Recovery of merged π^0's from ECL images of the Belle II detector using CNN

Naveen Kumar Baghel

(MS18102)

A dissertation submitted for the partial fulfillment of BS-MS dual degree in Science

Indian Institute of Science Education and Research, Mohali

May 1, 2023

Certificate of Examination

This is to certify that the dissertation titled **Recovery of merged π^0s from ECL images of the Belle II detector using CNN** submitted by **Naveen Kumar Baghel** (Reg. No. MS18102) for the partial fulfillment of BS-MS Dual Degree programme 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. Ambresh Shivaji

Dr. Satyajit Jena

Dr. Vishal Bhardwaj
(Supervisor)

Dated: May 1, 2023

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Vishal Bhardwaj 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 acknowledgment of collaborative research and discussions. This thesis is a bonafide record of my original work done by me, and all sources listed within have been detailed in the bibliography.

Naveen Kumar Baghel
(Candidate)
Dated: May 1, 2023

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. Vishal Bhardwaj
(Supervisor)
Dated: May 1, 2023

Acknowledgements

As I contemplate my academic journey, I am overcome with a profound sense of gratitude for the unwavering support and guidance of those who have accompanied me along the way. The completion of this thesis represents a pivotal juncture in both my academic and personal journey. It is the culmination of unrelenting diligence, assiduity, and the invaluable aid and guidance of the many individuals who have played a crucial role in my intellectual and personal growth. Without their contributions, I would not have been able to achieve this milestone, and for that, I extend my sincere gratitude to all those who have come across closely or distantly for being the guiding beacon in my life's odyssey.

Among the many individuals who have contributed to this thesis, First and foremost, I would like to express my sincere appreciation and thankfulness to my supervisor, Dr. Vishal Bhardwaj, who was the driving force behind this thesis. His profound expertise and competence in the field, combined with his invaluable advice and astute critical thinking, greatly influenced my thoughts and the caliber of my work. He truly was the think tank of this project, and I owe him a great debt of gratitude. As I contemplate the privilege of working under my supervisor, I wonder at my extraordinary good fortune. He gave me complete autonomy in this project, and any form of pressure never beset me throughout its completion. His unwavering support in addressing difficulties encountered along the way and fostering a scientific mindset has enriched my professional development and personal growth. I sincerely thank him for developing me on the personal and professional front.

In addition to my advisor, I wish to extend my heartfelt gratitude to the other thesis committee members, namely Dr. Ambresh Shivaji and Dr. Satyajit Jena. Their unwavering support, valuable feedback, and astute insights were instrumental in ensuring the successful completion of this thesis.

The Belle II Experiment is at the core of this project. As a member of the Belle II collaboration, I am deeply grateful for this vibrant community's invaluable support and contributions. The tools and resources provided by the collaboration were instrumental in accomplishing this thesis. Furthermore, I am thankful for the chance to interact with the diverse individuals in the Belle II community. Their guidance and assistance, as well as their willingness to share their knowledge and expertise, proved invaluable throughout this endeavor.

In addition to the Belle II collaboration, I would like to thank my affiliated institute IISER Mohali for providing me with an excellent platform to pursue science as a career. The institute has provided me with state-of-the-art infrastructure and resources, which have

been crucial in my research endeavors. Furthermore, the indescribable nature of the freedom provided by my affiliated institute played a crucial role in shaping my life. Apart from academics, the institute has also provided me with a vibrant and supportive community of friends who have been an essential part of my journey. The institute's sports facilities have kept me physically active, rejuvenated my mind, and allowed me to form lasting relationships with fellow enthusiasts. All in all, my time at this institute has been a transformative experience, and I appreciate the chance to be a member of such a distinguished institution.

Also, I express my gratitude to the esteemed REDD lab members, including Dr. Manish, Neetesh, Sourav, Latika, and Ipsita. Their unwavering assistance in various aspects of my project was truly invaluable. Additionally, their constant encouragement and the congenial ambiance they fostered within the lab were powerful motivations for me to derive enjoyment from my work.

Finally, I want to convey my sincere appreciation to my family, who have consistently been a reliable source of love and inspiration. They have stood by me through all the ups and downs, and their faith in me has been a driving force, especially during the most trying times. In addition, I would like to give a special shout-out to my close companions Rohit Negi, Hemant, and Sarita, who have been an indispensable part of my life. Their constant encouragement, support, and friendship have provided me with an inexhaustible source of motivation. They have always kept me grounded and reminded me of the significance of maintaining a healthy work-life balance by pushing me to pursue my passions and ensuring that I make time for leisure activities. I will always cherish their unwavering support and encouragement.

Abstract

This study aims to utilize a Convolutional Neural Network (CNN) to retrieve merged π^0 mesons lost in the Belle II experiment, where they appear as individual photons. The issue arises when dealing with high momentum π^0 mesons, i.e., beyond 2 GeV in the Belle II experiment, as the shower produced by both the π^0 meson and gamma appear indistinguishable at the Electromagnetic Calorimeter (ECL) detector. Currently, reconstruction software is utilized to match photon pairs created by the $\pi^0 \to \gamma\gamma$ decay; however, the efficiency of this process can be affected by the γ produced by the rest of the events (ROEs), which mimic the signal. One of the most challenging tasks in particle physics research is accurately identifying and reconstructing subatomic particles. By the nature of the problem and its importance, accurate reconstruction of π^0 mesons is crucial for identifying various B/D meson decays, including rare decays like $D^0 \to \gamma\gamma\left(8.5 \times 10^{-7}\right), D^0 \to \rho^0\gamma\left(10^{-5}\right)$, and $D^0 \to \phi\gamma\left(10^{-5}\right)$. These rare decays have dominant background arising from decays like $D^0 \to K_s\pi^0\left(1.24 \times 10^{-2}\right), D^0 \to \pi^0\pi^0\left(8.26 \times 10^{-4}\right)$, and $D^0 \to \phi\pi^0\left(1.17 \times 10^{-3}\right)$.

The Convolutional Neural Networks performed reasonably well on a test dataset, which is identical to real scenarios, achieving an area under the curve (AUC) of 0.86 for the Precision-Recall curve. These results demonstrate the potential of machine learning (ML) algorithms and highlight areas for improvement in the current work to enhance the efficiency of identifying π^0 particles with energy deposits in the ECL. The findings suggest that the 'raw' ECL images contain much more information than currently used expert-engineered features.

Disclaimer:- This master thesis solely focuses on the barrel region of the ECL detector within the Belle II Experiment. Furthermore, all the results presented in this work are based on Monte Carlo simulations.

List of Figures

List of Tables

List of Abbreviations & Notations

γ	Photon
π^0	Neutral Pion
AUC	Area Under Curve
basf2	Belle II Analysis & Software Framework
CNNs	Convolution Neural Networks
CP	Charge-Parity
ECL	Electromagnetic Calorimeter
FCNC	Flavor Changing Neutral Current
ML	Machine Learning
PDFs	Probability Density Functions
ROC	Receiver Operating Characteristic Curve
SM	Standard Model

Contents

Chapter 1

Introduction

Science emerged as a human endeavor to systematically understand and explore the natural world. From ancient times, humans have sought to understand the world around them. This quest for knowledge has led to the development of various scientific disciplines, one of them being particle physics.

Particle physics aims to understand the universe at the subatomic level through the study of elementary particles and their interactions. To the extent of our current level of exploration, the fundamental structure of matter and forces are encapsulated in the theoretical framework known as the Standard Model of particle physics. Particle Physicists try to test and predict the accuracy of the SM predictions experimentally through three primary approaches characterized as cosmic, energy, and intensity frontier. Through these approaches, they also tried to answer the questions unanswered within the standard Model framework, such as "why is there an observed matter-antimatter symmetry in the universe?" Although the CP violation observed in the quark sector is insufficient to explain the prevalence of matter in the universe, it suggests that undiscovered sources of CP violation may exist beyond what is currently understood.

In order to push the boundaries of the current SM and delve into the realm of physics beyond the SM, it is imperative to persist with the aforementioned experimental approaches. Among the numerous ongoing experiments in particle physics, The Belle II Experiment emerges as a prominent contender operating at the intensity frontier. The goal of this experiment is to search for new physics beyond the SM, and one of the ways is by exploring the rare decays with unprecedented precision. In experiments like Belle II, understanding the underlying physics principles governing particle interactions requires a stage known as particle identification, a fundamental step in performing physics analyses. This method entails using sub-detectors at various levels to identify the stable particles produced during the experiment, which are then used to reconstruct intermediate particles involved in the events. Chapter 3 provides a comprehensive overview of particle physics experiments, with

a particular focus on the Belle II experiment.

This thesis addresses a challenging problem in particle identification, which involves recognizing high momentum π^0 clusters that imitate γ clusters in the Electromagnetic Calorimeter at the Belle II experiment. The work focuses on using the neural network known as CNNs. CNNs possess unique features, including automatic feature extraction, parameter sharing, and translation invariance, making them a potent tool for analyzing large datasets. They excel in tasks such as image recognition, where specific features may only appear in some areas of an image. The network can also capture local connectivity in the data, providing spatial relationships between neighboring pixels. Its implementation and architecture are discussed in more detail in Chapter 5.

Currently, neutral particle identification of particles like γ and neutral hadrons is made using the concept of the likelihood function, i.e., based on the kinematics, shower shape variables, and timing information, which is carefully calibrated using known particles and continually refined to improve the accuracy of the PID step. For the high momentum π^0 mesons in the decay mode $\pi^0 \to \gamma\gamma$, i.e., energies above about 2.5 GeV, e.g., from $B \to \pi^0\pi^0$, identification relies on multivariate classifiers and the π^0 reconstruction is based on the combination two photon candidates. But in the case of high energies, as given above, the two photon-induced showers do not have separate maxima and are often reconstructed as one photon candidate [9]. A more detailed explanation is given in Chapter 2.

The work in this thesis is done to improve the identification of above mentioned high momentum π^0 just by taking raw energy information stored in the ECL crystals of the Belle II detector with the help of a neural network rather using expert-engineered features like showers second-moment shape variable in the case of π^0.

Starting from Chapter 2, the physics motivation and a brief overview of how this problem could be approached through CNNs are discussed. Chapter 3 discusses the Belle II experiment, its sub-detector components, and, more importantly, ECL. Then in the central part of the thesis, i.e., in Chapter 4, which discusses data generation through Monte-Carlo simulation and its selection using modules of the `basf2` and `ROOT` software. After that, in Chapter 6, the behavior of our model is checked under different setups, and finally, the performance of the model through simulated data with beam background collected only in the Barrel region of the ECL detector is measured and also compared its performance with `basf2` by the help of a toy model discussed in the same section. Chapter 7 presents the ultimate outcomes of the research and explores the broader implications of the findings. Additionally, the chapter discusses potential enhancements to the research in various areas related to the Belle II experiment.

Chapter 2

Physics Overview & Motivation

Through particle physics, we have made significant progress in gaining a basic understanding of our surroundings with the help of the Standard Model (SM). The SM is considered the best-tested theory of nature at a fundamental level due to its current level of experimental precision and the energies it has reached so far.

2.1 The Standard Model

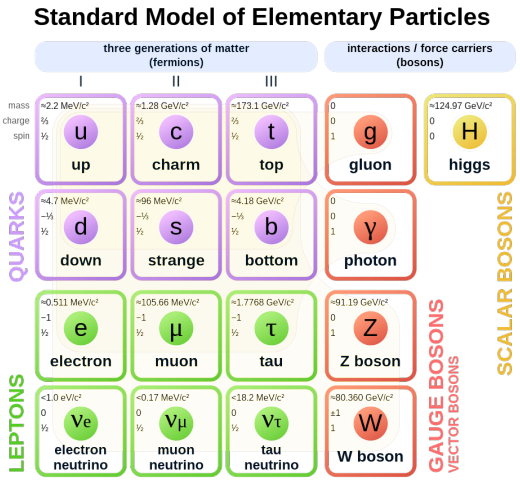
A well-known quantum theory called the Standard Model (SM) interprets the behavior of subatomic particles by considering them to be excited quantum fields. Fermions and bosons are the two types of particles that are categorized by the SM. The building components of matter, known as fermions, have half-integer spin and adhere to the Fermi-Dirac statistics. The interactions between fermions are mediated by bosons, which have integer spin and adhere to the Bose-Einstein statistics. In accordance with the Standard Model (SM), there are three basic forces[1]:

1. Electromagnetic force:- carried by photons(γ)
2. Weak force:- carried by the Z, W^+, and W^- bosons
3. Strong force:- carried by gluons.

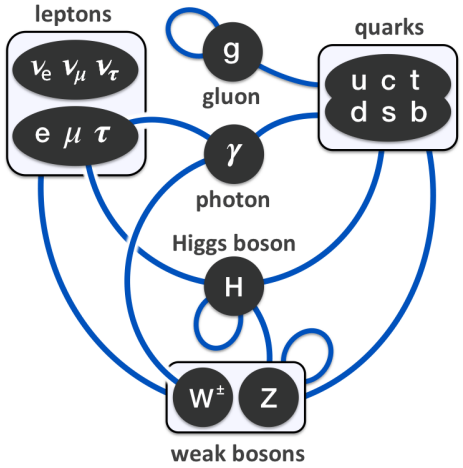
Finally, the most recent discovery includes Higgs boson into SM, which gives mass to elementary particles through the Higgs mechanism.
The fermions can further be classified into leptons (ℓ) and quarks (q). Quarks are non-integer charge charged particles that can only exist in bound states known as hadrons (h). Quarks have two different types of known bound states: mesons, which are restricted states

[1]The reconciliation of gravity with quantum mechanics is still pending and it is not currently regarded as a component of the SM.

of one quark and one antiquark, and baryons, which are confined states of three quarks. Quarks come in six different flavors (up (u), down (d), charm (c), strange (s), top (t), and bottom (b)), and they vary, for instance, in mass. Leptons can not feel a strong force, but quarks may interact through all the basic forces. The electron (e), muon (μ), and tau (τ) are three negatively charged leptons, whereas the neutrinos are three neutral leptons that are each associated with a charged lepton (ν).
The below-mentioned Figure 2.1 provides an overview of the fundamental properties of the particles in the SM and depicts the potential interactions that can occur among them.



(a) Standard Model of Elementary Particles



(b) Elementary particle interactions in the Standard Model.

Figure 2.1: The Standard Model: The SM consists of 12 fermions and 5 bosons. (a) shows the elementary particles grouped by their fundamental physical properties. The lines in the picture (b) represent possible interactions between the SM particles. Both figures adapted from [4, 11].

As illustrated in Figure 2.1, the elementary fermions comprise only three generations. The first generation comprises up and down quarks, electrons, and electron neutrinos, which are relatively long-lived and constitute a significant proportion of visible matter. The charm and strange quarks and top and bottom quarks form the second and third generations, respectively, with their corresponding leptons. These fermions have a larger mass and can decay into lighter particles via weak force.

The SM provides guidelines for permissible conversions and decays among particles through quantum number conservation laws, and the likelihood of a specific decay occurrence is provided by its branching ratio ($\mathscr{BR}$)[2]. Decays of elementary particles are of great

[2]Branching Ratio is the probability of a particular decay process occurring for a particle relative to all the possible decay modes available to that particle.

interest. Precise measurements of $\mathscr{BR}$ and particle lifetimes are valuable checks for SM and its conservation laws like CP. While doing so puts us on a stage of finding the new physics behind the SM.

2.2 *B* Physics

Flavor physics experiments primarily work on the intensity frontier emerged as a new sector in particle physics, aiming for the indirect search for dark matter and the mechanism and sources behind the CP violation. Exploring flavor physics offers an opportunity to gain knowledge about the characteristics of new physics at the TeV scale, which is not feasible through the direct production of particles at the CERN Large Hadron Collider (LHC) due to the influence of quantum effects, which enable virtual particles to modify the outcome of precise measurements in ways that reveal the underlying physics [6].

