correctly. Similarly the column normalized matrices, called recall, can be interpreted as purity which gives us what fraction of time an example actually belongs to a class the network predicts.

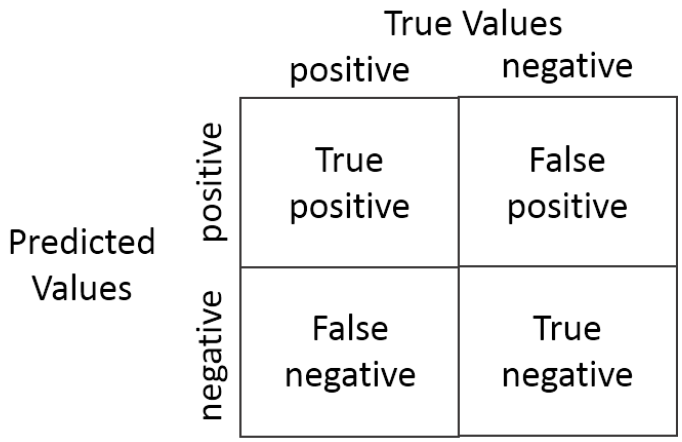

Figure 3.5: Confusion matrix

These two types metrics become really useful when the class distributions are not well balanced, i.e., number of examples in each class differs greatly in numbers. They also come handy in problems where tolerance against false positive and false negative are different.

3.3 Convolutional Neural Networks

As mentioned before, convolutional neural networks (CNNs) are a special variant of deep ANN. These networks vaguely resembles the organization of animal's visual cortex and are used for probelms involving visual image analysis. It was first proposed by Yan Le-Cun in 1989 and with the advent of modern computational architectures, it started gaining popularity. [LBBH01]

Like any other ANN, a CNN also have an input layer, an output layer and multiple hidden layers. The hidden layers consist of mainly convolutional layers, pooling layers and fully connected layers, and these convolutional and fully connected layers are what make them unique in the domain of images.

3.3.1 Convolutional layers

Convolutional layers are the core building blocks of a CNN. The layers parameters consist of a set of learnable filters or kernels which have a small receptive field, but extends through the full depth of the input volume. Each filter is convolved across the width and height of the input volume, computing the dot products between the filter and the part of the input in the receptive field. It produces a two dimensional feature map (also called activation map)

of the filter. Through training the network learns the weights in the kernels so that it can extract the features important for that particular problem.

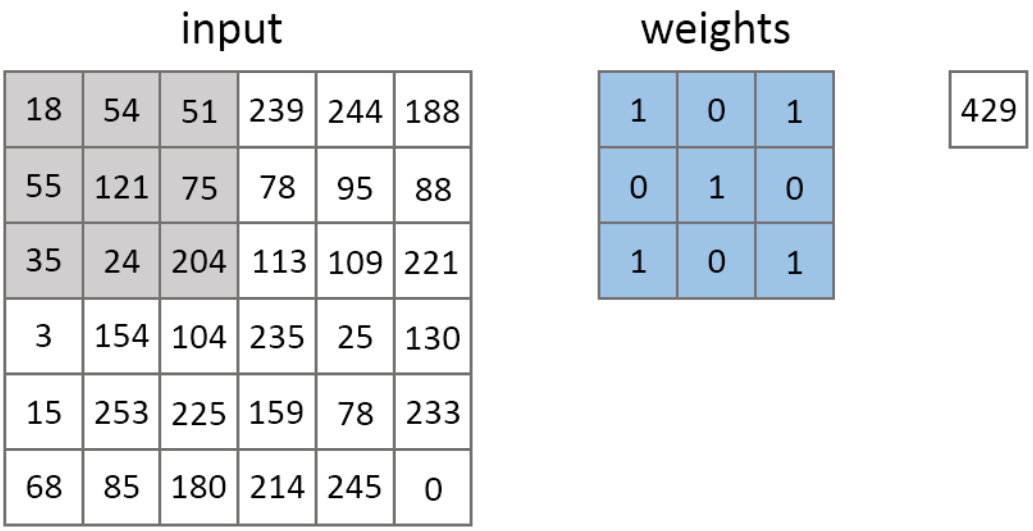

Figure 3.6: convolution in action

3.3.2 Pooling layers

Pooling is way of downsampling. If the feature map is large along the spatial axes, the number of parameters become really large as well which may lead to issues like overfitting. As a solution, based on the filter size, the input volume is divided into smaller volumes along desired axis; and from these smaller volumes a new value is obtained through some predefined operations like. Average pooling and max pooling are the most common pooling layers. Along dimension reduction, these layers also introduce non-linearity while keeping the number of parameters same. This also impacts the network performance.

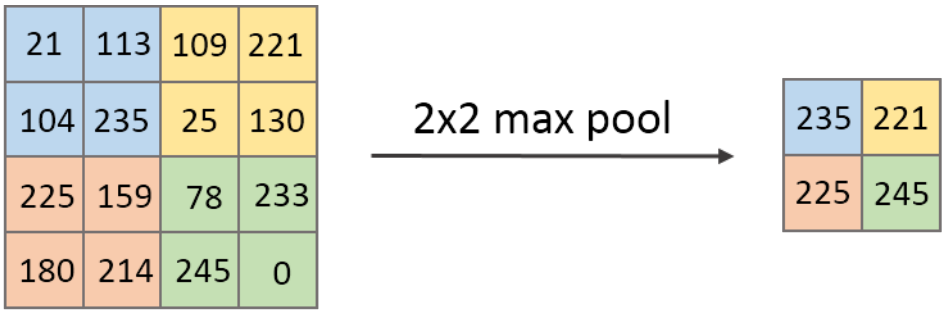

Figure 3.7: pooling in action

3.3.3 Fully connected layers

Fully connected layers or the dense layers consist of neuron stacked in layers. The output of multiple convolution and pooling is reshaped into one dimension and it becomes the input for the fully connected layers. These fully connected layers are repeated a few times and finally they get connected to output layer.

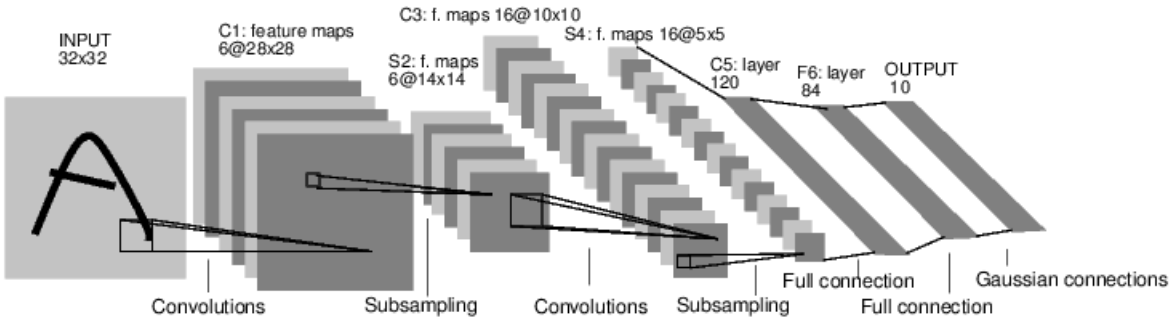

Figure 3.8: Complete CNN architecture. Credits: EBLearn Tutorial

3.4 Why CNNs work so well

There is no clear explanation of why CNNs excel in image analysis. A lot of researchers consider ANNs in general to be a black box. But CNNs are different; they are far more intuitive. The main principle CNN is based on is that the locality of the information doesn't matter. As long as the features of interest are present, we can detect them and use them for further recognition. By stacking convolutional and pooling layers, more and more abstract features can be extracted from an image as we move away from the input layer. These *more abstract features* can finally be mapped into the output layer and thereby enabling the network to have correct prediction. [QYLC18] [CW17]

Modern network architectures, like ResNets, do not completely follow the trend we just discussed. The new architectures exploits the convolutional feature extraction more and tend to avoid less intuitive approaches like use of pooling or fully connected layers. But the main idea still remain the same - extract the features from images and map them to the desired classes through subsequent more abstract feature extractions.

