PHYSICS-INFORMED NEURAL NETWORKS FOR BELLE II KAON VERSUS PION PARTICLE IDENTIFICATION

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

AUGUST 2020

By

Yannik Glaser

Thesis Committee:

Peter Sadowski, Chairperson
Mahdi Belcaid
Kurtis Nishimura
Kyungim Baek

Keywords: deep learning, particle identification, high-energy physics, neural networks, belle

Copyright © 2020 by
Yannik Glaser

ABSTRACT

Particle identification (PID) is a common problem in high-energy physics experiments conducted with particle accelerators, where some particles cannot be detected directly and rather have to be inferred based on their decay products.

The Belle II experiment will generate particle collision data using the superKEKB accelerator in order to study the properties of B-mesons. Among other things, it will be paramount to be able to accurately and reliably distinguish between two types of particles known as kaons and pions. When the charged particles pass through the iTOP detector component of the accelerator, photons are emitted and channeled into a detector array where their hit locations and times are being recorded. This project seeks to explore new models to better distinguish between kaons and pions based on the recorded photon hit patterns.

To determine the most effective model for this task, we propose and analyze several *physics-informed* deep learning architectures. By constraining the hypothesis space of the models to adhere to known physical rules observed in the data, we hope to improve performance over currently employed methods and gain insights into which properties of the data are most useful to explicitly integrate into our architectures. Specifically, permutation invariance, the ability to overcome signal sparsity, and the assumption of independence between individual photon hits are being analyzed as to their viability as informative modelling constraints. If successful, the resulting models could ultimately be employed for PID in the actual Belle II experimental setup.

All models are trained and evaluated on a dataset of Monte Carlo simulations which is readily available and sufficient to test the viability of these methods for real-world experiments. Our initial pass of experiments with a somewhat simplified subset of the data demonstrates that deep learning outperforms the analytical method currently being employed for PID. Further, the results indicate that sparse representations can yield improved results if known physical properties of the data like permutation invariance are accounted for. These results are promising and show that deep learning can outperform existing analytical approaches. The next step should focus on the wider range of data that such a PID model will encounter in the real detector, where particle trajectories and energies are varied. For future analysis, we propose sampling uniformly from a distribution over this space and training a single deep learning model to see if this approach is still able to outperform the analytical methods for the entire phase space.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Particle identification (PID) is one of multiple challenges in many high-energy physics (HEP) collision experiments. HEP, often also referred to as particle physics, is the branch of physics concerned with the study of elementary particles and their interactions. Research in this domain seeks to answer fundamental questions about the nature of our universe. Particle accelerators and colliders have become a critical tool for particle physicists as they are able to generate unique data about the way elementary particles behave and interact. One of the challenges with this data however, is the fact that most particles cannot be observed directly, but rather, they have to be identified by how they interact with a particle detector. Solving this problem often proves very difficult due to the low signal-to-noise ratio in the data collected.

Deep learning, whose emergent success in the last decade first began in fields like computer vision and natural language processing, has since permeated the physical sciences [2]. Deep learning tends to be more successful than shallow machine learning methods in modeling high dimensional data such as images and video, so while physicists have traditionally relied on manual feature engineering and dimensionality reduction, deep learning offers a method to automatically extract relevant information from high dimensional detector data [11]. This makes it a uniquely well-suited candidate for the problem of particle identification; furthermore, even though labeled data is hard to obtain, the solid theoretical understanding of the physical forward model underlying these experiments allows for the creation of practically unlimited amounts of labeled data from simulations and/or calibration experiments [5][8].

However, even with the intuition that PID problems are good applications for deep learning models, there are still various plausible candidate machine learning algorithms and deep learning architectures that could be employed. In this paper, we attempt to propose architectures and data representations that are informed by our knowledge about physical properties of the data. Specifically, we will constrain our search space to architectures that have one or multiple of the following attributes: permutation invariance, the ability to efficiently represent high-dimensional sparse data, assumption that photon hits are conditionally independent given the particle type. By building more informed models, we hope to improve performance over currently employed models and gain insights into which properties of the data are most useful to integrate into our models.

CHAPTER 2
BACKGROUND

Machine learning has become an increasing presence in particle physics even before the resurgence of deep learning. One popular instance of machine learning aiding in a HEP problem was the use of boosted decision trees to assist in the discovery of the Higgs boson [7]. Since then, deep learning has gained popularity, especially because of its ability to work with the low-level data generated in simulations or experiments as demonstrated by [2] on a variety of benchmarking tasks; this reduces the need for feature engineering, a process usually involving experts in the field deriving meaningful, distilled, representations of the data. Not reducing the data to some engineered representations allows the algorithm to potentially use all of the available data with no intermittent filters to fit the problem. In practice on a novel problem, [3] showed the potential of deep learning for jet substructure classification, a problem somewhat similar to that of particle identification, where they represented low-level signals collected by the particle detector as two-dimensional image-like graphs (hit data graphs) and subsequently used a convolutional neural network (CNN) for the classification task. While this way of encoding the data as a kind of image produced impressive results, it is unclear whether this representation and the choice of convolutional neural networks are also the best approach for particle identification in Cherenkov detector applications.