As precise measurements are crucial to uncovering new physics beyond the Standard Model, probing this *B* physics from the flavor sector has emerged as a significant area of study in particle physics, primarily focused on demonstrating the consistency of the CKM mechanism and exploring rare decays through precise measurements. Historical examples, such as beta decay, have shown that discoveries in rare processes can lead to a deeper understanding of nature. For ex., the observation of the W boson was ultimately predicted by beta decay, which provided insight into the electroweak mass scale.

B physics, also known as "beauty physics", mainly deals with the study of particles containing the bottom quark (also called beauty quark). The two main experiments which provide insights towards this are the KEKB collider for the Belle experiment[3] at KEK and the PEPII collider for the BaBar experiment at SLAC also known as B factories.

The two current *B* factories have contributed significantly to the confirmation of the quark mixing pattern proposed by CKM and the identification of new processes. Some recent results from these prove to be difficult to explain within the SM, such as the values of the angle ϕ_1 measured in some penguin process $b \to sq\bar{q}$ and the precisely measured value in $B \to J/\psi K_S^0$ differ by two to three standard deviations ($B^0 \to \pi^0\pi^0 K_S^0$, $B^0 \to K^+K^-K^0$) [3] and may suggest the existence of a new CP phase in this penguin process.

In search of other sources of CP-violating processes through the above experiments, Neutral *B* meson mixings are one of the most important FCNC processes in *B* physics. In the Standard Model, the $B_d - \bar{B}_d$ mixing involves the CKM matrix element V_{td} and thus gives a *CP* violating amplitude, which induces a variety of *CP*-violating observables through its quantum mechanical interference with other amplitudes. On the other side, in-

[3]This thesis work is based on the Super KEKB collider known as Belle II experiment, upgraded version of KEKB collider described in Chapter 3 .

direct CP-violation results from the quantum mechanical interference of the mixing and decay amplitudes. There are several ways to estimate the angles of the Unitarity Triangle, thanks to the mixing-induced asymmetry. So to prove this, measurement of $\sin 2\phi_1$ through various processes and decay modes involving many rare decays to a precision level is important e.g. At the hadron level, the corresponding modes are $B^0 \to \phi K_s$ and $B^0 \to \eta' K_s$, $B^0 \to K_S K_S K_S, B^0 \to \pi^0 K_S, B^0 \to \rho^0 K_S, B^0 \to \omega K_S$, etc. The CP phases can also be obtained through D meson intermediate states. e.g., The angle ϕ_3 can be obtained by measuring the interference among different decay amplitudes in $B \to D\pi$ and DK decays [3]. This can involve the study of many rare decays like $D^0 \to \gamma\gamma\left(8.5 \times 10^{-7}\right), D^0 \to \rho^0\gamma\left(10^{-5}\right), D^0 \to \phi\gamma\left(10^{-5}\right)$, etc. where the dominant background arises from $D^0 \to K_s\pi^0\left(1.24 \times 10^{-2}\right), D^0 \to \pi^0\pi^0\left(8.26 \times 10^{-4}\right), D^0 \to \phi\pi^0\left(1.17 \times 10^{-3}\right)$ this decays[4].

The above-provided examples of the rare decays which are crucial to CP measurements mainly involve π^0 and γ, and also approximately one-third of the B-decay products consist of π^0's or other neutral particles that generate photons across a broad energy spectrum ranging from 20 MeV to 4 GeV [2]. So to conduct various B physics analyses for the Belle II experiment, it is crucial to distinguish neutral hadrons such as π^0's that imitate γ in the range of high momentum[5].

2.3 π^0 Meson

The π^0 meson, known as the neutral pion, is one of particle physics's lightest and most fundamental particles. It is a neutral meson composed of two quarks - an up quark and an anti-up quark, or a down quark and an anti-down quark, bound together by the strong nuclear force. It has a mass of 135.0 MeV$/c^2$ and a mean lifetime of 8.5×10^{-17} sec [12].

The π^0 meson has a crucial role in strong nuclear interactions. It is produced during high-energy particle collisions, such as in B factories, where positrons and electrons are involved. Additionally, it is also associated with the decay of several other particles, including numerous hadrons like D^0, B^0, $\rho \to \pi^0\gamma$, and $\eta \to \pi^0\pi^0$, etc.

One of the unique features of the π^0 meson is its decay mode. The π^0 meson decays almost exclusively into two photons due to its spin and parity properties. It decays through the electromagnetic force, which explains why its mean lifespan is substantially shorter than the charged pion's (which can only decay via the weak force).

[4]The mentioned value in bracket is the branching fraction of the corresponding decays.

[5]The focus of this thesis work is on the energy range above 2.5 GeV/c, which is considered sufficient for studying the imitation of π^0 into single gamma in the Belle II experiment. [9]

2.3.1 Decay of π^0 into two gammas

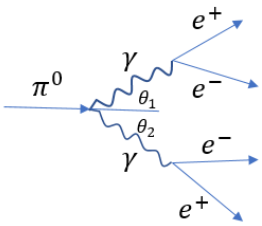


Figure 2.2: A general decay scheme of π^0 which has a branching ratio of 98%.

Let's take a given π^0 has a velocity v and mass m that goes into two gammas with angular separation θ, i.e., $\theta_1 + \theta_2$.
Rest Mass of a $\pi^0(m_{\pi^0})$ = 135 Mev/c^2
Rest Mass of a $\gamma = 0$
Therefore, Total Energy of a $\gamma = KE_\gamma$
From momentum conservation, in the x and y direction

$$P_{\pi^\circ} = P_{\gamma_1} \cos\theta_1 + P_{\gamma_2} \cos\theta_2 \tag{2.1}$$

$$P_{\gamma_1} \sin\theta_1 = P_{\gamma_2} \sin\theta_2 \tag{2.2}$$

From Energy Conservation,

$$E_{total(\pi^0)} = E_{\gamma_1} + E_{\gamma_2} \tag{2.3}$$

In natural units, where c = 1, the Energy-Mass relation is

$$E_{total}^2 = P^2 + m_0^2 \tag{2.4}$$

where m_0 is the rest mass.
For $\theta_1 = \theta_2$,
Upon Solving eq 2.1 & 2.2 and using eq 2.3 to substitute the result in eq 2.4, we get,

$$\cos\theta = \frac{P_{\pi^0}}{\sqrt{m_{\pi^0} + P_{\pi^0}^2}} \tag{2.5}$$

which gives the desired relation between angular separation and momentum of the pion(π^0). The above relation gives us the plot as shown in figure 2.2, and it clearly shows that for momentum above 2.5 Gev/c, the decay angle almost approaches less than 7°. So, in general, at high momentum range, both the gamma's of the π^0 appears to be merged, and therefore shower of π^0 looks like a shower of one gamma to a detector, and this is the domain of our problem in this thesis.

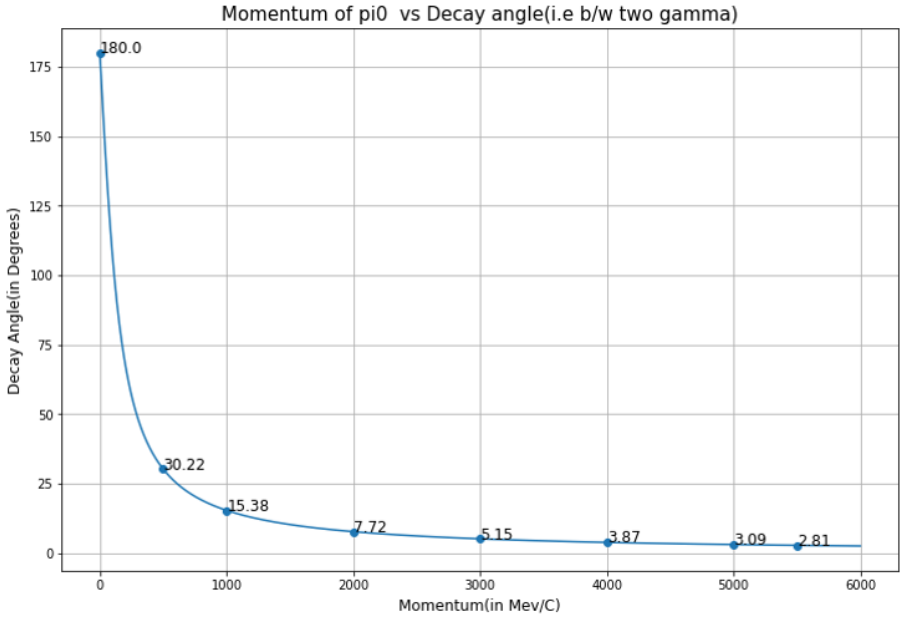


Figure 2.3: Plot between momentum of π^0 and its decay angle.

2.4 Particle Identification

Particle identification is a crucial component of High Energy Physics investigations in which the 4-momenta of secondary particles must be determined. Particle 3-momenta are typically acquired by detecting their trajectory deflection in a magnetic field. The mass, energy, or velocity is then calculated to get the value of the fourth component of the four-momentum. As it uniquely identifies the internal quantum numbers of particles based on their mass, this measurement is known as "particle identification."

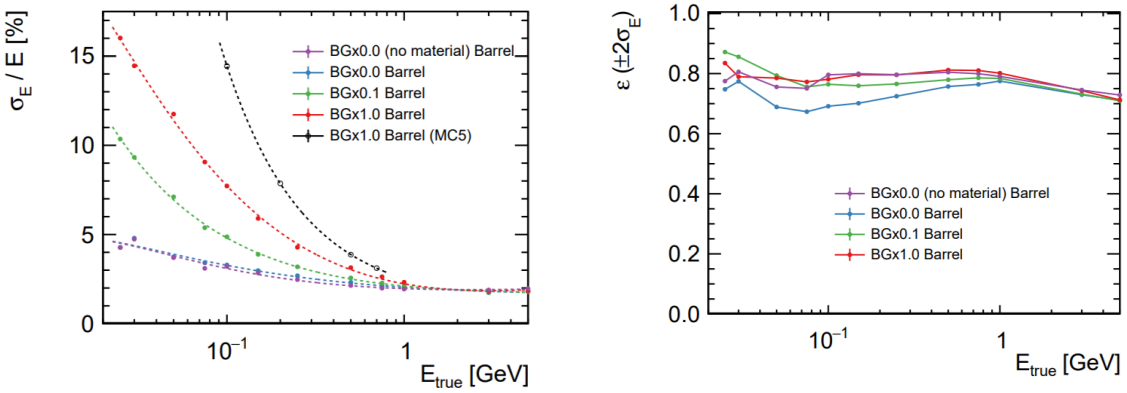
In a standard particle physics experiment like SuperKEKB or LHCb, particles are identified based on their identifiable signatures left in the detector [10]. These detectors are divided into a set of components where each component tests for a specific set of particle properties to get the 4-momenta of the particles produced in a collision with the help of different methods like tracking, time-of-flight, Cherenkov radiation, and calorimetry.

This thesis study involves the identification of π^0 and γ, neutral particles that do not leave visible traces or tracks in a detector. Detecting these particles relies solely on measuring their energy, achieved using calorimetry.

2.4.1 In the Belle II Experiment

The Belle experiment utilizes the Electromagnetic Calorimeter (ECL) to achieve optimal resolution and efficiency in reconstructing the energy and position of deposition from neutral and charged particles. Using the clustering code of release 00-05-03 (MC5) photon

energy resolution and photon reconstruction efficiency for different background levels in the Barrel region of ECL is shown in figure 2.4 [9].



(a) Photon energy resolution as a function of true photon energy.

(b) ECL reconstruction efficiency for single photons for different background levels.

Figure 2.4: Efficiency plots related to (a)photon energy (b) and its reconstruction in the barrel region of the ECL detector of the Belle II. Both figures adapted from [9].

Particle identification is carried out by analyzing shower shape variables and tracks matched to clusters, allowing for the identification of various particles such as electrons, muons, charged and neutral hadrons, and photons.

As for neutral particles in the Belle II experiment, the only source of information is ECL, which gives us energy and the hit position of a particle, and using a combination of these gives different variables like Zernike moments, Lateral moment(LAT), E9oE21, Highest energy crystal, Cluster Id, etc. [5]. These variables are used for the purpose of identification. In addition, the Belle II experiment uses a shower second-moment shape variable in the case of merged π^0 mesons, where two photons can not be separated into two different clusters.

Our Approach

Our approach involves directly identifying a π^0 shower that looks like a gamma shower by using energy deposited in the ECL crystals for each shower in the barrel region of ECL. Just by using raw energy information from the crystals and the position of the crystal that has the highest energy (known as seed crystal) for each affected crystal per event and took this energy as pixel intensity. So that we can have an image structure per shower per event to deploy the ML model, i.e., CNNs. A general layout of our method is shown in figure 2.5.

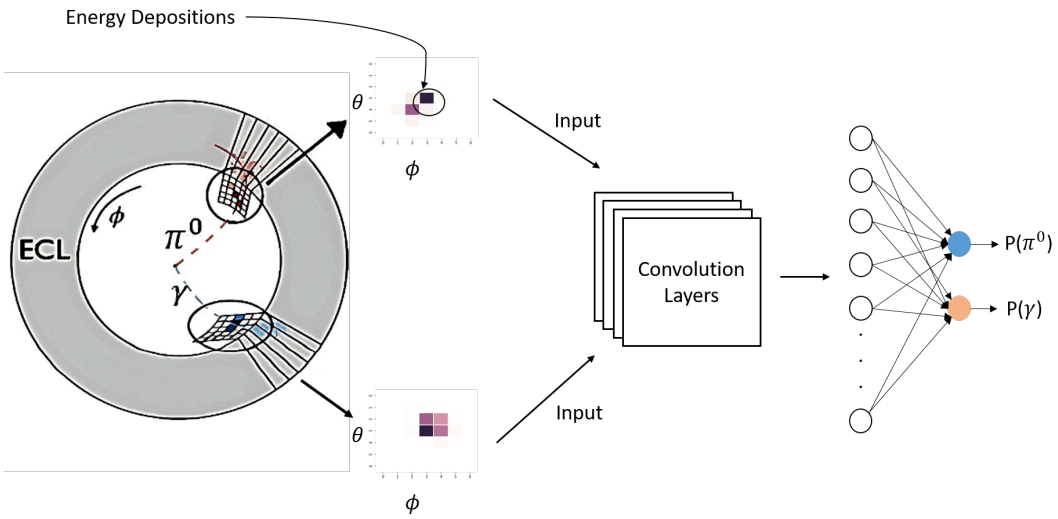


Figure 2.5: Schematic of how to deploy CNNs.

Note:- Our region of study is only the barrel area of the ECL detector, as endcaps are very irregular. Due to this, there is a chance that we can't get a uniform structure for energy deposition because of large leakage through the in-between gaps of ECL crystals. Also, we can't have the same resolution in energy measurements of photons (particles of interest in our case) compared to the barrel region used in our study.

Chapter 3

The Belle II Experiment

The Belle II experiment, as described, works primarily for the flavor sector of particle physics, performed with the help of the SuperKEKB accelerator at KEK in Tsukuba, Japan. SuperKEKB is a two-ring, asymmetric energy, e^+e^- collider. It accelerates asymmetric electron and positron beams with energies 7 GeV and 4 GeV, respectively, along its 3 km circumference ring to collide with a center of mass energy equal to the mass of the $\gamma(4s)$ resonance[1], i.e., 10.58 GeV. The two rings, respectively known as the High Energy Ring (HER) and Low Energy Ring (LER), are made to cross at the point called Interaction point (IP), where the Belle II detector was installed to measure various particles and their effects produced from the collision. A layout of the SuperKEKB accelerator and a 3D model of the Belle II detector is shown in Figure 3.1.