Chapter 4

The Multiclass approach

4.1 Getting back to our problem

As described before, we are interested in classifying the charged current neutrino-nucleus interaction events based on the final state particles using the data recorded by the detector. To simplify the problem, we decided to divide the data into the following classes -

- Class 1 : (0 proton, 0 pion, 1 muon) events
- Class 2 : (0 proton, 1 pion, 1 muon) events
- Class 3 : (1 proton, 0 pion, 1 muon) events
- Class 4 : (1 proton, 1 pion, 1 muon) events
- Class 5 : everything else

Our goal is to come with a network and train it with the labeled data of the pixel information recorded by the detectors. Before starting with all this, it is really important to understand the class balance of the data set as any imbalance may lead to bias in the network predictions.

4.1.1 Class balance

The data we used here are the processed subruns from MINERvA Medium Energy run NX ME1A MC, represented as images that do not contain target padding. Fig 4.1 shows the class balance for the dataset, and we can clearly see the class imbalance that is present.

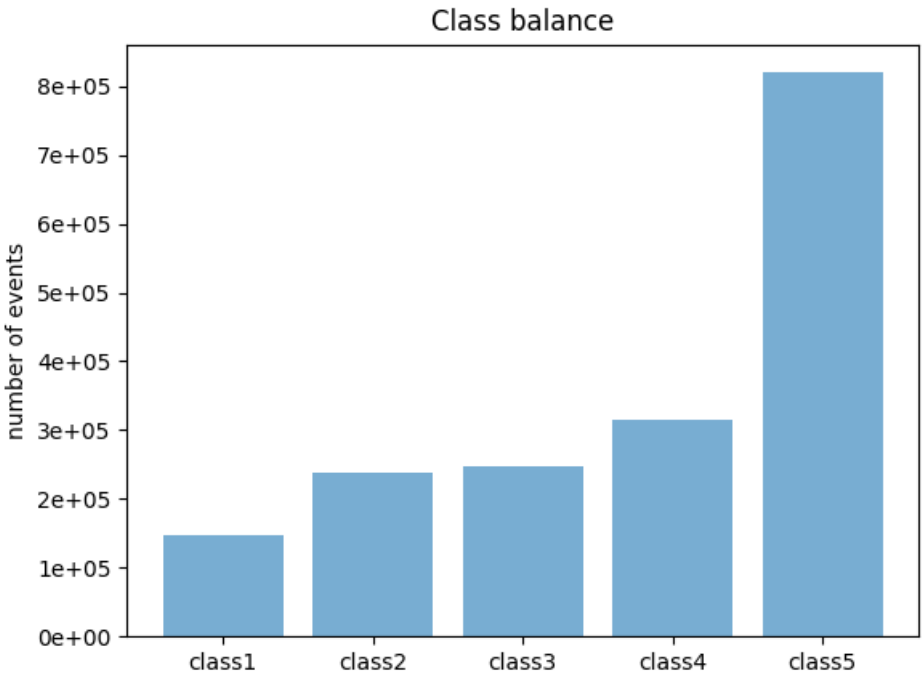

Figure 4.1: Class balance

Training with this dataset will allow the network to encounter more number of examples from class 5 and this may make the network prediction biased towards class 5. There are various methods of dealing such imbalance issues, but our first attempt would be to make a network, train it and see how it performs.

4.2 Choosing the network architecture

Choosing the right network architecture is always a crucial part of solving any problem and is mainly guided by the data and the information we want to extract from it. The data we are using are time and energy tensors recorded by the X, U and V detector arrays. The tensors have shape 2x127x47 for U and V views and 2x127x94 for the X view. Fig 4.2 shows the energy tensors for an event

As discussed in chapter 3, convolutional neural network is an excellent approach to our problem since the data we have can easily be interpreted as images. To be specific, we decided to try a VGGNet architecture. VGGNet was invented by the Visual Geometry Group (VGG) from oxford [SZ15] which is known for its performance in the ImageNet classifi-

cation competition. There are other modern architectures which outperform VGGNet, but because of its simplicity we considered it to be good starting point for our problem.

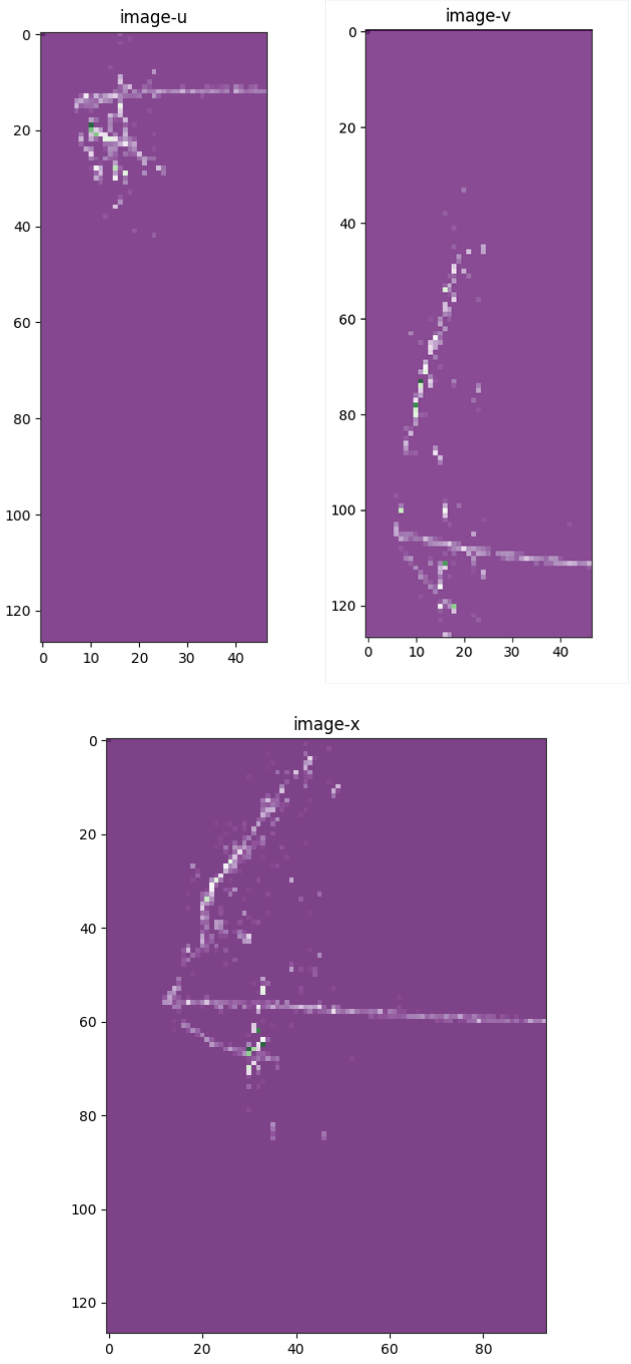

Figure 4.2: An event example (as recorded by the U, V and X planes of the detector respectively)

Fig 4.3 shows the network architecture we used. Like VGGNet it has two convolutional layers followed by a maxpooling layer. This process is repeated a few times and then the

output is flattened. We apply these layers independently on all the views and after flattening we concatenate all of them. Then a few fully connected layers are added and finally it gives the output vector.