2.1 Kaon versus pion particle identification in the Belle II experiment

The Belle II experiment is the successor to the Belle experiment that ran from 1998 to 2010 at the High Energy Accelerator Research Organization's (KEK) laboratory. The goal of the Belle II experiment is to study the properties of B-mesons using the SuperKEKB accelerator and ultimately extend and deepen our understanding of the Standard Model of physics [6].

An integral part of the upgraded SuperKEKB accelerator is the imaging Time-of-propagation (iTOP) detector used for particle identification. As charged particles pass through quartz bars on the outside of the iTOP modules, a cone of Cherenkov photons is produced that can then be detected by an array of photodectors at the end of each module (see Figure 2.1). These detectors have highly improved temporal resolution (as little as 50 picoseconds) enabling better PID, specifically for discriminating between kaons and pions [4][10]. Accurately detecting whether the particle resulting from an event was a kaon or pion is of great interest for several reasons. For instance, among other things the Belle II experiment aims to collect precision measurements for rare decays of B-mesons, which requires being able to identify the B-meson decay products with high certainty. In the case of kaon versus pion PID, one common decay of B-mesons is into one kaon and one pion, while a rare decay of interest would be into two pions. Additionally, correctly identifying

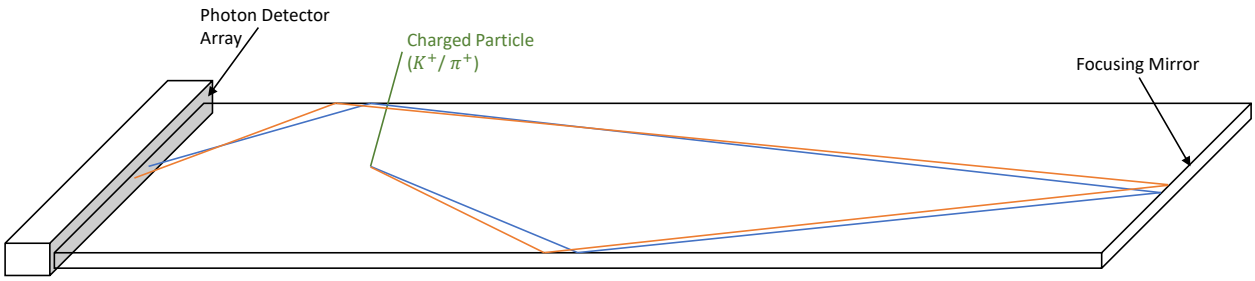

Figure 2.1: Basic illustration of iTOP detector module used for data collection associated with kaon versus pion PID task in the SuperKEKB accelerator. In green and red are photon paths illustrated created by different charged particle tracks.

kaons can be of particular interest because kaons can exhibit charge conjugation parity symmetry (CP-symmetry) violations during their decay. CP-symmetry roughly describes the idea that if a particle is interchanged with its antiparticle ("C" symmetry) while its spacial coordinates are also inverted ("P" symmetry), a symmetry should apply, meaning the same physical laws apply in both situations. This symmetry however, is violated during some weak decays and better understanding this process can be helpful for refining the Standard Model.

PID in the iTOP detector relies on the fact that the cosine angle at which Cherenkov photons are emitted in the detector is inversely related to the speed of the charged particle and the refractive index of the quartz bars [4]. Thus, given the difference in mass between kaons and pions, there is an observable difference in where and when hits are recorded for a given particle track, depending on what the emitting particle was (illustrated in Figure 2.1).

A good illustration of this difference are graphs, which depict hit locations over time, resulting in a so-called "ring image" (Figure 2.2). Analytical models that are currently being employed for this task are built on this knowledge by calculating an extended log likelihood probability for the particle being either a kaon or pion using expected signal and background photon distributions as functions of position and time. For more details on current analytical methods, see [13], [12].