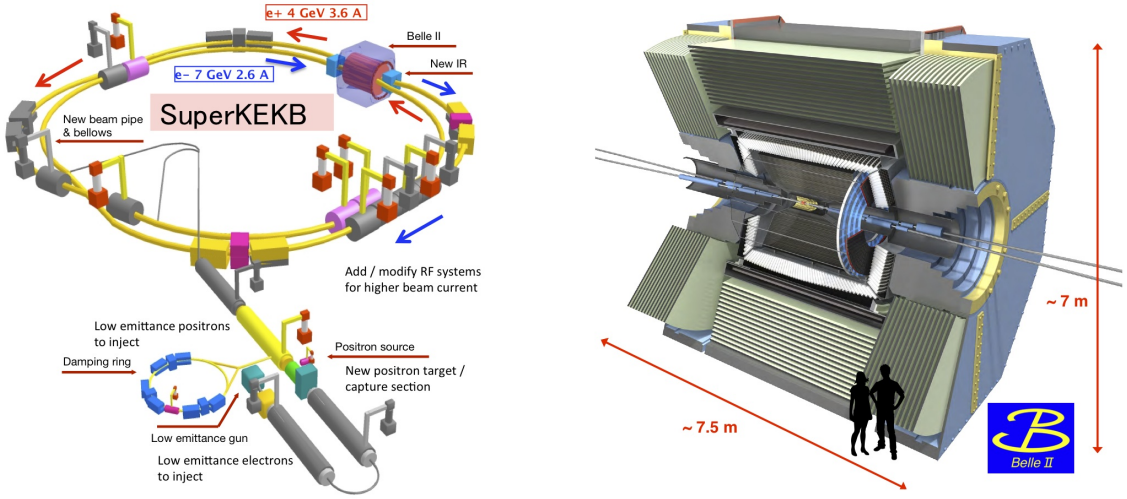


Figure 3.1: The SuperKEKB accelerator along with a 3D view of Belle II Detector. Figure adapted from [1].

[1]The resonance is an excited state in the bottonium spectrum.

Belle II detector is designed to hold on to the experiments at extremely high luminosity, i.e., $8 \times 10^{35} cm^{-2} s^{-1}$, which is 40 times higher than the previous Belle experiment. It retains the framework and certain elements of its predecessor, Belle, but all its sub-detectors are upgraded. One of the unique features of this experiment is that it produces a very clean sample of $B^0\bar{B}^0$ pairs in a quantum correlated 1^{--} state, and due to a large amount of B meson production it referred to as B-factory. A general production scheme of Belle II is shown below:

$$e^+e^- \rightarrow \gamma(4S) \rightarrow B\bar{B} \qquad \text{cross-section} = 1.2 \text{ nb} \tag{3.1}$$

Also Continuum-events $e^+e^- \rightarrow q\bar{q}$ where $q \in \{c, s, d, u\}$ also occur with a production cross-section of 2.8 nb.

3.1 Belle II Detectors

The Belle II detector consists of concentric layers of sub-detectors stacked around the interaction region at different levels. To fully reconstruct events, each sub-detector has a unique function and contributes broadly into three categories: tracking, calorimetry, and particle identification. Tracking detectors provide information on position and momentum; calorimetry enables the measurement of energy, while data on energy loss and penetration depth aids in particle identification. The various components of the Belle II detectors are described below:

Vertex Detector (VXD)

It is in the innermost part of the Belle II detector that consists of two sub-devices, silicon pixel detector (PXD) and silicon vertex detector (SVD), with a total of Six layers around the beam pipe (radius = 10 mm). The first two layers make up the PXD with a cylindrical assembly of rectangular-shaped modules containing arrays of depleted field effect transistors (DEPFET). The next four layers consist of double-sided silicon strip sensors, which make up the SVD. Both PXD and SVD task is to measure the spatial location of charged particles emitted from the IP and help in the precise measurement of the decay vertex of particles that decay within the radius of the beam pipe, such as B mesons. The first two layers are different from the next ones to provide high granularity pixel detectors, as larger beam backgrounds are expected at the IP region.

Central Drift Chamber (CDC)

After PXD and SVD, the next sub-detector is CDC consists of sense and field wires parallel to the beam direction, contributing to fully reconstructing a three-dimensional (3D) helix

track with the combined information from PXD and SVD. It also measures the energy losses from the particles.

Particle Identification System (TOP and ARICH)

The time-of-propagation (TOP) surrounds the CDC from the barrel region, i.e., parallel to the beam pipe, and aerogel ring-imaging Cherenkov detector counters (ARICH) covers the endcap region. Both have aerogels in which Cherenkov photons are created when a particle passes through them. In TOP, these photons are guided through a quartz radiator that forms the image in a ring and helps in PID by using the arrival time and position of these Cherenkov photons to determine the mass of the incident particle, whereas in ARICH, photons rings are formed on the detector surface, i.e., an array of photo-sensor modules helps in the detection of charged particles.

Electromagnetic Calorimeter (ECL)

The next sub-detector in the series is ECL, whose foremost purpose is to provide energy and position of the particles. It surrounds the TOP and ARICH with a highly segmented array of thallium-doped cesium iodide CSI(TI) crystals. The working principle of ECl is based on scintillation, as when particles penetrate into these crystals create a shower of lower-energy photons and electrons. These secondary particles ionize the atoms in the crystal, causing them to emit light, which is detected by photomultiplier tubes (PMTs) placed at the ends of the crystals. The amount of light produced is taken as a signal and then calibrated with the help of the data acquisition system of the Belle II experiment to reconstruct the properties of the particles produced in the collision.

The detector primarily helps in the detection of neutral particles as, for them, the starting point for information is ECL. Additionally, it helps in the detection of other charged particles like electrons, muon, etc., based on their shower shape in combination with other sub-detector information. Since both π^0 and γ particles are neutral, this sub-detector is the focal point for the thesis work. The discussion of this sub-detector is continued in the next section.

K_L muon detector (KLM)

It is the last and outermost detector of the Belle II which consists of an alternating sandwich of 4.7 cm thick iron plates and sensitive detector elements located outside the superconducting solenoid. It helps identify particles with long decay times and enough energy, such as K_L^0 mesons by glass electrode resistive plate chambers (RPC), present between iron plates.

3.1.1 The Belle II Electromagnetic Calorimeter

ECL serves our main purpose, i.e., measurement of energy through the principle of scintillation as described above. In detail, if we talk about this, it consists of a 3 m long barrel region with an inner radius of 1.25 m and endcaps at a distance z = 1.96 m (forward) and z = -1.02 m (backward) from the interaction point. The ECL crystals are arranged to achieve the smallest gap between crystals, which leads to minimum energy leakage.

The barrel region consists of 6624 crystals in 29 distinct shapes, and endcaps have 2112 crystals in 69 shapes with slight offset (irregularity) toward the IP. Figure 3.2 represents the schematic view of ECL with the arrangement of crystals and geometrical parameters given in Table 3.1.

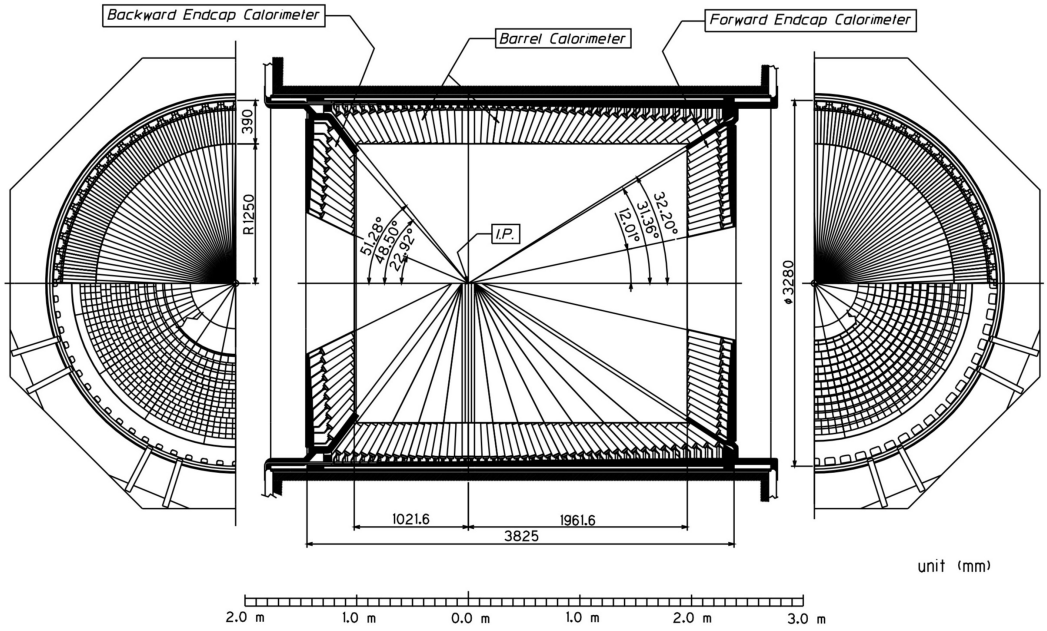


Figure 3.2: Schematic View of overall Cs(TI) Electromagnetic calorimeter. Figure adapted from [7].

ECL Region	θ Coverage (in degrees)	No. of Crystals
Full Detector	$[12.4, 155.1]$	8736
Forward Endcap	$[12.4, 31.36]$	1152
Barrel	$[32.2, 128.7]$	6624
Backward Endcap	$[131.5, 155.1]$	960

Table 3.1: Geometrical Parameters of ECL Detector

The ECL detector comprises three named regions that collectively cover 91% of 4π.

The gaps between the endcap and barrel facilitate cabling and piping. Additionally, each crystal segment is slightly tilted at an angle of 1 2° in both theta and phi directions to prevent photons from escaping between crystals. However, the tilt angle was not incorporated for the endcap phi direction due to challenges in mechanical construction.

Typically, the ECL barrel crystal takes on the shape depicted in Figure 3.3, with a trapezoidal cross-section resulting from its pointed design. Its front face measures around 5.5cm while its back face measures around 6.5cm, and the length of each crystal is approximately 30cm. There are a total of 29 distinct types of barrel crystal, each weighing roughly 5kg [7].

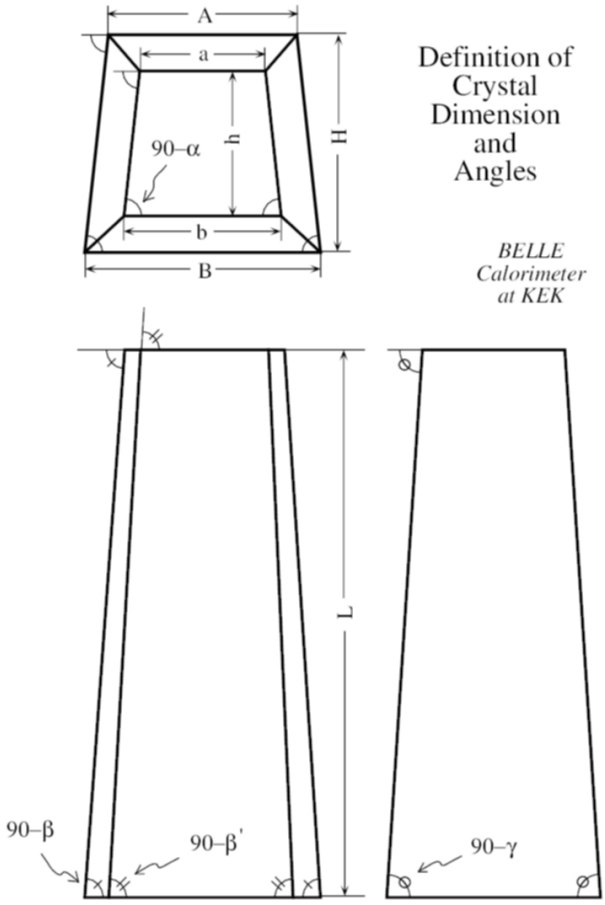


Figure 3.3: Geometry of ECL Barrel Crystal. Figure adapted from [7].

3.1.2 Belle II Analysis Software Framework (`basf2`)

The Belle II Analysis Software Framework (`basf2`) provides the complete package for the Belle II experiment, from collecting data from detectors to simulating the same events, including the framework for post-analysis tasks such as reconstruction, clustering, and PID through its built-in modules and libraries on the acquired data. The basic pipeline of Data handling and processing at the Belle II experiment is shown in Figure 3.4..

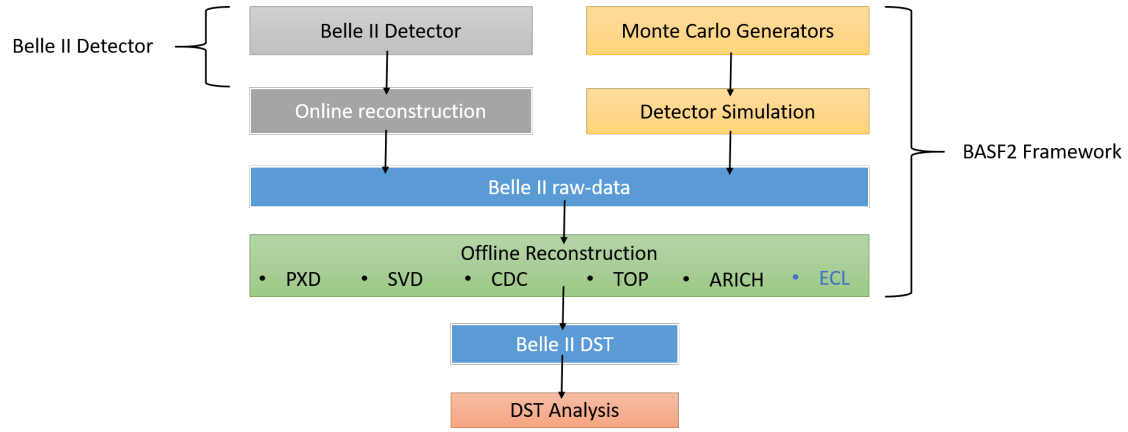


Figure 3.4: Basic Pipeline in Belle II Experiment

A module is `basf2`'s fundamental processing unit. Everything is done in modules, which are typically written in `C++`. A user can choose which modules should be performed and in what sequence they should be executed. Typically, event processing is represented as a linear chain of modules on a path. A `Python` script defines commands for creating the path, loading modules, and running the path. This Script for particular analysis is known as the steering file. During event processing, the data sets required by `basf2` are saved in "DataStore," a shared storage location. `basf2`'s data storage format is `ROOT` I/O, which means that information or output can be printed after event processing or recorded in a `ROOT` file.

In essence, the whole event cycle is defined in `basf2` as a chain of modules representing event production, geometry creation, simulation, reconstruction, and user analysis [8]. The well-known third-party libraries such as `EvtGen` (Event Generation), `Geant4` (Detector Simulation), and `ROOT` (Data Processing and Analysis) are used in the Belle II experiment to execute the above-shown pipeline.

Note:- `basf2` *is not used for analysis like (histogramming, fitting 1D distributions, ...). These final steps are usually called the "offline" analysis done by other software.*

Chapter 4

Data Generation & Analysis

Simulations[1] have gained massive popularity in the 21^{st} century as they predict the outcome of our theories without actual experiments—especially Monte Carlo simulations, which are one of the first steps before doing any particle physics analysis. Experiments like Belle II are tedious and demand a considerable workforce to operate the detector at its best to provide the collision data. So it becomes necessary for physicists in this field to analyze simulated data, i.e., MC data, before plugging their theory into real data. As by utilizing MC simulation, physicists can better understand particle properties and behavior, as well as the detector's performance. The simulated data produced through MC is analyzed and processed similarly to real data, except that MC samples contain the reconstructed objects and the "truth" information of the originally generated events.