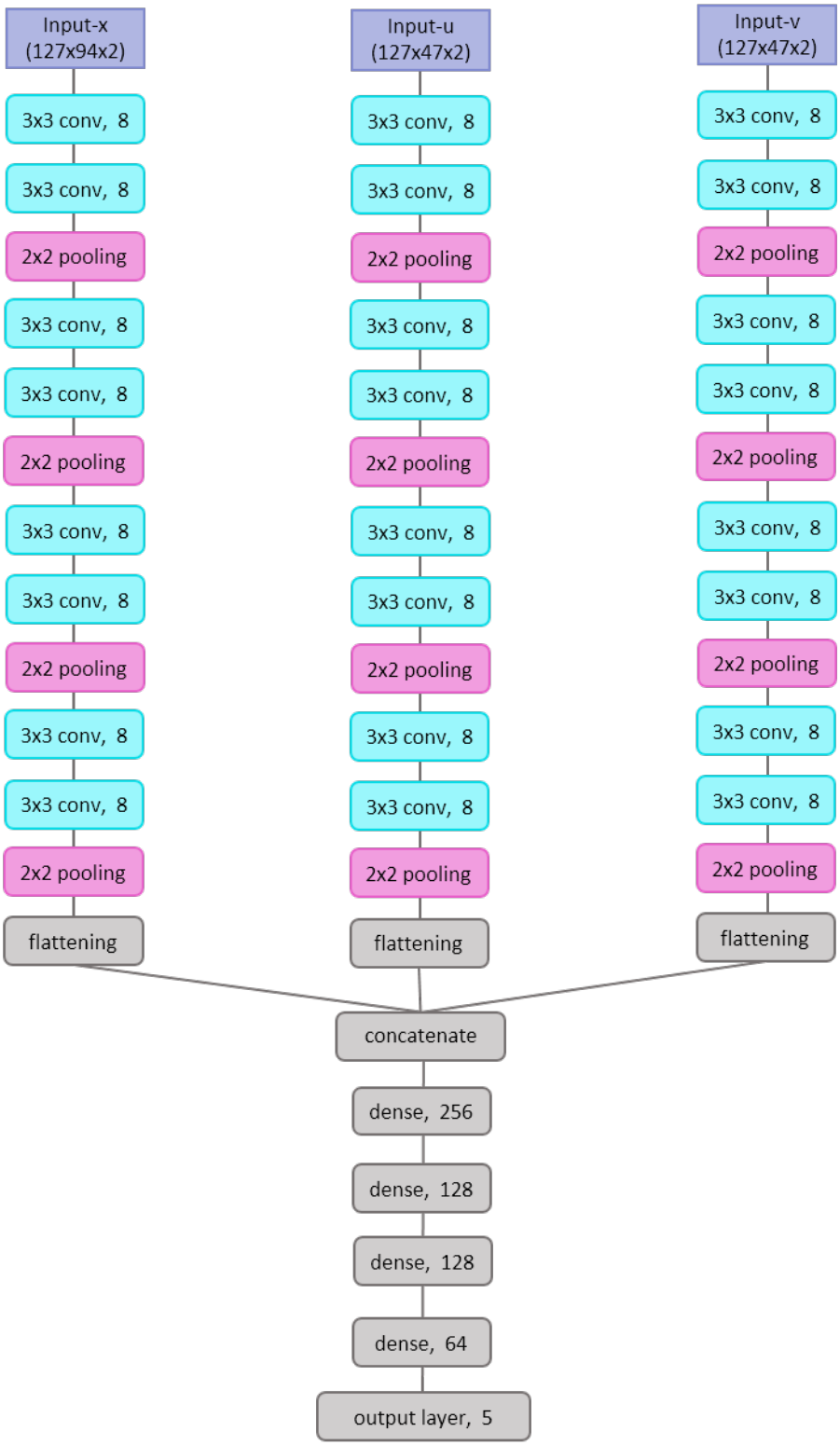

Figure 4.3: the network used (inspired by VGGNet)

The kernels used for convolution were all 3x3 and for pooling it were 2x2. The ReLU activation function was used with all the hidden layers, and output layer has softmax activation function which is defined as

$$\sigma(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{k} e^{a_j}} \tag{4.1}$$

where, a_i is the output of the i^th neuron and k is the number of classes. We used corss-entropy as the loss function which is again defined as

$$H(p, q) = -\sigma_{i=1}^{k} p_i log q_i \tag{4.2}$$

Here, p_i is the known probability of the i^{th} class and q_i is the predicted probability of the same. Batchnormalization and dropout layers were also added to the network for regularization and preventing overfitting. [SHK+14] [IS15]

4.3 Framework for implementing the network

There are quite a few good frameworks available for building and training ANNs. Tensorflow, pytorch, cafe are some of the most commonly used frameworks for this purpose. For our project, We decided to use tensorflow mainly because of its popularity and rapidly growing community. Despite not being as user friendly as some other machine learning frameworks like pytorch, the vibrant community and having the option of choosing one from the large number of high level APIs make working with tensorflow easier. The network was mainly written in the *keras API*, and the *estimator API* was used for training and evaluation. The data was converted to the *tfrecords*, a binary format native to the tensorflow framework which boosts up the training time by a significant amount.

4.4 Network performance

The network was trained for 5 epochs using the data and then the loss and accuracy profiles were plotted. (Here, an epoch is a measure of the number of times all of the training examples are used once to update the weights).

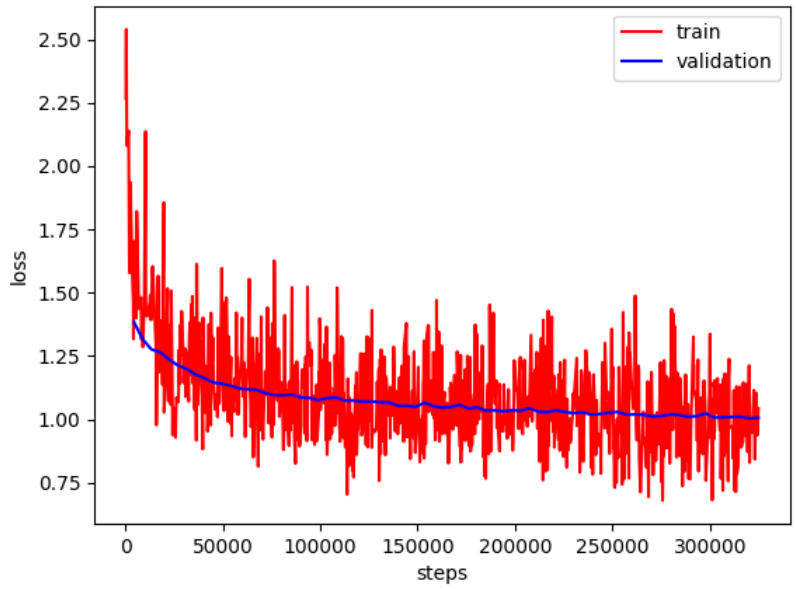

Figure 4.4: Loss profiles for training and validation

The fig 4.4 shows both the training and validation loss profile. As expected the losses decrease with training and get saturated eventually. The training loss curve and validation loss curve almost overlap, indicating no overfitting.