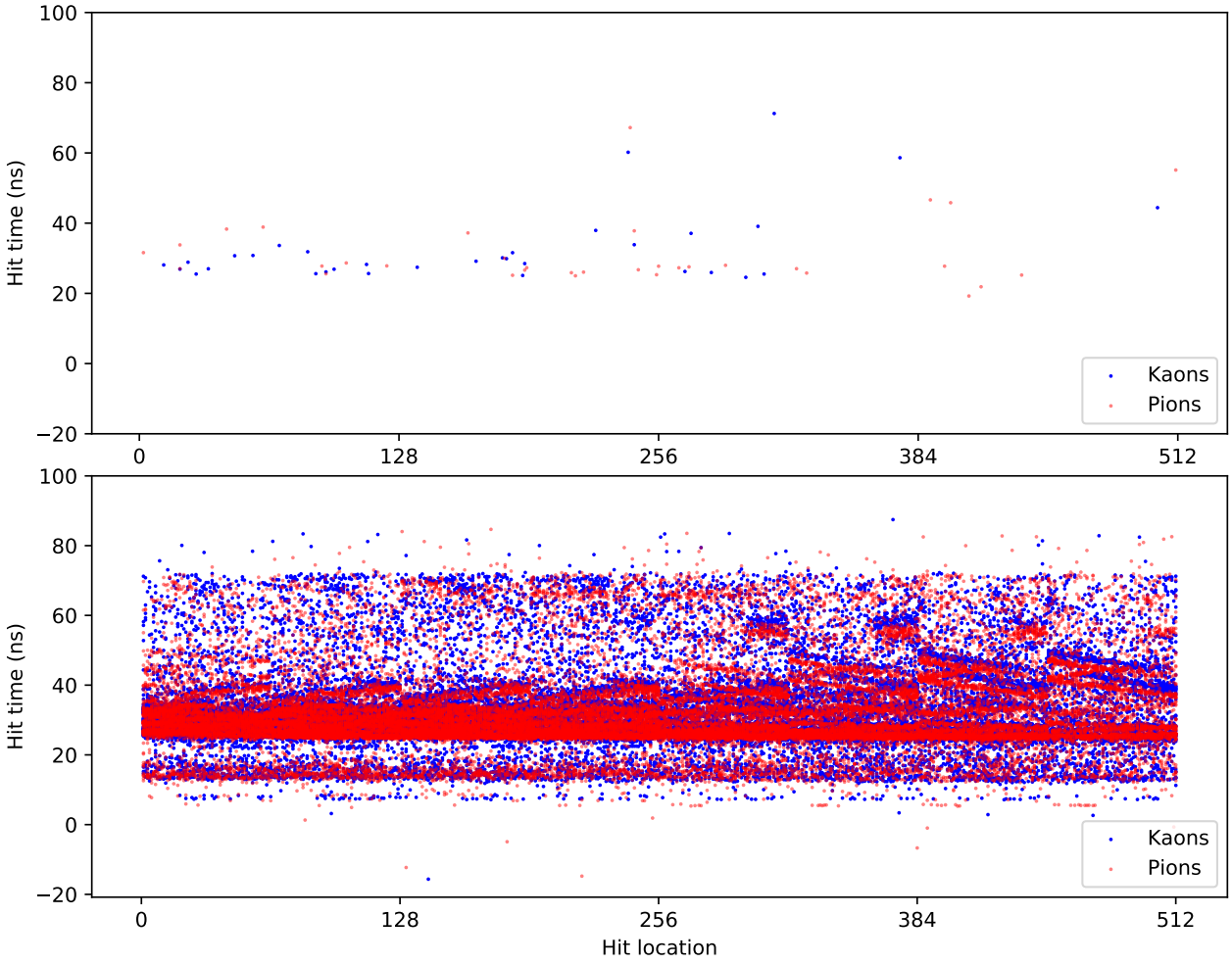

Figure 2.2: Photon hit locations for kaons (blue) and pions(red) over time for a single event (top) and across 1,000 events (bottom) with concatenated slices along the second spatial dimension to reflect the layout of the detector array.

CHAPTER 3
DATA

3.1 Dataset

Our model is trained and evaluated on Monte Carlo simulation data since this is readily available and sufficient to test the viability of PID methods. Simulations are done with a pure kaon and pion particle gun, simulating the particle track and potential decay from the point of collision. These simulations also include some background radiation resulting in a small subset of the detected photons not being emitted from a charged particle passing through the detector module and thus constituting noise. This is in line with some of the real-world noise the models would encounter. The particle gun is locked to one point in phase space, shooting all particle tracks at the angles =60 and =60 at the point of origin and the point of collision with the quartz bar respectively and a momentum of P=1.5GeV. Classification models successfully fit and evaluated on this simulation data can then be extended to a larger portion of the phase space and ultimately potentially integrated in the real Belle II particle identification process. For computational simplicity, simulated events do not encompass the entire physical process of particle collision, but rather only focus of the path and decay of either a single pion or kaon from the point of the collision.

The entire dataset contains 1,000,000 kaon and pion events each. From this, we simplify the task by selecting events with the following properties: (1) Events are limited to only simulations resulting in a single particle track registered in the module, meaning the original kaon or pion particle did not decay further before hitting the module (2); only events with positively charged particles are selected; and (3) our analysis is restricted to events registered on Module 3, where we expect the majority of positively charged particles to hits the quartz bar. This reduces the size of the dataset to 440,275 kaon events and 473,614 pion events.