Throughout this thesis, Monte Carlo-generated and simulated events (MC Data) are utilized, and the underlying truth information provided by this data is crucial for training the Convolution Neural Network introduced in Chapter 5. This chapter outlines the methods for generating and selecting events that the models are trained and evaluated on.

4.1 Monte-Carlo Simulation

The `basf2` described in the last section of the previous chapter is the same computing tool used to perform our necessary MC simulation[2] that generates the particles according to our given requirement.

For this thesis work, π^0 and γ events are generated using the `ParticleGun`[3] module from the Event Generator package of the `basf2`. This module could generate single or

[1] Simulation is the use of computing tools to generate a virtual model of a physical system using mathematical equations and algorithms to mimic its behavior under various situations.

[2] It utilizes random numbers to solve complex problems, models PDFs for the given system, and obtains results by averaging over multiple simulations.

[3] The module is purposefully used to know the effects of π^0 and γ in the detector separately.

multiple particles without specifying any given decay reaction. Basically, it shoots particles with a specified momentum from a designated vertex position in the direction indicated by the coordinates (θ, ϕ). The simulation is completed by adding the remaining packages and modules, which account for the response of the Belle II detector when generated particles interact with sub-detectors. Moreover, for ease of post-analysis, we have included the `ECLFillCellIdMapping` module in the steering file[4], which populates a data object that facilitates mapping between cell ID and store arrays. This neighbor map enables mapping between cell ID and the `ECLCalDigit` store array. At the start of the steering file, the total number of events is given by `EventInfoSetter` module received by the generator, and the output is saved as a `ROOT` file with the help of `Output` module. In the appendix, in listing A.1, the sample source code (steering file) and a more detailed description of these modules can be found.

4.2 Data Preprocessing

Our study employs the above-described general framework for steering files, which we use to create two separate steering files: one for π^0 and the other for γ. Through these files, we simulate particles for each π^0 and γ, with momenta, θ (Polar angle), and ϕ (Azimuthal angle) within the specified range for different setups, respectively. Chapter 6 provides a more comprehensive explanation of these setups.

The above data is preprocessed to get the desired features, i.e., energies deposited in the crystals and the position of the Seed crystal (Crystal of maximum energy) as inputs given to CNN for each particle.

4.2.1 Events Selection

If both merged π^0's and γ are involved in an event from different mother particles, then the showers' hit pattern in the form of an ECL image is shown in Figure 4.1. The image shown represents the deposited energy by the merged π^0's and γ when interacting with the ECL detector's barrel crystal at different hit locations. Also, it clearly illustrates the challenging task of distinguishing merged π^0's and γ showers, which our method aims to address. At present, various clustering algorithms and shower shape momenta variables are employed in `basf2` to identify these particles and their associated showers. Therefore, determining which shower corresponds to which particle can be challenging with `basf2`, making it impossible to obtain accurate information.

[4] A steering file or script refers to a piece of Python code designed to configure and initiate a specific data processing or analysis task by compiling several modules with given parameters.

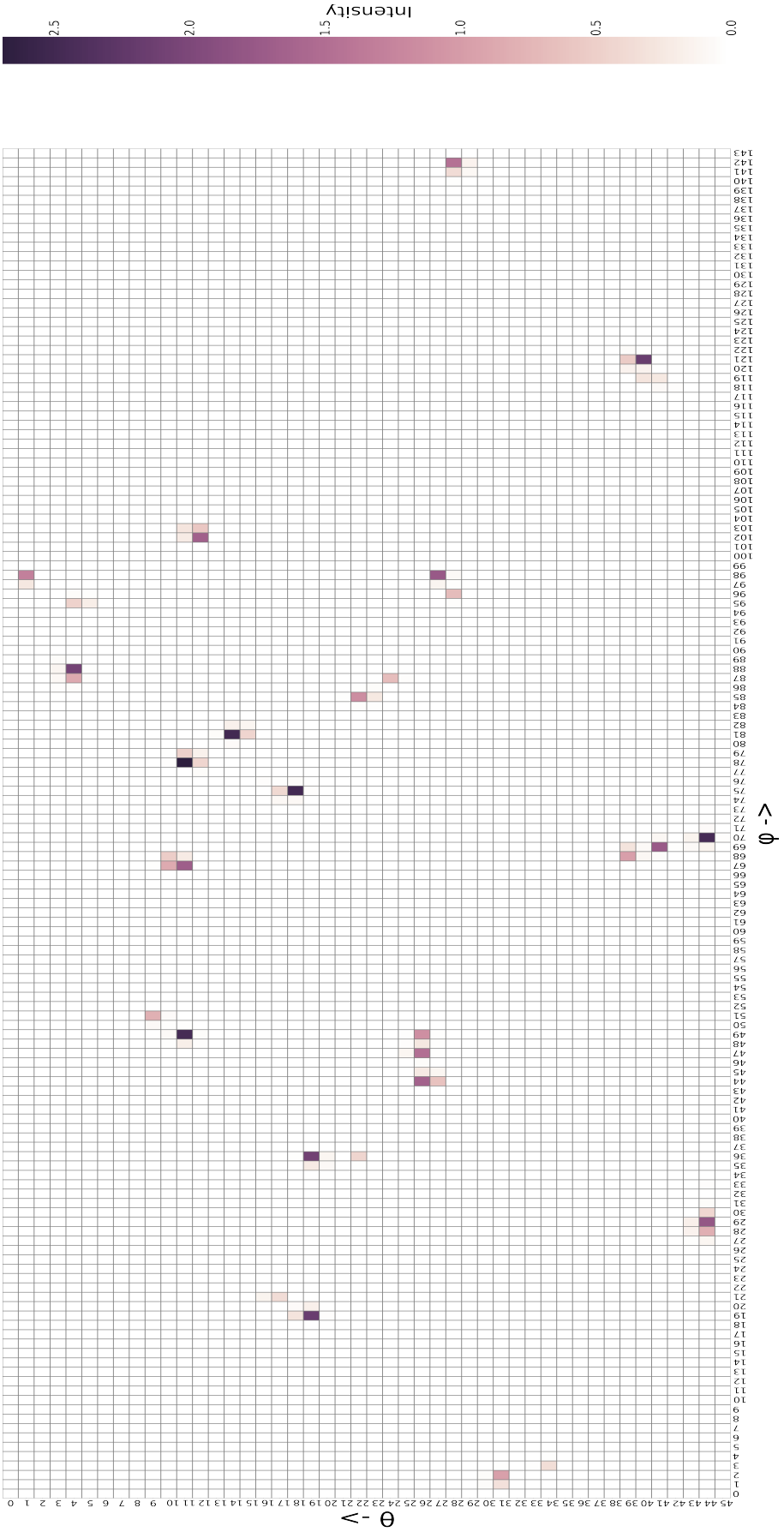


Figure 4.1: Showers of π^0's and γ at different hit locations in Barrel region of ECL.

Therefore, in order to train the CNN effectively, a well-controlled environment is the first step to accurately interpreting and comprehending the performance of a Convolutional Neural Network (CNN) while ensuring comparability with the current `basf2` algorithm for identifying these particles. Furthermore, in machine learning, utilizing a set of specific samples free from outliers can lead to stable training. So to obtain such reliable samples for training, the focus must be on obtaining distinct ECL clusters exclusively associated with either π^0 or γ particles. This requirement underscores the importance of carefully selecting events for training and testing our model.

The arguments mentioned above are reinforced by creating individual root files for each π^0 and γ particle that solely encompasses these particle's effects without interference[5] from other particles. This approach ensures that each shower accurately reflects the unique characteristics produced by these specific particles, resulting in a more representative signature. This signature is the Region of Interest (ROI) containing a window of 7×7 neighborhood crystals around the seed crystal, obtained with the help of variables related to ECL. These variables are stored in a `ROOT` file after getting from the steering file and looked for this purpose using Uproot library with the help of stored variables like `ECLLocalMaximums`, `ECLCalDigits`, and `ECLCellIdMapping`. This provides us with the desired features that are stored in a CSV file. A piece of code to handle and get the desired feature from the `ROOT` file is mentioned in listing A.2 of the Appendix.

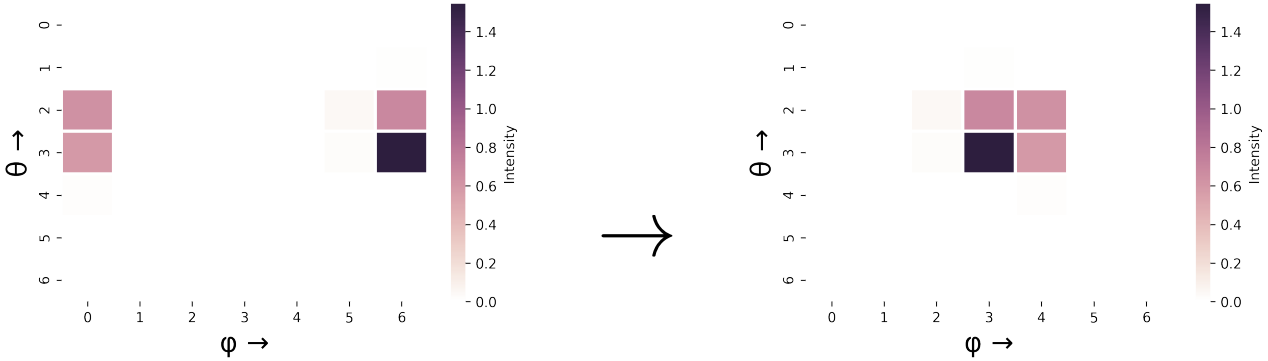
4.2.2 Feature Engineering

The selection of a 7×7 window around the seed crystal ensures the 49 features, i.e., pixel values with their label stored in a CSV file. As stated earlier in this section, these features are obtained as the raw form of energy, so to improve model performance, these features are further refined. The following images are shown in Figure 4.2 illustrates the processing of one gamma sample, with each step demonstrated in sequence. Also, the techniques used for this refinement are described in sequence,

1. **Centering**

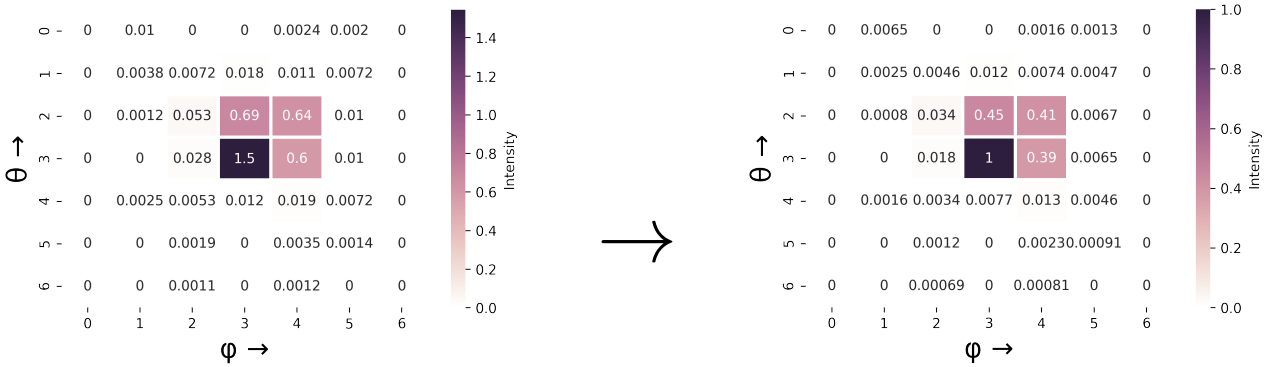
    - The centering transformer shifts the image such that its center (seed crystal) is aligned with the center of the image.
    - It can help improve image quality and make it easier for CNN to extract meaningful features. This is useful when the same cluster has two showers of different particles.

[5]This interference is completely avoided by associating each event with either only one merged π^0 or one γ, means that setting parameter 'ntracks = 0' while creating root file for each particle.



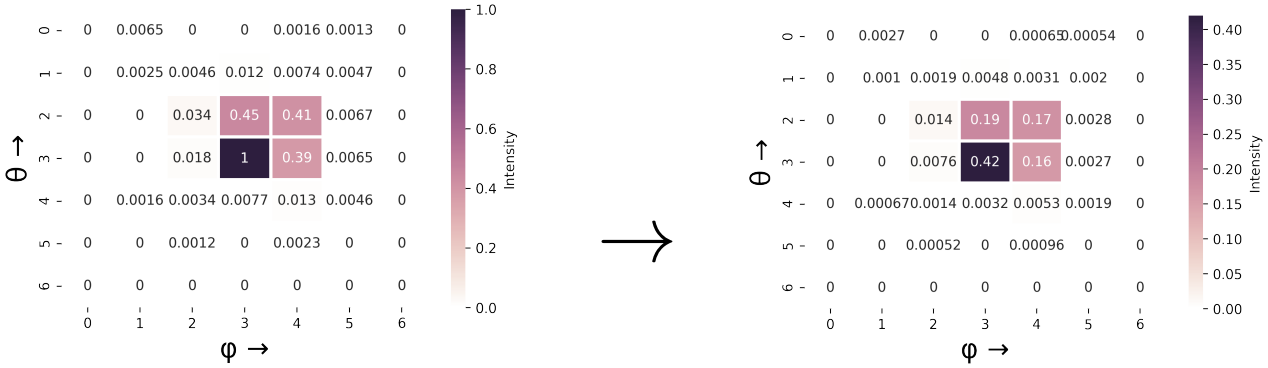
(a) Raw Sample Image for One γ

(b) Centering transformation on (a)



(c) Centered image with intensity

(d) Scaling transformation on (b)



(e) Noise reduction on scaled image i.e,(d)

(f) Normalization transformation on (e)

Figure 4.2: Images of one γ cluster after sequential steps of transformation to obtain a final image for feeding into the model.

2. **Clipping and Noise reduction**

   - This transformer removes extreme pixel values from the image and reduces the noise by setting all pixels to 0 if the pixel intensity is below the provided threshold.
   - The purpose of this is to ensure that the object of interest in the image is in the center, which can improve the performance of the CNN during training.

3. **Scaling**

    - It scales the image's pixel values to a typical range between 0 and 1.
    - It helps to ensure that all the images have the same range of values, which can improve the performance of the CNN during training.

4. **Normalization**

    - The normalization transformer scales the image's pixel values such that the sum of all pixel intensities is equal to 1.
    - It ensures that the image represents a probability distribution, which can be helpful in applications such as object detection or segmentation.

In general, using these transformers, which ultimately enhance the quality of images being input into the CNN, improved the model's accuracy.

Chapter 5

Convolution Neural Network

When it comes to classifying images, feature extraction is a critical process, and several effective methods are available for different visual recognition tasks. However, Convolutional Neural Networks (CNNs) have emerged as the most widely used and successful method for image recognition tasks. One of the reasons for their success is their ability to learn hierarchical representations of image features directly from raw pixel values, achieved through deep learning. This approach eliminates manual feature engineering and provides more flexibility, making CNNs well-suited for various image recognition tasks. Due to this reason, In this thesis, CNN is used to classify our dataset as it consists of images created from a raw matrix of 7x7, with energy values taken as pixel intensities. In the upcoming sections, we will take a closer look at the basic theory of CNN, also its architecture and implementation for our dataset.

5.1 Basics

Convolutional Neural Network (CNN) is a deep learning neural network architecture designed to work with images and other grid-like structures, such as time-series data. It is inspired by the organization of the visual cortex in animals, where neurons in different layers of the cortex respond to increasingly complex visual features.

The network comprises multiple layers, each of which performs a specific operation on the input data and is therefore trained to classify images into multiple classes, typically using a softmax activation function at the output layer to produce probability scores for each possible class. However, in our case, the task involves binary image classification, assigning images to one of two possible classes. While this task may seem simpler than multi-class classification, it still requires an effective approach (discussed in the implementation section 5.2) for distinguishing between the two classes. So, we will focus on using CNNs for binary image classification, exploring how these networks can be trained to classify images

accurately into two distinct classes.