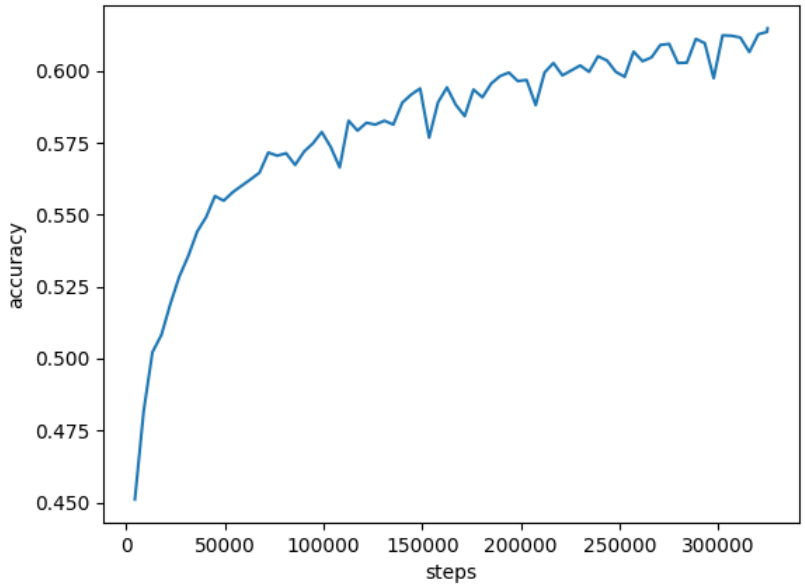

Figure 4.5: Accuracy profile

Fig 4.5 shows the accuracy profile. With training, accuracy also shows a steady increase

and it gets saturated eventually. The maximum accuracy obtained here is 0.63 which is a good score compared to 0.20 accuracy (since there are 4 classes) from random guessing.

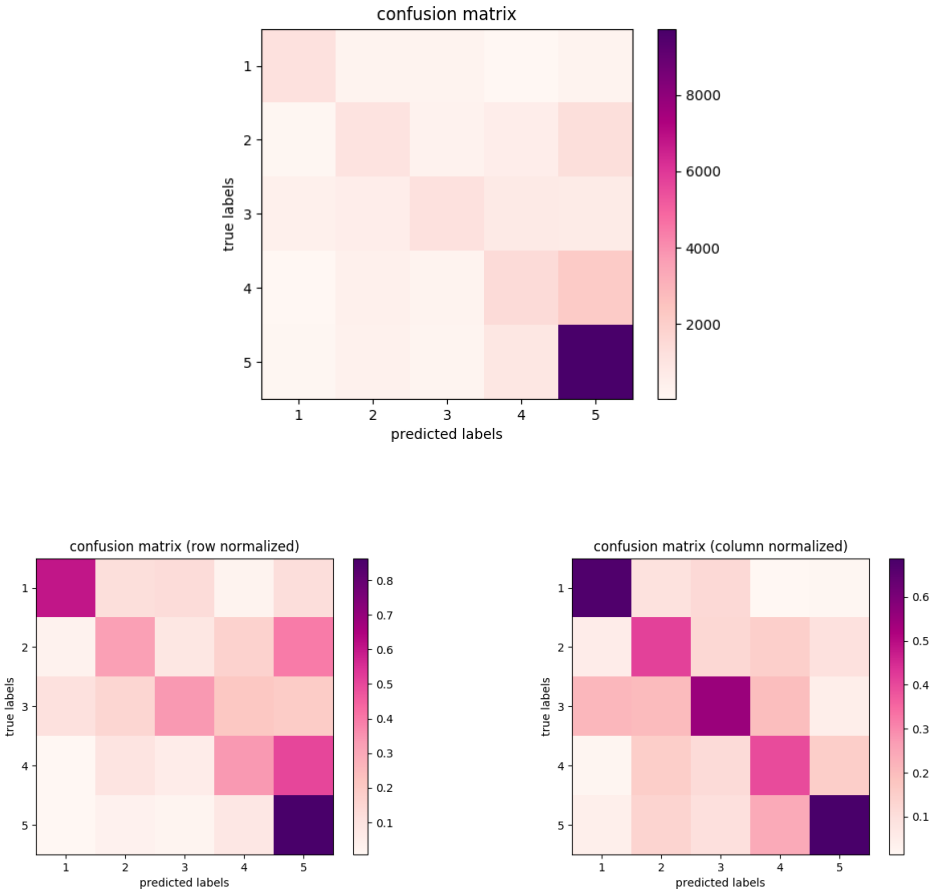

Figure 4.6: Confusion matrix, row-normalized and column normalized confusion matrices

Looking at the confusion matrices gives us a better understanding of network performance. The confusion matrix clearly states that the network is biased towards class 5 as speculated before. But other than that, it is fairly diagonal. The row normalized and column normalized confusion matrices also describe the same picture. In this sort of experiment, we are more concerned about purity and hence the column normalized matrix carries more significance. From the fig 4.6, we can see that the column normalized matrix is fairly diagonal as well. If we look at the efficiency, we see a similar story but this time the effect of class imbalance becomes prominent.

4.5 Can we improve the network performance

From the confusion matrices, it is clear that in order to improve the network performance we need to address the class imbalance issue. The most common solution to such class imbalance issue is trying oversampling or undersampling. In over sampling, one populates the training sample with repeated examples from the classes with low count while in undersampling, one populates the training sample with a subset of the examples with high count. Both of these two methods have its pros and cons and hence can be chose over one another based on the situation. But in our case, there is another way of establishing the class balance which is to use a *particle canon* sample.

The particle canon is a detector based simulation, i.e., we don't really simulate the collisions here. Instead we directly inserts the particles inside the detector arrays with the desired momentum, energy and initial positions and the tracks are recorded. This gives us the freedom to set the distribution of particles beforehand and thus achieving class balance becomes possible.

Chapter 5

Multilabel Approach

In the previous chapter, we saw that CNNs can be used to classify the charged current events and with proper implementation on a well balanced dataset, the network performance can be really promising. The main hurdle here was the imbalance in the dataset and particle canon was the solution to it. But due to technical issues, getting a sample from the particle canon was delayed and hence, in the mean time, we decided to investigate a multilabel approach since it gives us more information about the final state particle numbers. We also tried out a comparative study between two CNN architectures in order to come up with a better strategy.