The main components of a CNN are:

1. **Convolutional layers**: These are responsible for extracting features from the input image. They consist of a set of learnable filters, also known as kernels or weights, that are convolved with the input image to produce a set of feature maps. Each filter has a feature map highlighting a specific pattern or feature in the input image, such as edges or corners. The convolution operation can be mathematically expressed as follows:

$$F_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} I_{i+m-1,j+n-1} K_{m,n}, \tag{5.1}$$

where I is the input image, K is the convolution kernel or filter, and F is the resulting feature map. M and N are the dimensions of the kernel, and i and j are the indices of the output feature map.

The size of the feature map F after applying the convolution operation can be computed using the following formula:

$$n_{out}^{row} = \frac{n_{in}^{row} - k_{row} + 2p_{row}}{s_{row}} + 1, \; n_{out}^{col} = \frac{n_{in}^{col} - k_{col} + 2p_{col}}{s_{col}} + 1 \tag{5.2}$$

Where n_{in}^{row} and n_{in}^{col} are the number of rows and columns in the input feature map, respectively. Similarly, n_{out}^{row} and n_{out}^{col} are the number of rows and columns in the output feature map, respectively. k_{row} and k_{col} are the size of the convolution filter in the row and column dimensions, respectively. p_{row} and p_{col} are the amounts of zero padding (if any) in the row and column dimensions, respectively. Finally, s_{row} and s_{col} are the stride sizes in the row and column dimensions, respectively.

2. **Pooling layers**: These are used to reduce the spatial dimensionality of the feature maps produced by the convolutional layers. They achieve this by downsampling the feature maps using max or average pooling. Max pooling selects the maximum value within a local region of the feature map, while average pooling computes the average value within the same region. Pooling layers help make the network more robust to small variations in the input image and reduce the network's computational complexity. The max pooling operation can be expressed as follows:

$$F_{i,j} = \max_{m=1}^{M} \max_{n=1}^{N} I_{iM+m-1,jN+n-1}, \tag{5.3}$$

where M and N are the dimensions of the pooling kernel, and i and j are the indices of the output feature map.

In general, the size of the output feature map F after applying max pooling can be computed using the formula:

$$F = \left\lfloor \frac{n_{in} - k}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_{in} - k}{s} + 1 \right\rfloor, \tag{5.4}$$

where n_{in} is the size of the input feature map, k is the size of the pooling window, and s is the stride. The symbol $\lfloor x \rfloor$ denotes the floor function, which returns the largest integer less than or equal to x.

3. **Fully connected layers**: These are used to map the extracted features to the binary output. They are similar to the dense layers in a traditional neural network, in which each neuron in the preceding layer is linked to every neuron in the layer above it and vice versa. They are typically placed at the end of a neural network architecture after one or more convolutional or pooling layers that extract relevant features from the input data.

Working: Convolutional Neural Networks (CNNs) operate on a hierarchical approach, where each layer builds upon the previous one to extract more complex and abstract features from the input image. The first layer typically consists of convolutional filters that slide over the input image and perform a dot product operation to extract certain features and visual features, such as edges or corners. These filters can be of varying sizes and depths and include additional parameters such as padding and stride. The convolutional layer's output is then sent through an activation function, such as ReLU, tanh, or others, to introduce nonlinearity into the network. The subsequent layers learn more complex features by combining the previous layers' outputs, which include pooling layers, that downsample the feature maps to reduce the computational cost and introduce translation invariance. Finally, fully connected layers perform the final classification task based on the learned features by giving output to some activation functions that provide probabilistic output. In our case, the output of the last fully connected layer is fed to a sigmoid function, which produces a probability score for the binary output. The sigmoid function can be expressed as follows:

$$P = \frac{1}{1 + e^{-z}}, \tag{5.5}$$

Where P is the probability score for the binary output, z is the output of the last fully connected layer.

During training, the network is presented with a labeled set of images, and the network weights are iteratively updated to minimize the difference between the predicted and actual labels. This process is done by minimizing a loss function that quantifies the difference between predicted and actual output, and it typically involves using backpropagation to compute the gradient of the loss with respect to the network parameters and updating the weights with an optimization algorithm such as stochastic gradient descent (SGD) or the Adam optimizer.

The binary cross-entropy loss function can be expressed as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)], \tag{5.6}$$

Where N is the number of samples in the training set, y_i is the actual output (either 0 or 1) for the i-th sample, and $\hat{y}_i$ is the predicted output for the i-th sample. The loss function measures the difference between the predicted probability score and the actual output.

Some other techniques, such as Dropout and Batch Normalization and several hyper-parameters, need to be tuned in a CNN, which involves the number of convolutional and pooling layers, the size of the filters, the number of filters in each layer, and the learning rate. The choice of hyperparameters depends on the specific problem and the size of the training set. The goal is to find the set of weights that result in the lowest possible loss on the training data, hopefully resulting in a good performance on new, unseen data. In our case, i.e., binary classification, the binary cross-entropy loss function, and the Adam optimizer are used. Once trained, the network can predict new, unseen images.

Some terms used above are described below:

1. **Padding**: It adds extra zeros (i.e., extra border pixels to an image) around the input image to ensure the output size remains the same, allowing the convolutional filters to extract features from the edges of the input image and prevent loss of information due to border effects. Padding width should obey the given mentioned formula:

$$\text{Padding width (p)} = \frac{k-1}{2} \tag{5.7}$$

   Where k is the filter size.

2. **Stride**: It specifies how much the filter should move over the input image at each step.

3. **Dropout**: It is commonly used in CNNs to prevent overfitting, where some neurons in the network are randomly dropped out during training to reduce their interdependence.

4. **Batch Normalization**: It is also frequently used to standardize the inputs to each layer, improving the stability and convergence of the network during training. It normalizes the inputs to each layer, making the network less sensitive to the scale of the input features. This helps to prevent the internal covariate shift problem and allows for faster and more stable training.

In summary, CNNs are a powerful tool for binary classification tasks involving image data.

5.2 Network Architecture and its Implementation

In order to design an efficient CNN architecture for binary image classification, it is crucial to carefully consider various significant factors, as discussed earlier. These factors primarily comprise the input image size, the number, and dimensions of convolutional and fully connected layers, the utilization of pooling and regularization techniques, and the selection of the optimization algorithm and loss function.

Deep learning frameworks are essential tools for developing and training complex neural networks such as CNNs. Various open-source deep learning frameworks such as Keras, TensorFlow, and PyTorch are available. These frameworks provide a high-level API that allows developers to focus on the model's architecture and design rather than low-level implementation details, eventually helping to easily build and deploy their models by defining the model's layers, specifying the loss function, optimizer, and evaluation metrics, and train the model on a labeled dataset. These frameworks also provide access to various pre-trained models and architectures that can serve as a foundation for building a customized CNN model. By utilizing these pre-built models, we can accelerate the process of creating a CNN architecture and streamline the development process, which further allows experimenting with different architectures and hyper-parameters faster and more efficiently, ultimately resulting in the selection of the optimal model for the specific task.

Keras was chosen as the deep learning framework for this thesis project due to its widespread adoption in the machine learning community due to its versatility, ease of use, and compatibility with both CPU and GPU architectures. Keras is designed on top of popular deep learning frameworks, including TensorFlow, Theano, and CNTK, and offers a streamlined and consistent API across different backends. It provides a wide range of pre-built layers and models, including convolutional layers, pooling layers, and fully connected layers, making it easy to build complex models such as CNNs. It also supports many optimization algorithms and loss functions, doing training and optimizing our models easy. Additionally, Keras provides valuable data preprocessing and augmentation tools, which can be critical for achieving high performance in computer vision tasks. These features help explicitly with our problem also.

So basically, achieving optimal results in CNN-based binary image classification involves an iterative process of experimenting with different architectures and hyper-parameters. This entails evaluating various combinations of hyper-parameters and architectures to identify the best model for the given problem. Thus, building an effective CNN architecture for binary image classification is a challenging yet critical undertaking that demands a profound comprehension of CNNs and a willingness to engage in experimental approaches to obtain the best possible results.

5.2.1 Optimized CNN

A number of well-known CNN architectures, including AlexNet, VGG, ResNet, and Inception, have demonstrated exceptional performance in image classification tasks. We began with a straightforward VGG structure comprising 2 to 3 convolution layers and two dense layers for our specific problem. After that, applying several modifications to this base CNN architecture by gradually increasing its complexity results in improved training set performance. Some basic steps to achieve this optimal Configuration of CNN are:

- The Random Search Optimization method is employed for refining architecture by randomly sampling hyperparameters such as kernel size, features, activation functions, and dropouts from a specified distribution. The model is then trained and assessed for each set of hyperparameters. This approach is advantageous for large data sets as it is computationally less demanding.

- Subsequently, we explored different hyperparameter values, such as learning rate, batch size, and number of epochs, to improve accuracy. Our results demonstrate that reducing the learning rate and increasing the batch size significantly improved accuracy, whereas increasing the number of epochs did not improve performance beyond a particular limit.

- Also, implementing dropout regularization and weight decay prevents overfitting and improves generalization. This led to a reduction in the validation loss and a better trade-off between bias and variance.

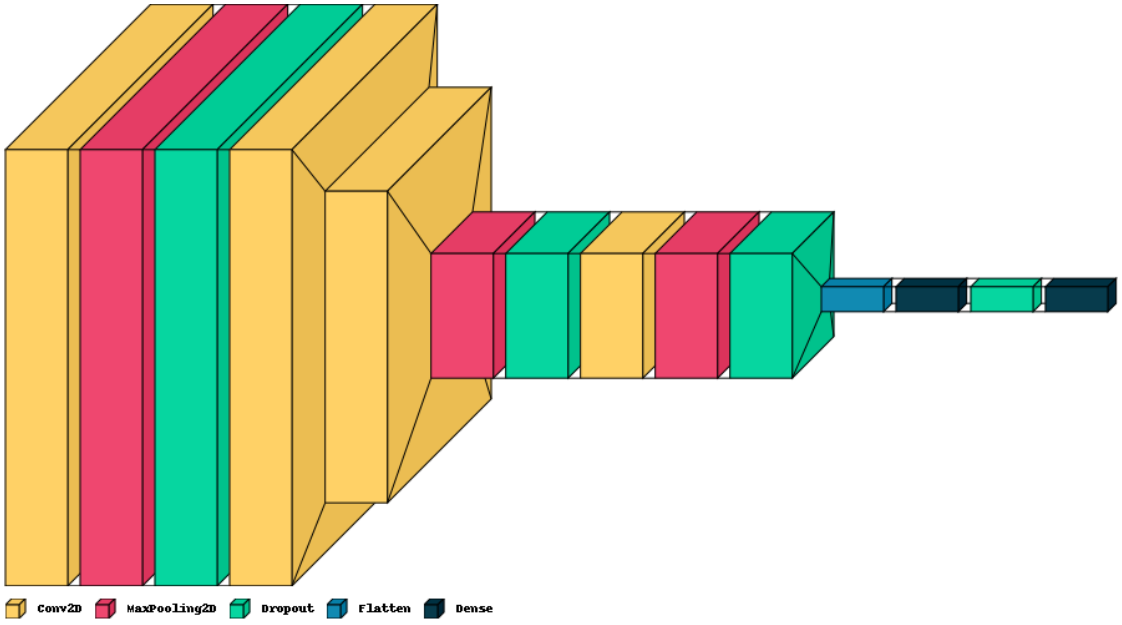


Figure 5.1: Optimized CNN Architecture

The optimized network architecture derived in this thesis is presented in Figure 5.1, and Table 5.1 shows a summary of this optimized CNN. It comprises three hidden convolution layers with 32, 64, and 64 nodes for the first, second, and third hidden layers. The rectified linear unit (ReLu) is each layer's activation function. The output layer consists of two nodes and a sigmoid activation function. The model employs the Binary cross-entropy loss function and Adam optimizer. The optimized network achieved an accuracy of 97%, which was 7.2% higher than the base CNN (not shown in the thesis). The optimized network section showcases our ability to modify CNN architecture to achieve better results.

Layer (type)	**Output Shape**	**Number of Parameters**
Input	$(None, 7, 7, 1)$	0
Conv2D (ReLU)	$(None, 7, 7, 32)$	320
MaxPooling2D	$(None, 7, 7, 32)$	0
Dropout	$(None, 7, 7, 32)$	0
Conv2D (ReLU)	$(None, 7, 7, 64)$	18496
Conv2D (ReLU)	$(None, 5, 5, 64)$	36928
MaxPooling2D	$(None, 2, 2, 64)$	0
Dropout	$(None, 2, 2, 64)$	0
Conv2D (ReLU)	$(None, 2, 2, 64)$	36928
MaxPooling2D	$(None, 2, 2, 64)$	0
Dropout	$(None, 2, 2, 64)$	0
Flatten	$(None, 256)$	0
Dense (ReLU)	$(None, 128)$	32896
Dropout	$(None, 128)$	0
Dense (softmax)	$(None, 2)$	258

Total number of parameters: 125,826

Table 5.1: Summary of the CNN Architecture

5.3 Measuring Performance

The Optimized Network section describes the modifications and enhancements to the CNN architecture to improve accuracy and performance. This section discusses various evaluation methods commonly used for Binary image Classification problems. These evaluation methods assess the effectiveness of the modifications using several performance metrics.

The primary evaluation metric is accuracy, which measures the percentage of correctly classified images in the test set. However, accuracy may not always provide the best model

performance evaluation, especially when the dataset is imbalanced or the misclassification of certain classes is more critical than others. In such cases, precision, recall, and F1-score can be utilized to assess the model's performance for each class. Additionally, other metrics, such as the Confusion matrix and Receiver Operating Characteristic (ROC) curve, are also used in classification tasks to evaluate model performance. Each of these metrics will be discussed in detail below:

1. **Accuracy:-** It determines the percentage of all correctly categorized observations. As it takes into account the total number of true positives (TP), false positives (FP), and false negatives (FN), it is also the global mean (micro-average) of the F1 score.

2. **Precision:-** It measures the percentage of the predicted positive cases that are actually positive and tells how accurate the positive predictions are. High precision means that the model predicts very few false positives.

$$\text{Precision} = \frac{TP}{TP+FP}$$

3. **Recall:-** It measures the percentage of actual positive cases that the model can correctly identify and how well it can identify positive cases. A high recall means the model can identify the most positive cases.

$$\text{Recall} = \frac{TP}{TP+FN}$$

4. **F1 Score:-** It basically considers both precision and recall to provide a more balanced summarization of performance.

$$\text{F1 Score} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$$

5. **Confusion matrix:-** It provides a summary of the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions for each class. The confusion matrix can be used to calculate each class's precision, recall, and F1 score.

6. **ROC and AUC:-** The ROC curve plots the true positive rate (sensitivity) against the false positive rate (specificity), and the area under the curve (AUC) measures the degree of separability between the two classes.

Chapter 6

Training Analysis & Evaluation

In the realm of physics, machine learning algorithms are frequently employed for simulating intricate systems, data analysis, and prediction-making. However, to ascertain their efficacy, they must be tested against a baseline setup of the experiment. This establishes a reference point for evaluating the performance of the algorithm and determining its degree of success.

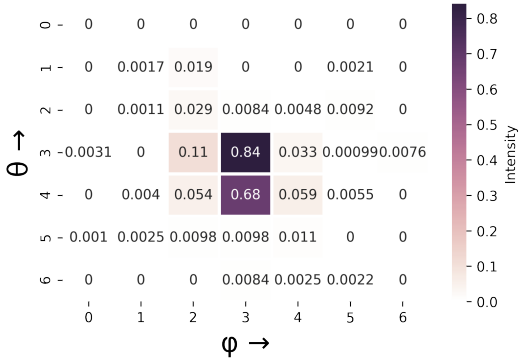
The experimentation process and methodology provide diverse scenarios for conducting various studies to optimize algorithms. This chapter aims to evaluate and comprehend the CNN behavior in these scenarios to improve their efficiency. Starting with a controlled setup, then evaluating the CNN behavior for more generalization in an uncontrolled environment across the full region (i.e., Barrel Region of the ECL Detector). By conducting these evaluations, we can thoroughly train the model and enhance its ability to reach the optimal solution (already provided in section 5.2.1).

6.1 Controlled Study

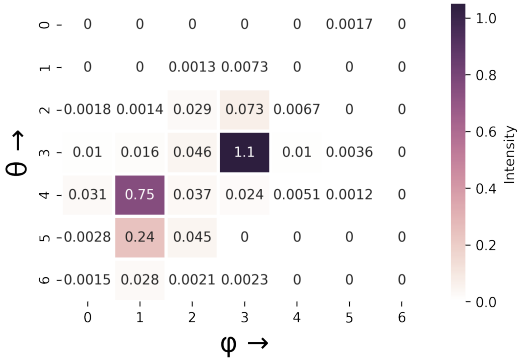
This investigation aims to evaluate the efficacy of CNN on the base setup of our study. Doing so will ascertain the feasibility of utilizing CNN for the whole region of ECL to classify the particles of our interest, namely, π^0's and γ.

Base Setup:- In this experimental configuration, our focus is on the vicinity of a randomly selected ECL crystal. By utilizing the `ParticleGun` Module of `basf2`, We produced multiple events within a specified momentum range, each consisting of a single π^0 or a single γ traveling toward the selected crystal. The relevant information, specifically energy values, was recorded for each instance as the particle interacted with the crystal.

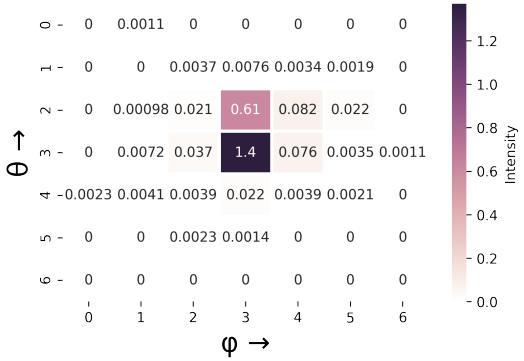
Considering the parameters mentioned in Table 6.1 will produce the effect of the hit from both π^0 and γ in the vicinity of the crystal located at $\theta = 90$ and $\phi = 0$ for different momenta within the specified range above.



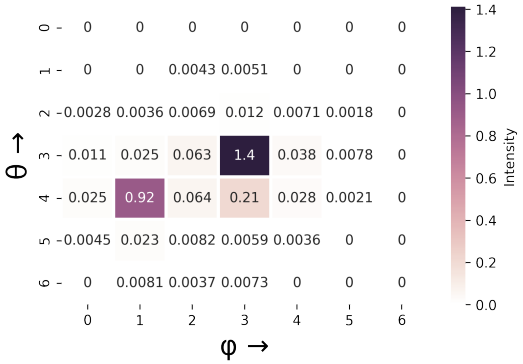
(a) γ cluster at 2.5 GeV



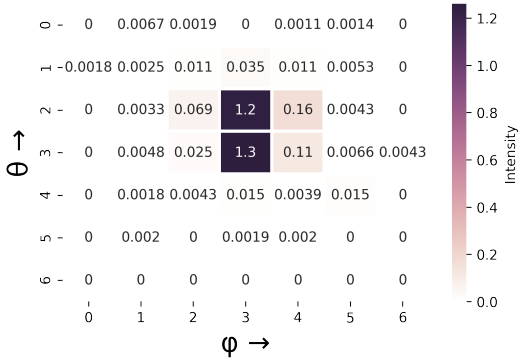
(b) π^0 cluster at 2.5 GeV



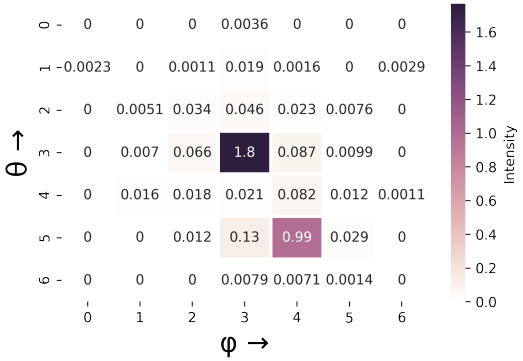
(c) γ cluster at 3.0 GeV



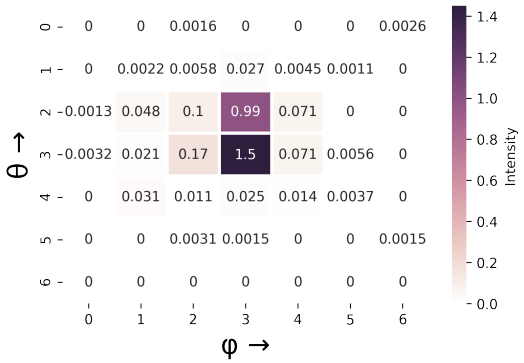
(d) π^0 cluster at 3.0 GeV



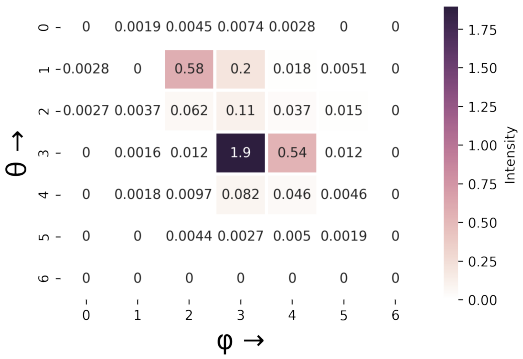
(e) γ cluster at 3.5 GeV



(f) π^0 cluster at 3.5 GeV



(g) γ cluster at 4.0 GeV



(h) π^0 cluster at 4.0 GeV

Figure 6.1: Images of γ and π^0 cluster at different momenta for the Controlled setup.

The images above display shower patterns for π^0 and γ at various momenta for the mentioned region, as described in Table 6.1. After undergoing preprocessing steps outlined in Chapter 4, these images were utilized for training a CNN for this particular setup.

Parameters	Distribution	Value
Momentum	Uniform	[2.5,4.0] (in GeV)
Phi	Uniform	[-1.25,1.25] (in degrees)
Theta	Fixed	90 (in degrees)

Table 6.1: Randomly picked parameters for base setup

Analysis

In order to perform the analysis for this particular configuration, we produced a total of 0.5×10^5 individual π^0 and γ candidates utilizing the parameters mentioned above. For each event, a window of 7×7 crystals containing the measured deposited energy was chosen around the seed crystal of every shower for each candidate. This selection process is explained in detail in section 4.2.1. Subsequently, the dataset was partitioned into three parts, approximately 7.0×10^4 for training, 1.5×10^4 for validation, and 1.5×10^4 for testing purposes.

To ensure accurate information was obtained for each shower or cluster, we limited the event generation to a single particle for π^0 and γ. This was achieved by setting the "ntracks = 0" parameter in the steering file, which we also discussed in the Event Selection section. By doing this, we were able to obtain only one shower or cluster region that corresponded to the generated particle in the entire Barrel region. That's how true labels were assigned to the data. This process constituted the entire dataset for both particles, which was used to evaluate our model's performance.

6.1.1 Efficiency & Performance

This controlled experiment provides us with a more objective and systematic evaluation of the effectiveness of our optimized CNN model for further use in the generalization of the whole barrel region. The various results obtained from this setup are shown as:

1. The model's performance during the training phase was evaluated using the validation loss and training loss plots, depicted in Figure 6.2 below. The plots demonstrated a consistent decrease in both losses over time, signifying that the model was effectively learning. Consequently, the model was able to classify the binary input data with a high degree of accuracy.

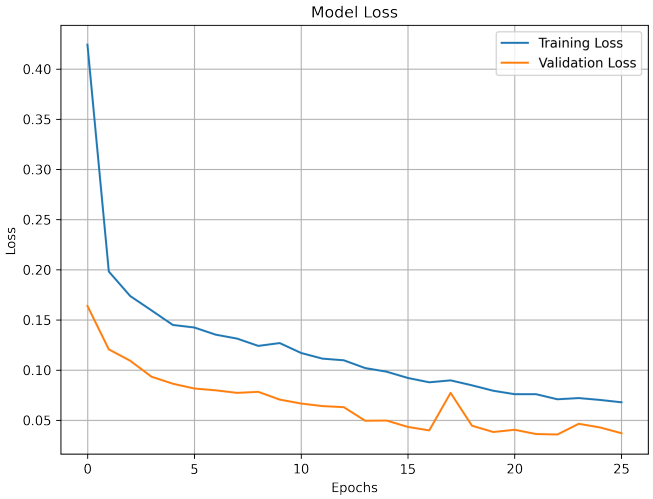


Figure 6.2: Loss Curve for controlled experiment

2. The confusion matrix evaluated the model's performance on this binary classification task. The model showed the predicted and actual values of a data set, indicating that the model could correctly classify the majority of positive and negative samples in this fixed region.

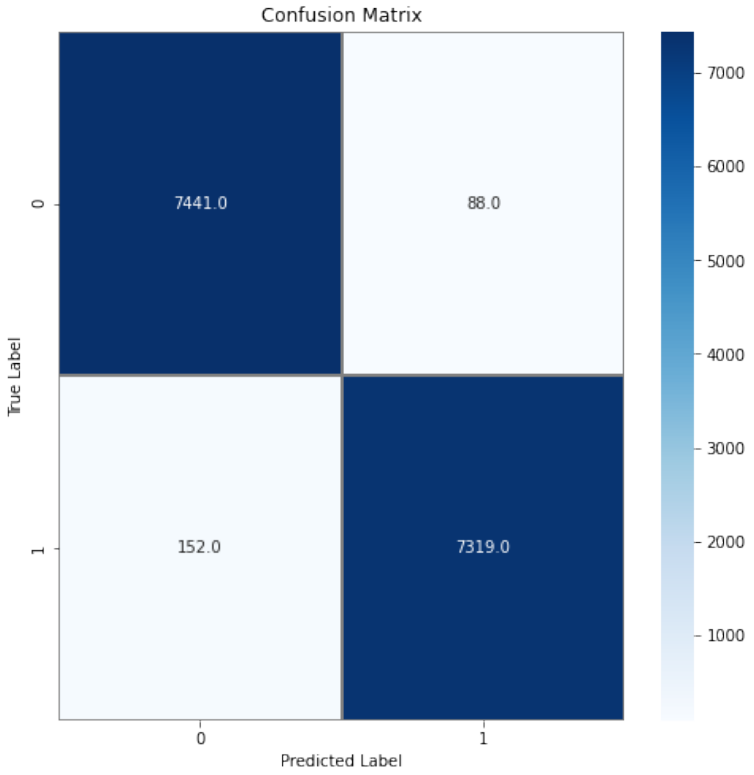


Figure 6.3: Confusion matrix for controlled experiment

3. Also, various metrics, as described in the Measuring Performance section of Chapter 5, were analyzed on this setup to measure the efficiency of the CNN model. The table below summarizes the metrics, revealing that the model's accuracy on the controlled setup was 98%.

Metric	Precision	Recall	F1 Score	Support
γ (Label: 0)	0.98	0.99	0.98	7529
π^0 (Label: 1)	0.99	0.98	0.98	7471
Accuracy			0.98	15000
Macro Average	0.98	0.98	0.98	15000
Weighted Average	0.98	0.98	0.98	15000

Table 6.2: Performace of CNN Classification in Controlled Experiment

4. Figure 6.3 displays the ROC curve, which assesses the model's proficiency in categorizing positive and negative samples. The calculated area under the curve (AUC) was 0.99, signifying that the model performed exceptionally well on the binary classification task in the defined region on the cleaned window.

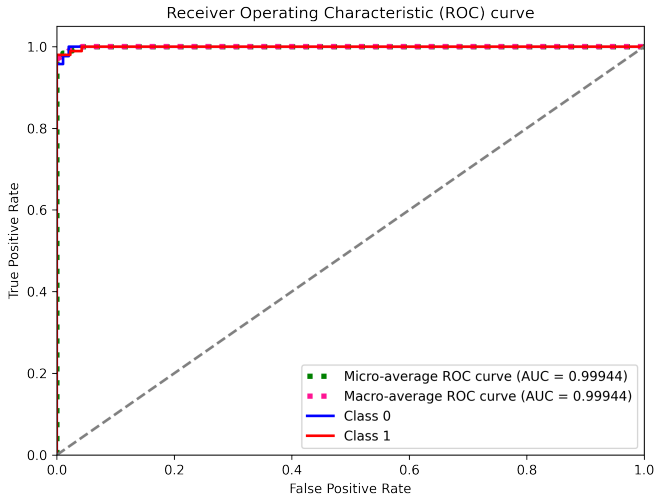


Figure 6.4: ROC Curve for controlled experiment

6.2 Uncontrolled Study

The results obtained from the controlled experiment are remarkable and provide ample scope to implement this model for generalizing the entire barrel region of the ECL detector. However, in order to generalize this model for the entire region, it is imperative to have

a larger dataset covering nearly the whole Barrel region, which would require data to be generated around each crystal. This leaves us with three potential scenarios shown in Figure 6.5 and described below:

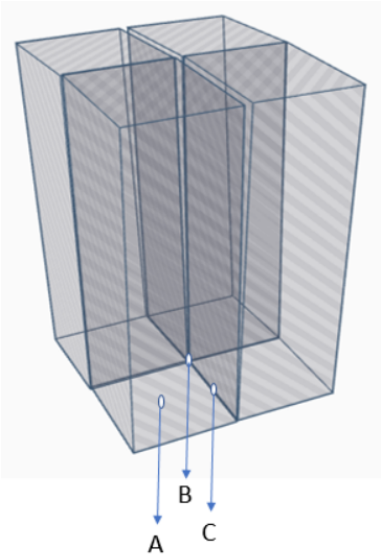


Figure 6.5: Layout of 4 adjacent ECL crystals

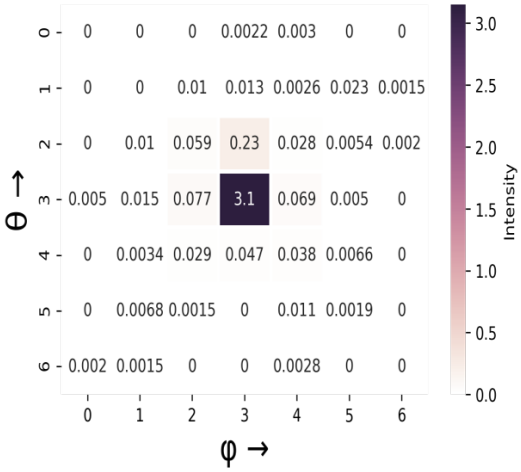
1. When a particle hits the center of crystal (A).

2. When a particle hits the corner of the crystal (B).

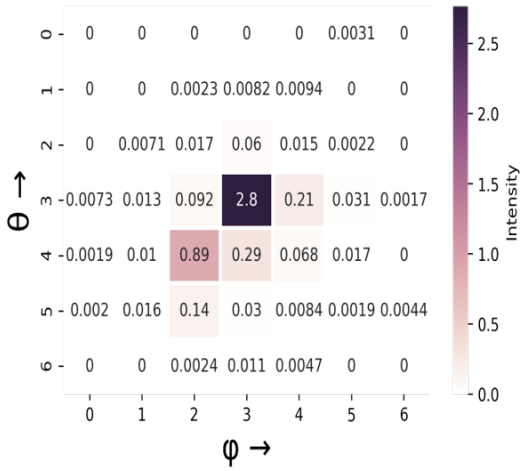
3. When a particle hits the edge of the crystal (C)

Figure 6.6 illustrates the distinct behavior of π^0 and γ at the specified locations. It includes sample figures depicting the behavior of both particles. In this setup, several events for the above-described cases are created using the same data generation method. The particleGun module is modified to cover the entire barrel region by adjusting the values of theta and phi. Additionally, to ensure that each 7×7 crystal window is properly labeled, only one particle is generated per event (same procedure followed described in previous sections).