5.1 Defining the multilabel problem

So far we have been dealing with a *multiclass* problem. In multiclass classification, we assume that each example is assigned to one and only one label which will belong to one of the predefined, mutually exclusive classes. But in a *multilabel* approach, every example has a set of target labels, and each label will have its own classes. The labels are usually more or less related and hence a shared network becomes more appropriate to tackle them, rather than tackling each of the label separately with separate networks.

We considered four labels mainly based on their relevance in the physical picture. The labels are -

- number of charged kaons
- number of charged pions
- number of protons
- number of *other* particles (all the hadrons other than the above three)

Each of the labels had four classes, namely -

- class 1: 0 particle events
- class 2: 1 particle events
- class 3: 2 particles events
- class 4: 3 or more particles events

5.2 Class balance

Fig 5.1 shows the class balance for each of the labels. Here we used the same data set as before and the class imbalance issue is quite evident from the plots. For charged kaons, almost 95% of the examples belong to class 1, and training a network with this data will surely lead to bias towards class 1. Similarly, for the last label we don't have any example belonging to class 1. For charged pions and protons, the situation is slightly better but they still suffer from class imbalance. Proceeding to training with this dataset will not clearly lead to good prediction model. Moreover, oversampling or undersampling is not an option anymore to fix the class balance. Since it is a multilabel problem, restoring the class balance for one label using any of the two methods will disturb the class balance for the other labels.

Even though it was not the most ideal step, we decided to proceed with this dataset. The plan was to get the frameworks ready and we can start training as soon as the particle canon

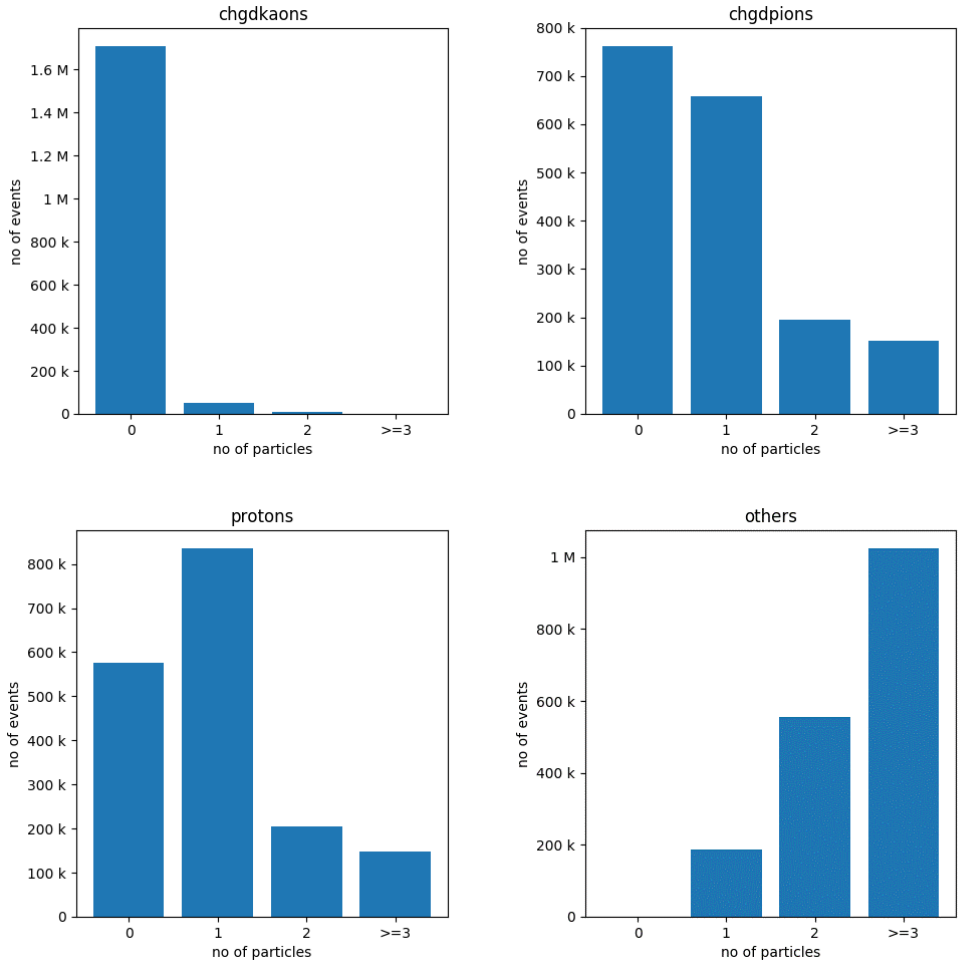

Figure 5.1: Class balance

sample is ready. But unfortunately it was not possible to get the sample in time and using a well balanced dataset for training the network remained as a future work.

5.3 The network architectures

We trained two different network architectures for this problem. The first one is an extension of the network used in the multiclass problem. In order to accommodate multiple labels, the dense section of the network was split into four part. The last layer of each of the section had softmax activation as used before. The loss was defined as the sum of all the individual losses, and the optimizer tried to minimize this total loss. Fig 5.2 shows the complete network.