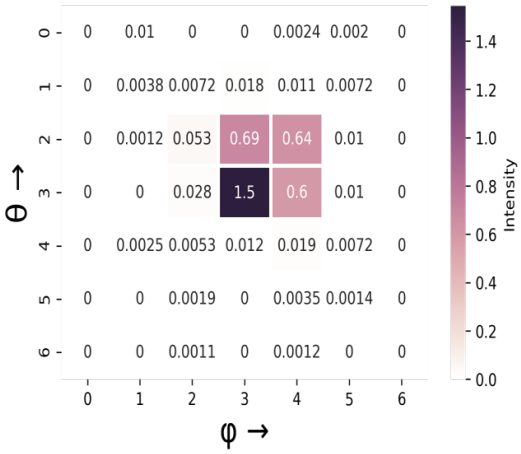
A total of 10^5 individual π^0 and γ candidates were generated for this study. Out of these candidates, approximately 1.4×10^5 were used for training purposes, while 0.3×10^5 were allocated for validation and another 0.3×10^5 for testing events. These sets were utilized to assess and evaluate the performance of the CNN model.



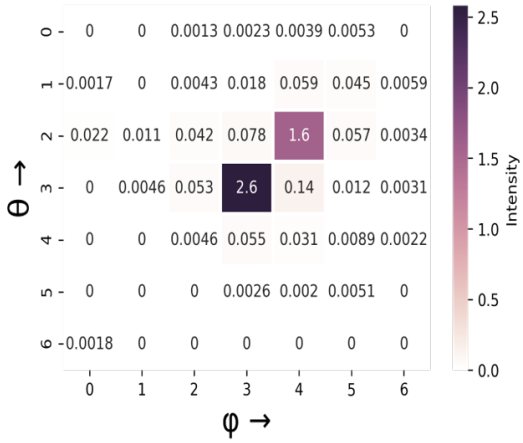
(a) γ cluster in case (A)



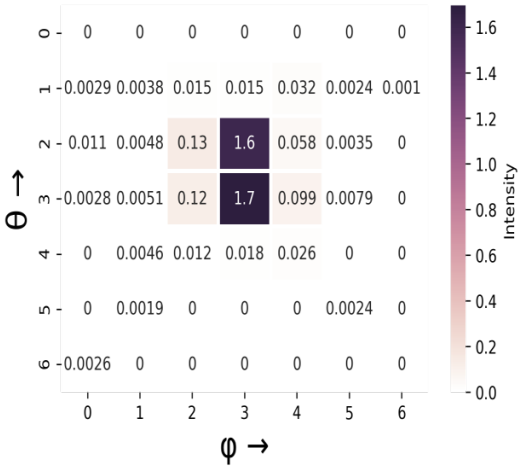
(b) π^0 cluster in case (A)



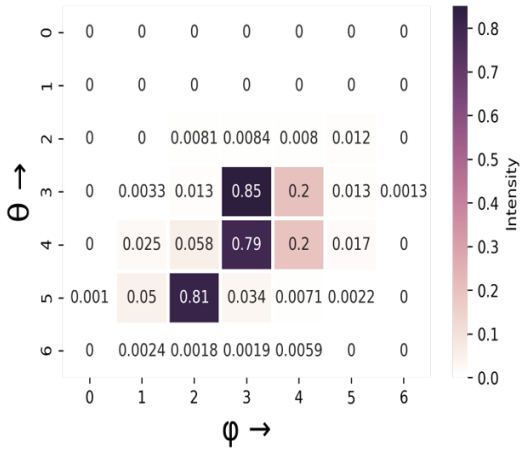
(c) γ cluster in case (B)



(d) π^0 cluster in case (B)



(e) γ cluster in case (C)



(f) π^0 cluster in case (C)

Figure 6.6: Images of γ and π^0 cluster at three possible cases around any crystal.

6.2.1 Efficiency & Performance

The findings of the uncontrolled experiment used to assess the effectiveness and performance of the Convolutional Neural Network (CNN) model are shown in this section. We examine a number of performance metrics for the model, including loss curves, confusion matrices, and Receiver Operating Characteristic (ROC) curves. By doing additional testing, discussed in the next section, 6.3, the study's findings offer insights into the usefulness and efficiency of the CNN model in real-detector instances.

1. Starting with an analysis of the model's performance during the training phase was again shown by validation loss and training loss plots, depicted in Figure 6.7 below. The plots consistently show a decrease in both losses over time, and after 45 epochs, it almost becomes constant, indicating that further improvement is not possible beyond that point. The overall plot signifies that the model was effectively learning with a high degree of accuracy on this cleaned data.

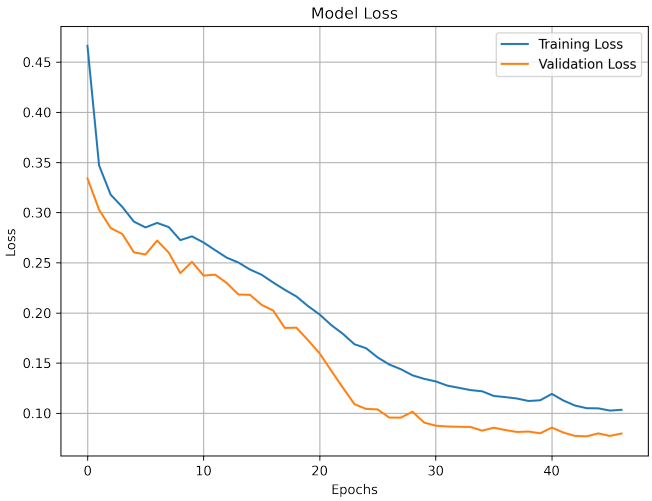


Figure 6.7: Loss Curve for uncontrolled experiment

2. The model's performance on this binary classification task was evaluated on the testing data set, which resulted in a confusion matrix shown in Figure 6.8 that shows the predicted and actual values. The results indicate that the model could accurately classify most positive and negative samples for the generated barrel region data.

3. Also, again, various metrics used to summarize the model's performance on binary classification tasks are shown in Table 6.3, which reveals the model's accuracy was 97%.

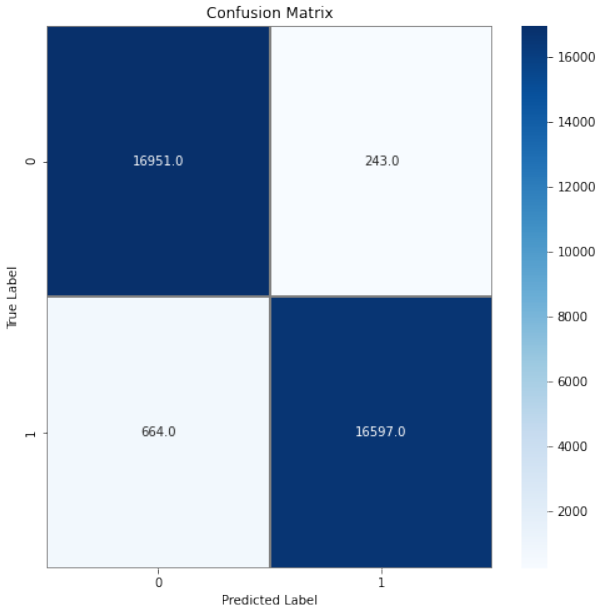


Figure 6.8: Confusion matrix for uncontrolled experiment

Metric	Precision	Recall	F1 Score	Support
γ (Label: 0)	0.97	0.98	0.97	17194
π^0 (Label: 1)	0.98	0.97	0.97	17261
Accuracy			0.97	34455
Macro Average	0.97	0.97	0.97	34455
Weighted Average	0.97	0.97	0.97	34455

Table 6.3: Performace of CNN Classification in Uncontrolled Experiment

We can conclude that our model act as a good classifier on this generated dataset.

6.3 Testing

It is important to note that the dataset used for training the model always contained a clear window of either particle, meaning that no influence of other particles was present in this window. However, in real scenarios, if we test our algorithm, we cannot guarantee the presence of a clear window, as it may or may not contain the effect of other particles. Therefore, additional testing is required to check the efficiency of our trained model. Also, when feeding data for this testing, a separate algorithm is needed to provide a 7×7 window around the seed crystal, as there are multiple π^0 and γ in the barrel region. In such cases, a separate resolving algorithm is required to provide a cleaned window, which is currently beyond the scope of this work. However, for the sake of testing the model's performance

in real scenarios, we simulated scenarios where multiple π^0 and γ exist in the same event. By doing so, we are in a much better position to accurately determine the model's ability to classify particles.

For this testing, we selected a sample of 20000 events containing multiple π^0 or γ, which provided us with 198178 sample images to evaluate the performance of our trained CNN. The testing resulted in an accuracy of 0.81 for our model. These results are presented in the metric Table 6.4 for quick reference.

Metric	Precision	Recall	F1 Score	Support
γ (Label: 0)	0.77	0.88	0.82	98220
π^0 (Label: 1)	0.86	0.74	0.80	99958
Accuracy			0.81	198178
Macro Average	0.82	0.81	0.81	198178
Weighted Average	0.82	0.81	0.81	198178

Table 6.4: Performace of CNN Classification in general

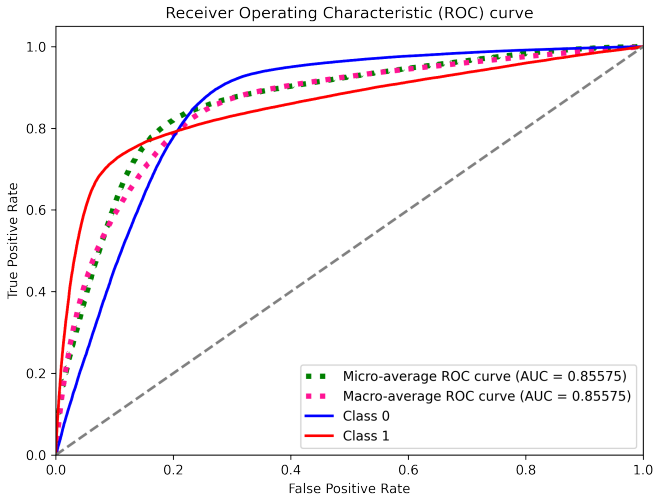


Figure 6.9: ROC Curve to evaluate model performance.

The calculated area under the curve (AUC) was 0.86, which tells us that with this probability, our classifier will be more confident that a randomly chosen positive example (either π^0's and γ) is positive than that a randomly chosen negative example is positive.

Furthermore, Figure 6.10 displays the predicted results of our model for a sample event containing multiple π^0's and γ. The figure depicts the clusters present in the barrel region and the model's probability predictions for each shower, indicating the likelihood of it being a certain particle with a corresponding probability value.

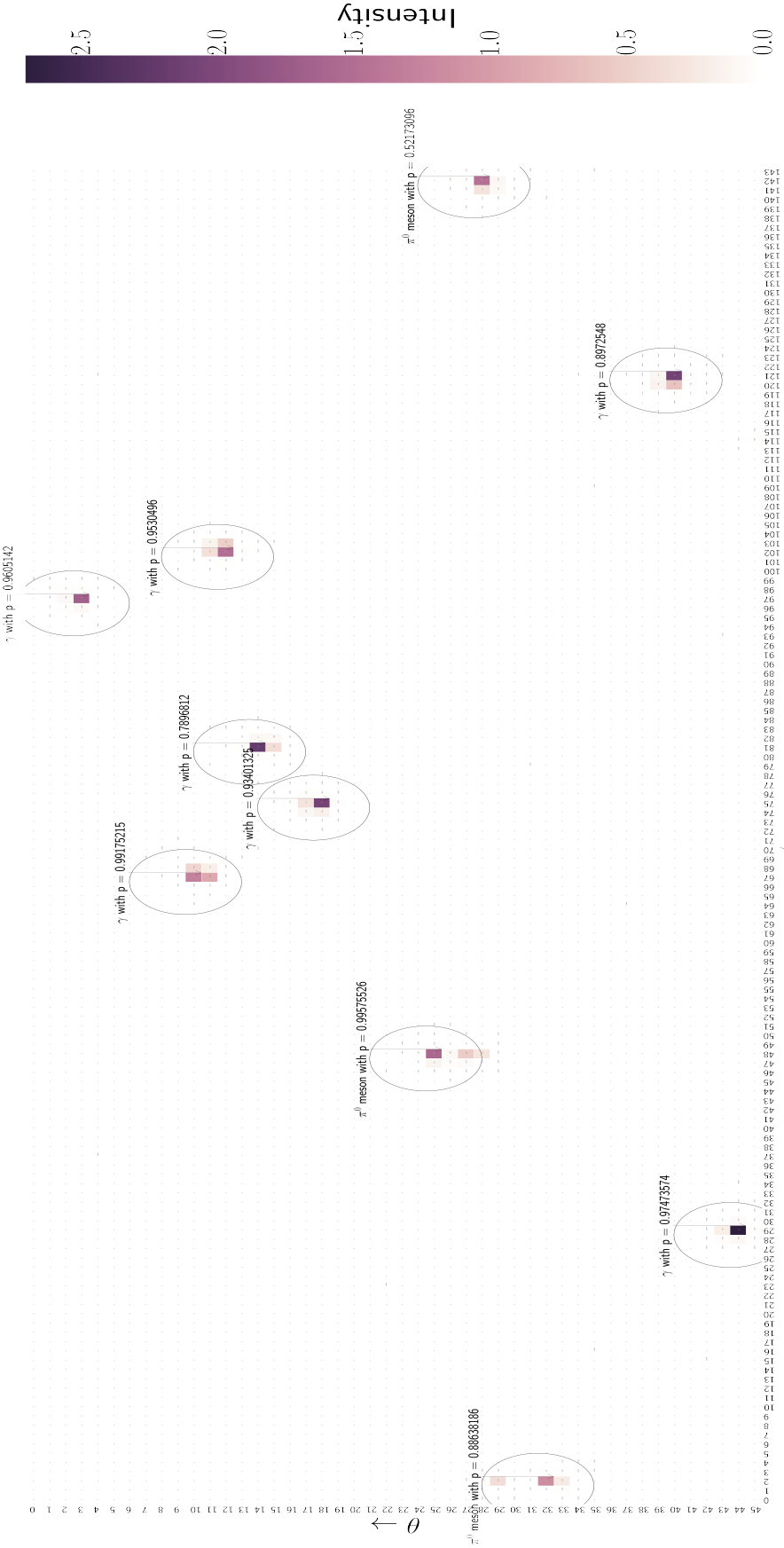


Figure 6.10: Predicted particle with probability (p) by our trained model for each shower of π^0's and γ in the Barrel region of ECL.

Chapter 7

Discusssion & Conclusion

This chapter aims to thoroughly analyze our work and draw conclusive insights based on the results mentioned in the previous chapter. In addition, we will explore potential future implications, identifying areas for improvement and acknowledging the current limitations of our study. Doing so will give us a comprehensive understanding of our research and its implications.

In this work, we try to recover the merged $\pi^{0,}s$ lost in the current framework of `basf2`, where they appear in the form of gammas. Currently, the point of identification for both particles is the ECL detector of the Belle II experiment. This identification is based on variables such as shower shape momenta, $E9/E21$, and Zernike moments. Also, the reconstruction algorithm on two gammas can recover $\pi^{0,}s$. Still, these methods only partially recover most of the merged $\pi^{0,}s$. Since most of these variables rely on energy as the source of information, we have explored the use of ML algorithms to recover these particles. By considering the deposited raw energy information as pixel intensities to form images, the most intuitive algorithm to distinguish these particles appears to us is CNN.