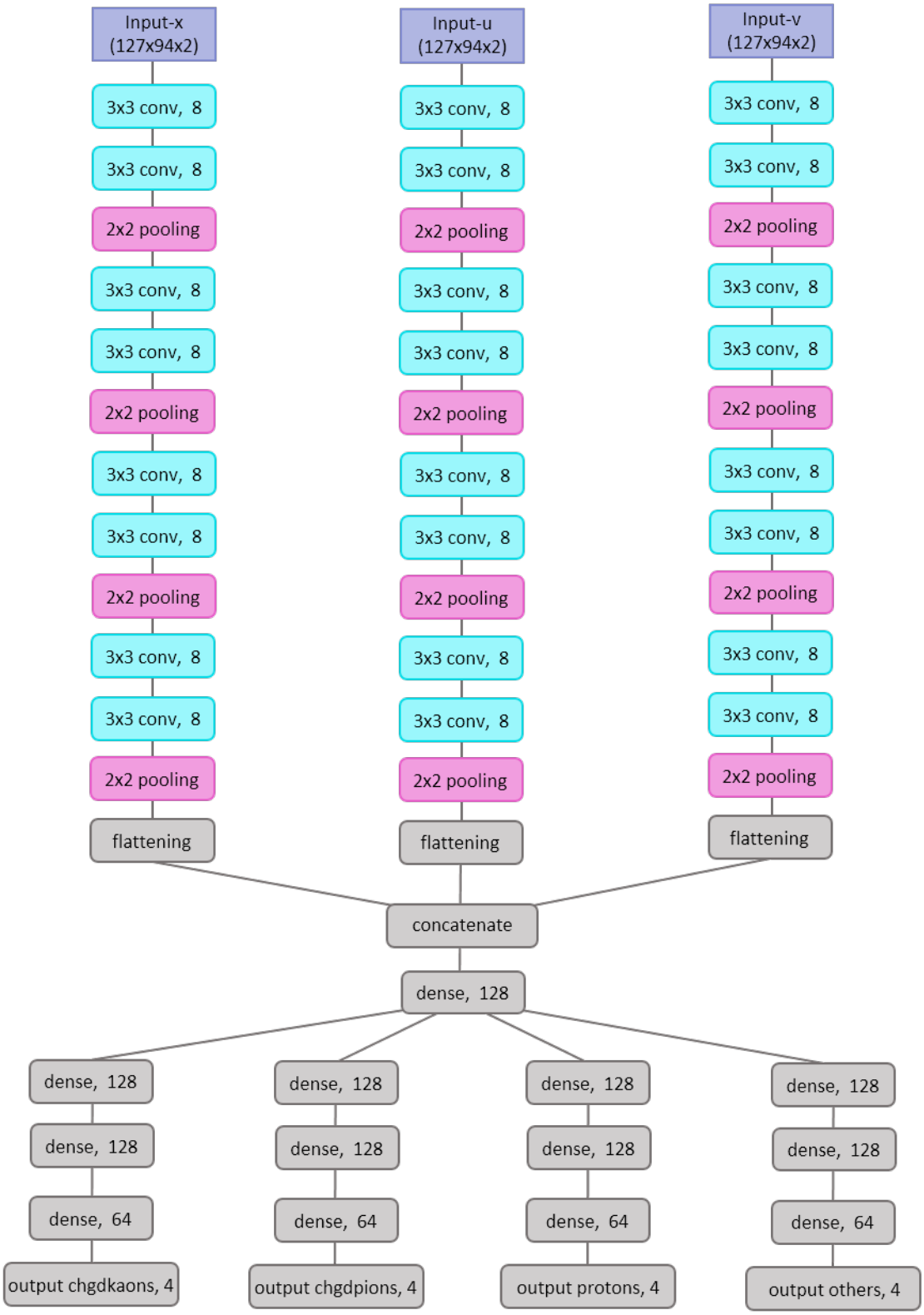

Figure 5.2: the multilabel network (VGGNet)

5.3.1 The ResNet architecture

Along with the previous network, we also trained a ResNet architecture for a comparative study. ResNets refers to residual networks. It was first published in 2016 by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun as an attempt to address the difficulties in training deeper networks by introducing residual connections or shortcuts. [HZRS15]

Deeper networks usually perform better as they can grasp higher level of abstraction and generalize richer structures in the underlying distribution. So, in theory, adding more and more layer should improve network performance. Even if it suffers from overfitting, we would expect the training loss to reduce as new layers are added. But in practice, the loss seems to saturate or increase once the network depth exceeds some limit. This decline in performance is due to the vanishing/exploding gradient problem. In deeper networks, as the gradient is backpropagated, the repeated multiplications make it really small (or large) and hence weight updates in the earlier layers are either almost negligible or really large leading to saturation or degradation in performance.

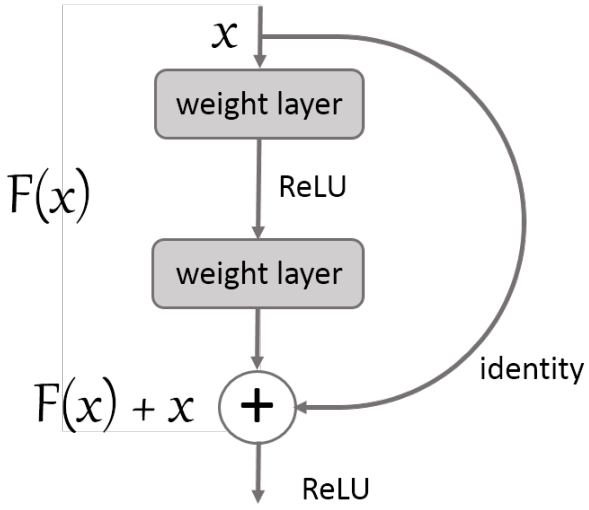

Figure 5.3: Residual block

ResNet tries to solve this problem using the 'identity shortcut connections'. This shortcut connects the input to the output of a few layers. Thus changing the output of the layers from $\mathbf{F}(\mathbf{x})$ to $\mathbf{H}(\mathbf{x})$ where $\mathbf{H}(\mathbf{x})$ can be written as $\mathbf{H}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{x}$. Fig 5.3 shows a residual block, here $\mathbf{x}$ is the input. The residual blocks solve the gradient issue. Even if the weight

layers have vanishing gradient, the identity connections help the flow of the gradient to the earlier layers.

Another way of looking at it is that the residues make the identity mapping easier to learn. When the network encounters an extra layer which may lead to problems described above we would expect the network to set it to identity, i.e. $\mathbf{F}(\mathbf{x}) = \mathbf{x}$. But adding the residue changes the identity map to $\mathbf{H}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{x} = \mathbf{x}$ or $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. It turns out that the second identity map is much easier for the network to learn than the first one.

5.3.2 Getting rid of the FC layers

Along with introducing the residual blocks, we also got rid of the fully connected layers in our network. The motivation here is that convolutional layers give us an intuitive picture where it extracts a feature map from the input. Thus stacking up the convolutional layers helps us extracting more and more abstract feature maps which are important for the classification. But the fully connected layers are still some sort of black boxes to us. We have some idea about how they work but it is not as clear as the convolutional layers. So, we got rid of the FC layers and made the network in such a way that the feature maps can directly be connected to the output classes. In order to achieve that we used global average pooling. [LCY14] For that we changed the number of channels in the second last layer to number of classes, then we take the average of the feature maps and after passing through a softmax activation it becomes the output probability. Fig 5.4 shows the modified ResNet architecture that we used.

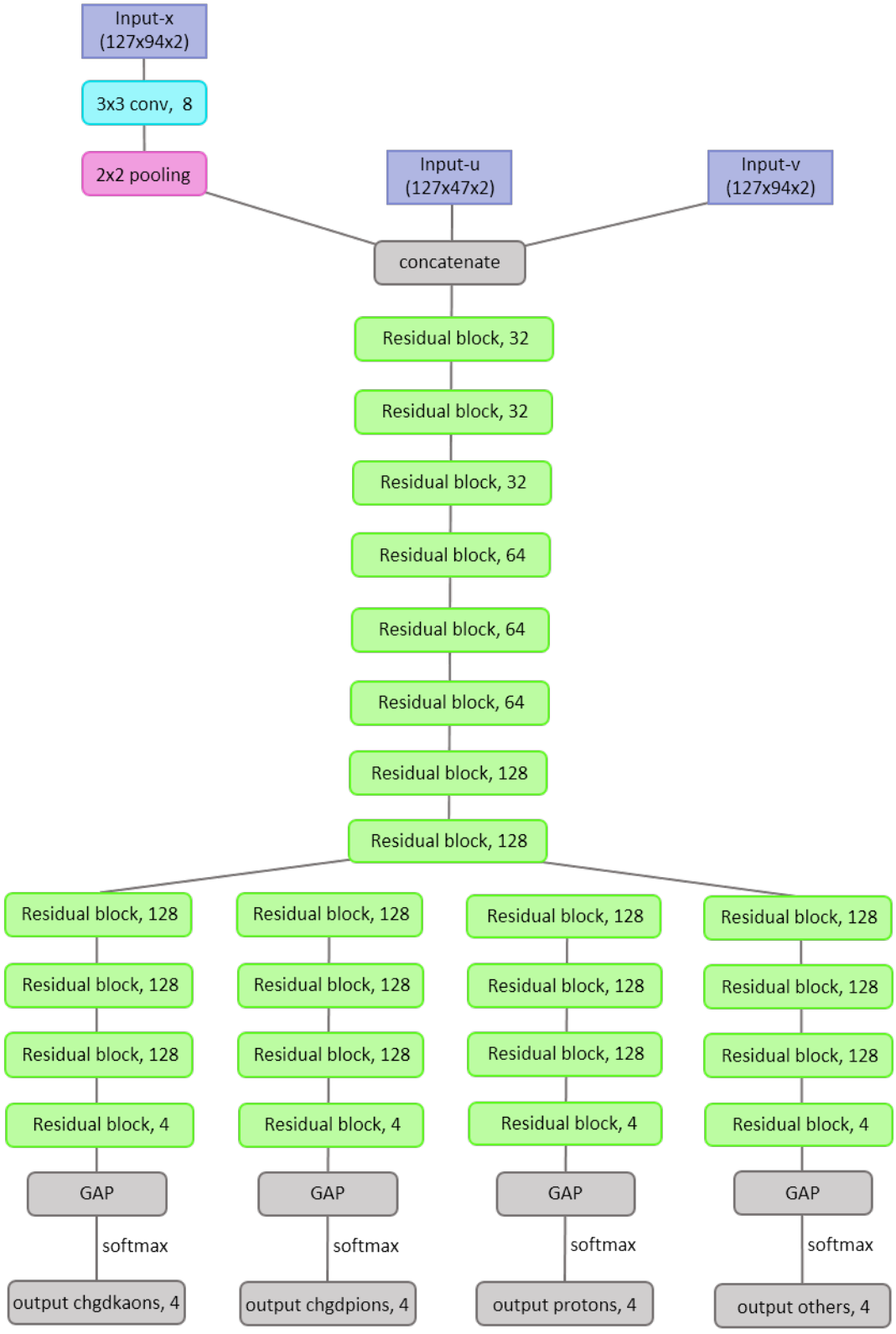

Figure 5.4: multilabel network (ResNet)

5.4 Network performance

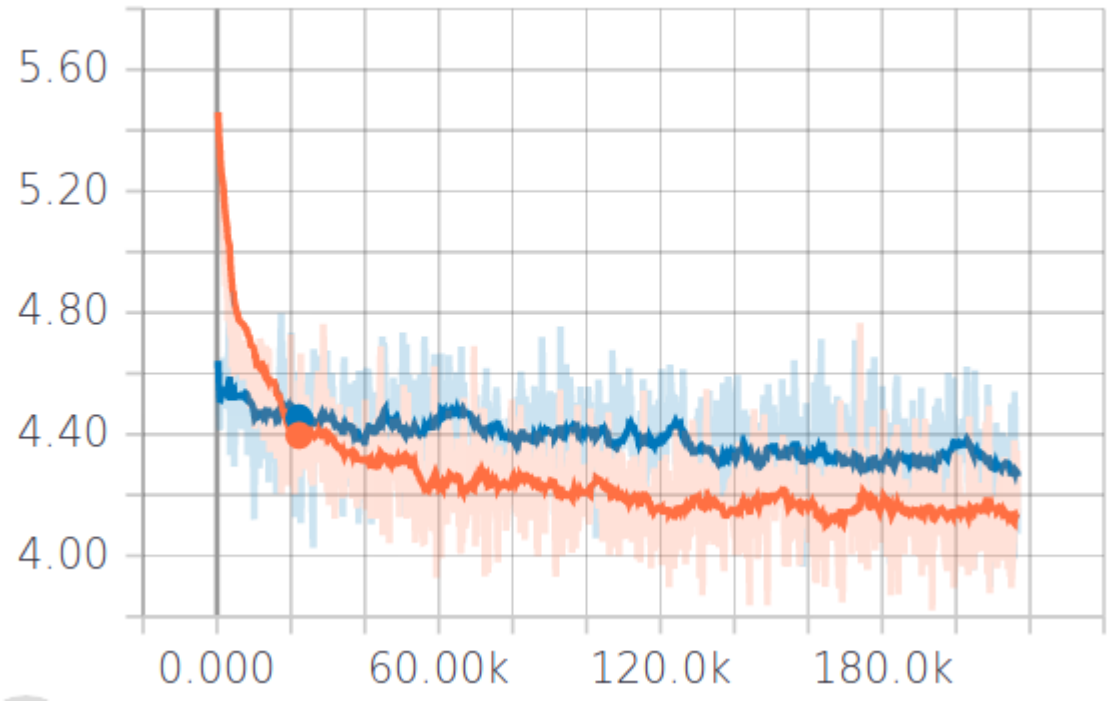

Figure 5.5: Loss profiles for VGGNet (in blue) and ResNet (in red)

Fig 5.5 shows the loss profile for both the networks. We can see the that the total loss decreases in both cases, but ResNet seems to achieve better loss value than the VGGNet. Though the profiles seem fine, the networks didn't really learn much and we can see it from the loss values.

It becomes clear when we look at the confusion matrices. Fig 5.6 shows the column normalized confusion matrices for both the VGGNet and ResNet. (We are considering only the column normalized confusion matrices since we are more interested in the purity of the predictions.) For number of charged kaon label, both the network predict class 1 100% of the time. We could expect such bias since 96% of the examples belong to class 1 as we have seen before. The other labels, though not as severe, show similar trend as well. Thus, it is fair to conclude that both the networks failed to generalize the problem. But if we compare the matrices we got from VGGNet and ResNet, we see that diagonal dominance is more prominent in the ResNet confusion matrices. This seems to be in accordance with the trend in the loss profile leading to the conclusion that, though not significantly, the ResNet outperformed the VGGNet.