By using CNN and performing the different case studies in Chapter 6, we have satisfying results that leave us with a simple CNN model which have acceptable performance in the defined region. The AUC value, which equates to 0.99, achieved in the controlled setting (i.e., in proximity to a randomly selected crystal) attests to the near-perfect classification performance of our approach in differentiating $\pi^{0,}s$ and γ. This outcome not only validates the efficacy of our proposed methodology but also underscores the potential for CNN to address the problem at hand.

Upon applying our neural network to the barrel region, we achieved an accuracy of 0.97, provided that the training and testing data sets were thoroughly cleaned. It is important to note that this level of accuracy may not be achievable in real scenarios. Hence, towards the end of our research, we assessed the performance of CNN in events that involved multiple $\pi^{0,}s$ and γ. Ultimately, our analysis revealed that the accuracy of the model on such test

data sets was 81%. The reduced accuracy can be accounted for the following reasons:

1. During the model training process, it is possible that we may overlook the impact of particles in certain regions. This is because the 160,790 samples used in the training set may not fully represent the entirety of the region of interest. It would be advantageous to include a wider variety of data samples near each crystal to increase the accuracy of our method.

2. During the neural network training process, we ensure that the input window does not contain the effects of other particles. This initial step helps to achieve high testing accuracy. However, we observed a drop in accuracy when the model was tested on real scenarios. One of the reasons for this drop in accuracy is the occurrence of two γ particles in close proximity, which means our model input considers a region around one maximum that may also contain the maximum of the other γ, leading to the prediction of false π^{0}'s.

While the aforementioned points represent the current limitations of our model, they can be effectively addressed by incorporating a larger number of data points in the barrel region. Additionally, we can deploy an appropriate algorithm that is capable of resolving the images in the second scenario.

Furthermore, it should be noted that our current study is limited to the barrel region of the ECL detector, as the endcaps exhibit irregularities that result in significant energy leakage. Given that our current source of information is based solely on energy, we must maintain the accuracy of our model. However, we can account for the endcaps by incorporating additional variables. It is worth mentioning that the generalization of endcaps will require a substantial amount of data points, which may prove to be a tedious task, but that can also be accomplished within the scope of our current study in the future.

In summary, our model has demonstrated satisfactory classification performance, achieving an accuracy of 86%. Based on these results, we have drawn several conclusions, which are outlined below:

- The present findings demonstrate the potential of machine learning (ML) algorithms in enhancing the efficacy of identifying π^0 particles with energy deposits in the ECL, suggesting that the raw ECL images contain a wealth of information beyond the expert-engineered features currently utilized.

- Also, it forces us to draw a conclusion that CNN performance is superior since it does not rely on the ECL cluster reconstruction algorithm.

Appendix A

Codes: Algorithm & Analysis

The source code and all the analysis performed for this study can be found in the GitHub repository at the following address: `https://github.com/baghelnk10/Master_Thesis`. All the scripts, programs, and upcoming work for this study are accessible through the repository. Readers can reproduce the findings of this study and learn more about the procedures and approaches used in this research through this repository. Additionally, Some necessary sample codes are also given below in the following listings.

Listing A.1: Steering file to generate events

```
import glob
import argparse
import sys
import os
from basf2 import *
from simulation import add_simulation
from reconstruction import add_reconstruction
from reconstruction import *
from modularAnalysis import *



# suppress messages and warnings during processing:
set_log_level(LogLevel.WARNING)
# Create Modules
eventinfosetter = register_module('EventInfoSetter')    #
    Create Event information
eventkinematics = register_module('EventKinematics')
progress = register_module('Progress')                  # Show
    progress of processing
```

```
19 gearbox = register_module('Gearbox')                    # Load
     parameters
20 geometry = register_module('Geometry')                  #
     Create geometry
21 particlegun = register_module('ParticleGun')             # Load
     parameters
22 output = register_module('RootOutput')
23 # Setting the random seed for particle generation
24 set_random_seed(123)
25 # Setting the option for all non particle gun modules: want to
     process 100 MC events
26 eventinfosetter.param({'expList': [0], 'evtNumList': [10000],
     'runList': [0]})
27 eventkinematics.param({'usingMC': True})
28 # Set parameters for particlegun
29 particlegun.param({
30     # Generate pi0 if you want gamma then use [22]
31     'pdgCodes': [111],
32     # Generate no. of particles
33     'nTracks': 0,
34     # But vary the number of tracks according to Poisson
         distribution
35     'varyNTracks': False,
36     # having a uniform transversal momentum
37     'momentumGeneration': 'uniform',
38     # with a fixed momentum of 1 GeV
39     'momentumParams': [2.0,4.0],
40     # use uniform distribution for phi
41     'phiGeneration': 'uniform',
42     # with a mean of 90 degree and w width of 30 degree
43     'phiParams': [-180,180],
44     # and generate theta uniform
45     'thetaGeneration': 'uniform',
46     # between 17 and 150 degree
47     'thetaParams': [33,126],
48     # finally, vertex generation, fixed, we use smearing module
49     'vertexGeneration': 'fixed',
50     'xVertexParams': [0],
51     'yVertexParams': [0],
```

```
52      'zVertexParams': [0],
53      # all tracks should originate from the same vertex
54      'independentVertices': False,
55  })
56  # Set output filename
57  output.param({"outputFileName":
       "gamma_final_testing_events.root", "branchNames": [
                              # Set output filename  # mdst objects
58      'MCParticles',
59      'MCParticlesToECLHits',
60      'MCParticlesToECLSimHits',
61      'BremHitsToECLClusters',
62      'ECLCalDigits',
63      'ECLCalDigitsToMCParticles',
64      'ECLCellIdMapping',
65      'ECLClusters',
66      'ECLClustersToECLShowers',
67      'ECLClustersToExtHits',
68      'ECLClustersToMCParticles',
69      'ECLConnectedRegions',
70      'ECLDiodeHits',
71      'ECLDsps',
72      'ECLDigits',
73      'ECLDigitsToECLHits',
74      'ECLHits',
75      'ECLLocalMaximums',
76      'ECLPidLikelihoods',
77      'ECLShowers',
78      'ECLSimHits']
79  })
80
81  # create processing path
82  main = create_path()
83  main.add_module(eventinfosetter)
84  main.add_module(eventkinematics)
85  main.add_module(particlegun)
86  main.add_module(gearbox)
87  main.add_module(geometry)
```

```
add_simulation(main)                                                    #
    detector and L1 trigger simulation
add_reconstruction(path=main)                                           #
    reconstruction
main.add_module('ECLFillCellIdMapping')
main.add_module(output)
main.add_module(progress)
# generate events
process(main)
print(statistics)
```

Listing A.2: Python code to generate CSV file for training CNN

```
column_list =[]
for i in range(0,49):
    column_list.append("pixel_{f}".format(f=i))
import uproot
import time
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import heapq
import numpy as np
# Record the start time
start_time = time.time()
# Open root file and get the desired branch
file = uproot.open("pi0_final_testing_events.root")   # use
    gamma or pi0 root file generated from steering file
dummy_data = pd.DataFrame(columns = column_list)
ECLLocal_maxCell_Id = file["tree"]["ECLLocalMaximums"]
["ECLLocalMaximums.m_CellId"].array(entry_start=0,entry_stop=10000).tolist
    # Entry_stop is the no. of events stored in root file
ECLCal_CellId = file["tree"]["ECLCalDigits"]
    ["ECLCalDigits.m_CellId"].array(entry_start=0,
    entry_stop=10000).tolist()
ECLCal_Energy = file["tree"]["ECLCalDigits"]
    ["ECLCalDigits.m_Energy"].array(entry_start=0,
    entry_stop=10000).tolist()
mapping = file["tree"]["ECLCellIdMapping"]
    ["m_CellIdToNeighbours7"].array(interpretation=None,
    entry_start=0, entry_stop=1, library='pd').tolist()
```

```
i = 0
while i < len(ECLCal_CellId):
    print(i)
    time_elapsed = time.time() - start_time
    if not ECLCal_Energy[i]:
        i += 1
    else:
        count = 0
        shower = {}
        top_10 = []
        top_10_dict = {}
        for j in ECLLocal_maxCell_Id[i]:
            key = f'key{j}'
            ECLNeighbours_mCellId = mapping[0][j]
            pixel_intensity = []
            for k in ECLNeighbours_mCellId:
                if k in ECLCal_CellId[i]:
                    Cell_index = ECLCal_CellId[i].index(k)
                    centered_Cell_index =
                       ECLCal_CellId[i].index(j)
                    ECLCal_CellEnergy =
                       ECLCal_Energy[i][Cell_index]
                    centered_energy =
                       ECLCal_Energy[i][centered_Cell_index]
                    pixel_intensity.append(ECLCal_CellEnergy)
                else:
                    pixel_intensity.append(0.0)
            if len(pixel_intensity) < 49:
                num_zeros = 49 - len(pixel_intensity)
                for k in range(num_zeros):
                    pixel_intensity.insert(0, 0)
            if len(pixel_intensity) > 49:
                num_zeros = len(pixel_intensity) - 49
                for k in range(num_zeros):
                    pixel_intensity.remove(0)
            image = np.array(pixel_intensity)
            max_energy = max(image)
            if max_energy not in top_10_dict:
                heapq.heappush(top_10, max_energy)
```

```
                top_10_dict[max_energy] = [image]
                if len(top_10) > 10:
                    smallest = heapq.heappop(top_10)
                    del top_10_dict[smallest]
        check_cluster = {}
        for key, value in top_10_dict.items():
            pixel_intensity_centered = CenterImage('Local',
               7)._center_image(value, key).tolist()
            image2d = np.array(pixel_intensity_centered)
            image_norm = (image2d - np.min(image2d)) /
               (np.max(image2d) - np.min(image2d))
            data = pd.DataFrame(image_norm).T
            data.columns = column_list
            data.insert(0, "Label", [1], True)  # use [0] in
               case of gamma
            dataframe = dummy_data.append(data,
               ignore_index=True)
            dummy_data = dataframe
            count += 1
        i += 1
dataframe.to_csv('pi0_last_testing.csv')  # use other name in
   case of gamma
```

The above listing contains some preprocessing transformers that are shown in the below listing.

Listing A.3: Transformer used in preprocessing image data.

```
from sklearn.base import TransformerMixin
from sklearn.base import BaseEstimator
import numpy as np
from scipy import ndimage
class CenterImage(BaseEstimator, TransformerMixin):
    """
    The transformer takes as an input image in form of a
       vector of length n x n and
    applies transformation so that either MAX pixel or Local
       pixel(any pixel that you want) will be in the image
       center (n/2, n/2)

    Parameters
```

```
11      ----------
12      center_around : string
13          translate the image so that 'MAX' pixel or 'Local'
               pixel(any pixel that you want) is in the image
               center
14
15      image_size: int
16          size of the sqauare image (n x n)
17
18      order : integer, optional
19          order of interpolation used in the translation of the
               image (default = 0)
20      """
21      def __init__(self, center_around, image_size, order=0):
22          self.center_around = center_around
23          self.image_size = image_size
24          self.order = order
25
26      def _center_of_maximum(self, data):
27          """Returns the position of the MAX pixel"""
28          maxPos = ndimage.maximum_position(data)
29
30          return maxPos
31
32      def _center_of_local(self, data, value):
33          """Returns the position of the MAX pixel"""
34          pos = np.where(data == value)
35          if isinstance(pos[0], np.ndarray):
36              return tuple(pos[i][0] for i in range(len(pos)))
37          else:
38              return tuple(pos[i] for i in range(len(pos)))
39
40
41      def _center_image(self, image, value):
42          """Perform transformation on 2D image"""
43
44          image2D =
               np.resize(image,(self.image_size,self.image_size))
45          shift=[0.0,0.0]
```

```
        center=(float(self.image_size)/2.,float(self.image_size)/2.)
        if self.center_around == 'MAX':
            myMAX=self._center_of_maximum(image2D)
            shift[0]=center[0]-myMAX[0]-0.5
            shift[1]=center[1]-myMAX[1]-0.5
        elif self.center_around == 'Local':
            mylocal=self._center_of_local(image2D, value)
            shift[0]=center[0]-mylocal[0]-0.5
            shift[1]=center[1]-mylocal[1]-0.5
        else:
            return image2D.ravel();

        image2D =
           ndimage.shift(image2D,shift,order=self.order,
           mode='grid-wrap')

        return image2D.ravel()
```

Listing A.4: CNN Model

```
import itertools
from keras.utils.np_utils import to_categorical # convert to
   one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D,
   MaxPool2D, MaxPooling2D
from keras.optimizers import RMSprop,Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from keras.regularizers import l1, l2
from keras.layers import BatchNormalization
from keras.callbacks import EarlyStopping
# Define the optimizer
optimizer = Adam(lr=0.001, beta_1=0.9, beta_2=0.999)
# Define the CNN model
model = Sequential()
#
model.add(Conv2D(32, (3, 3), padding='same',
   activation='relu', input_shape=(7,7,1)))
model.add(MaxPool2D(pool_size=(1, 1)))
model.add(Dropout(0.5))
```

```
19
20 model.add(Conv2D(64, (3, 3), padding='same',
       activation='relu'))
21 model.add(Conv2D(64, (3, 3), activation='relu'))
22 model.add(MaxPool2D(pool_size=(2, 2)))
23 model.add(Dropout(0.2))
24
25 model.add(Conv2D(64, (3, 3), padding='same',
       activation='relu'))
26 model.add(MaxPool2D(pool_size=(1, 1)))
27 model.add(Dropout(0.1))
28
29 model.add(Flatten())
30 model.add(Dense(128, activation='relu'))
31 model.add(Dropout(0.1))
32 model.add(Dense(2, activation = "sigmoid"))
33
34 # Compile the model
35 model.compile(loss='binary_crossentropy', optimizer=
       optimizer, metrics=['accuracy'])
```

Bibliography

[1] Super KEKB and Belle II. https://www.belle2.org/project/super_kekb_and_belle_ii, 2023.

[2] Tetsuo Abe, I Adachi, K Adamczyk, et al. Belle II Technical Design Report. *arXiv preprint arXiv:1011.0352*, 2010.

[3] T Aushev, W Bartel, A Bondar, et al. Physics at Super B Factory. *arXiv preprint arXiv:1002.5012*, 2010.

[4] Eric Drexler. Elementary Particle Interactions in the Standard Model. https://commons.wikimedia.org/wiki/File:Elementary_particle_interactions_in_the_Standard_Model.png.

[5] T Ferber. Photon and ECL variables. https://confluence.desy.de/display/BI/Photon+and+ECL+variables.

[6] JL Hewett, H Weerts, R Brock, et al. Fundamental Physics at the Intensity Frontier. *arXiv preprint arXiv:1205.2671*, 2012.

[7] Hitomi Ikeda. Development of the CsI(Tl) Calorimeter for the Measurement of CP Violation at KEK B-Factory. https://opac2.lib.nara-wu.ac.jp/webopac/TD00003756, 2014.11.06.

[8] DorisYangsoo Kim and Belle II Software Group. The Software Library of the Belle II Experiment. *Nuclear and particle physics proceedings*, 273:957–962, 2016.

[9] Emi Kou, Phillip Urquijo, W Altmannshofer, et al. The Belle II Physics Book. *Progress of Theoretical and Experimental Physics*, 2019.

[10] Christian Lippmann. Particle Identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors, and Associated Equipment*, 666:148–172, 2012.

[11] Miss MJ. Standard Model of Elementary Particles. https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg.

[12] R. L. Workman et al. Review of Particle Physics. *PTEP*, 2022:083C01, 2022.