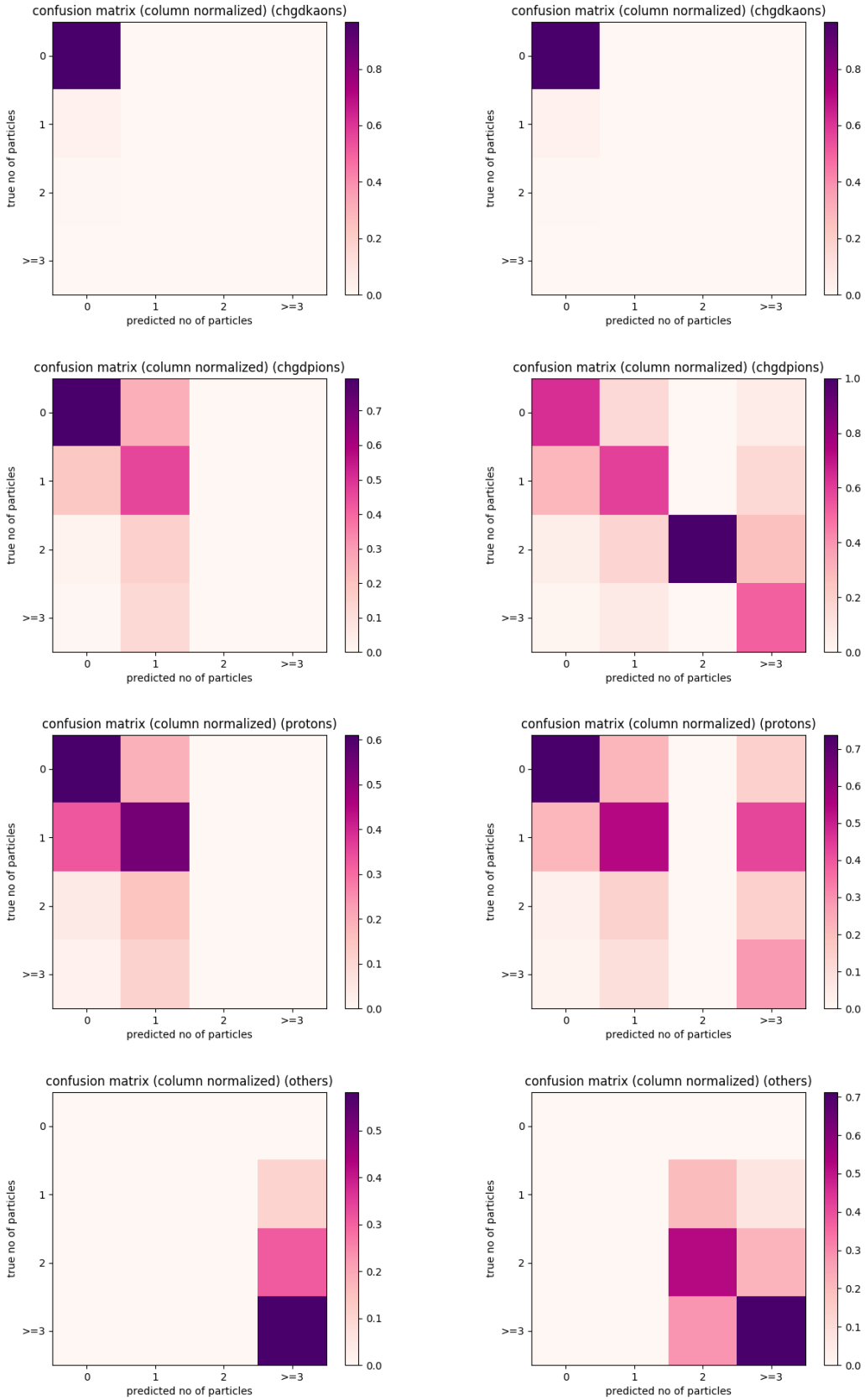

Figure 5.6: Column normalized confusion matrices for chagrd kaons, charged pions, protons and others respectively. The left column is for prediction with VGGNet and the right one is with ResNet

Chapter 6

Conclusion

The goal of the project was to use artificial neural networks to classify the charged current events. We took two approaches in order to achieve that goal - the multiclass approach and the multilabel approach. In this chapter, we summarize the findings and also discuss what measures can be taken in order to address the problems that we faced.

6.1 Multiclass approach

In the multiclass approach, we considered the following classes -

- Class 1 : (0 proton, 0 pion, 1 muon) events
- Class 2 : (0 proton, 1 pion, 1 muon) events
- Class 3 : (1 proton, 0 pion, 1 muon) events
- Class 4 : (1 proton, 1 pion, 1 muon) events
- Class 5 : everything else

Training a VGGNet gave us almost 63% accuracy. The confusion matrices were fairly diagonal. training dataset contained more number of class 5 examples and it made the network somewhat biased.

6.2 Multilabel approach

Here we considered four labels -

- number of charged kaons
- number of charged pions
- number of protons
- number of *other* particles (all the hadrons other than the above three)

and each of the labels had four classes, namely -

- class 1: 0 particle events
- class 2: 1 particle events
- class 3: 2 particles events
- class 4: 3 or more particles events

We trained two networks for the multilabel predictions - a VGGNet and a ResNet. In both the cases the network failed to understand the underlying distribution and showed poor performance. The obvious reason behind this is again the extreme class imbalance. Even though both the network showed poor performance, the more intuitive network, ResNet seem to reach a lower loss value. The evidences are not strong enough to support any claim that ResNet will perform better than VGGNet in our problem, but it's definitely worth investing.

6.3 Future perspective

In future our main focus will be on improving the multilabel approach, since it encompasses the multiclass approach as well. We can summarize the future aspects of the work as following-

- In the multiclass problem, the class balance issue can be fixed with oversampling or undersampling.

- The class balance issue is more serious in the multilabel approach, and oversampling or undersampling won't help. We clearly need a new dataset for it to work properly. A particle canon sample will definitely help in both the cases.
- Other than fixing the imbalance, we can also try to come up with new loss function where we punish the mispredictions more in case of the imbalanced classes. This will prevent the network from predicting the same wrong class again and again.
- Once the data balance issue is taken care of, exploring the hyper parameter space will help in improving the performance further.

Bibliography

[A+14] L. Aliaga et al., *Design, Calibration, and Performance of the MINERvA Detector*, Nucl. Instrum. Meth. **A743** (2014), 130–159.

[CW17] Qiming Chen and Ren Wu, *CNN is all you need*, CoRR **abs/1712.09662** (2017).

[F+98] Y. Fukuda et al., *Evidence for oscillation of atmospheric neutrinos*, Phys. Rev. Lett. **81** (1998), 1562–1567.

[fun]

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016.

[GCW18] Dan Guest, Kyle Cranmer, and Daniel Whiteson, *Deep Learning and its Application to LHC Physics*, Ann. Rev. Nucl. Part. Sci. **68** (2018), 161–181.

[Hay06] Simon Haykin, *Perceptron*, Encyclopedia of Cognitive Science (2006).

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, CoRR **abs/1512.03385** (2015).

[IS15] Sergey Ioffe and Christian Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, CoRR **abs/1502.03167** (2015).

[LBBH01] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-based learning applied to document recognition*, pp. 306–351, IEEE Press, 2001 (English (US)).

[LCY14] Min Lin, Qiang Chen, and Shuicheng Yan, *Network in network*, CoRR **abs/1312.4400** (2014).

[LS16] Shiyu Liang and R. Srikant, *Why deep neural networks?*, CoRR **abs/1610.04161** (2016).

[LZCS14] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola, *Efficient mini-batch training for stochastic optimization*, Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '14, ACM, 2014, pp. 661–670.

[min] *Why study neutrinos?*

[neu]

[NIGM18] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, *Activation functions: Comparison of trends in practice and research for deep learning*, CoRR **abs/1811.03378** (2018).

[QYLC18] Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen, *How convolutional neural network see the world - A survey of convolutional neural network visualization methods*, CoRR **abs/1804.11191** (2018).

[Rud16] Sebastian Ruder, *An overview of gradient descent optimization algorithms*, CoRR **abs/1609.04747** (2016).

[RZL17] Prajit Ramachandran, Barret Zoph, and Quoc V. Le, *Searching for activation functions*, CoRR **abs/1710.05941** (2017).

[S+16] James Strait et al., *Long-Baseline Neutrino Facility (LBNF) and Deep Underground Neutrino Experiment (DUNE)*.

[SHK+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958.

[SM16] Ian M. Shoemaker and Kohta Murase, *Probing BSM Neutrino Physics with Flavor and Spectral Distortions: Prospects for Future High-Energy Neutrino Telescopes*, Phys. Rev. **D93** (2016), no. 8, 085004.

[SZ15] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[Wil94] R. Jeffrey Wilkes, *Evidence for neutrino oscillation from Super-Kamiokande*, Results and perspectives in particle physics. Proceedings, Les Rencontres de Physique de la Vallee d'Aoste, La Thuile, Italy, March 1 - 7, 1998, 1994, pp. pp.73–91.