The resulting data is then randomly split into 60% training data, 20% validation data for tuning hyperparameters like learning rate and batch-size during training, and 20% testing data that is held out for final model evaluation.

3.2 Data representations

Each individual event is represented by the position of the photon hits that are recorded in detector Module 3 as well as the times at which they are recorded. There are two approaches to representing this data: as a dense "image" or as a sparse list of hits. We discuss the advantages and disadvantages of each below.

3.2.1 Hit-grid approach

The first approach is to represent the hit data as an image on a two-dimensional grid. Because of the layout of the detector modules, these grids are 64 horizontal cells by 8 vertical cells, giving an overall spatial resolution of 512 pixels. Each pixel then represents a count of hits detected in that region within a given time step. Figure 3.1 depicts the resulting hit-grids for a random single event, as well as the normalized hit distributions over 50,000 events per class for both particle classes. The respective top graphs for each particle represent the roughest possible temporal resolution with a single grid for the entire event. Instead, hits can be divided into different time-bins, thereby creating a sequence of grids, comparable to a video.

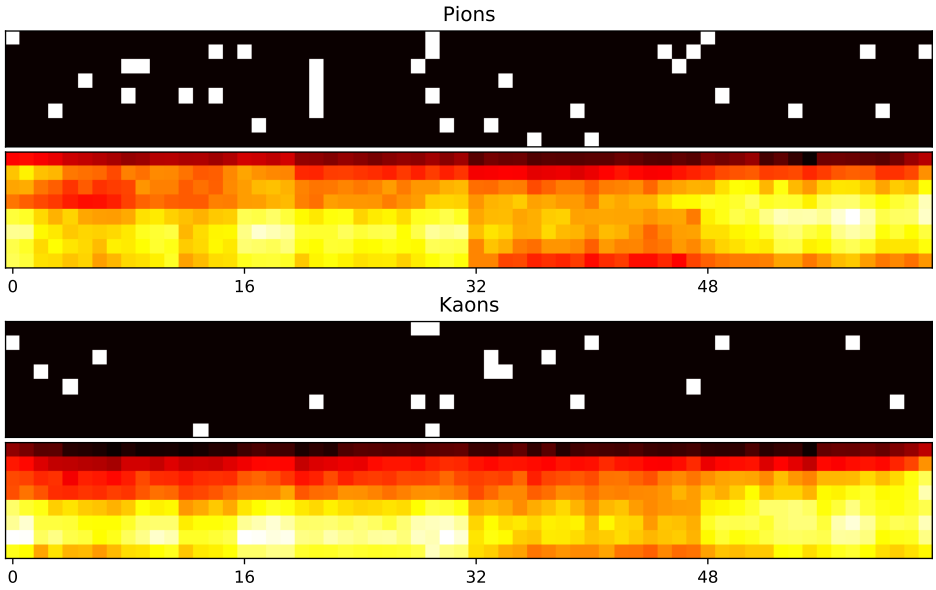

Figure 3.1: Hit-grid representations for both particle classes for single particle events (respective top graphs) and averaged across 50,000 events (respective bottom) across the entire event duration.

Two different strategies for selecting the time bins are compared. In the first strategy, events are binned at regular time intervals; this method is the most analogous to actual video data. The time bin size is calculated based on the longest observed event in our training data which lasted 141 nanoseconds (ns). Thus, for instance, if each event is to be split into 10 time bins, the resulting encoding would consist of 10 consecutive grids, each containing the information for hits collected over 14,1 ns. While this time-binned hit-grid approach has some merits, specifically the fact that it caters to already existing computer vision algorithms by using essentially the same data format, it does come at a cost. There is a trade-off between time resolution and data sparsity. To make the maximum amount of information available to the model, one would have to choose the maximum possible time resolution for the data, which, for the Belle II iTOP detector, would be 50ps. This however, is not only computationally not feasible (since it would result in 2,820,000 individual

grids for each event), but would also mean that the vast majority of those grids are blank, with the average event only registering circa 33 hits (see the top of Figure2.2). Even the grids containing hits would not exceed two hits per grid based on our training data, making them extremely sparse as well. On the other hand, while choosing larger time bins combats that problem, it means losing information about timing.

The second strategy is to use irregularly-spaced bins based on the empirical hit distribution. To calculate the borders for all bins, all hits and their timing recorded in the training dataset are divided up such that each bin represents a quantile of the data, containing the same number of hits. Compared to the previous approach, while this still contains some timing information since the grids are still ordered sequentially, it removes most information on when the hits occur, since time bins are no longer equally spaced. The main advantage of this approach is that the information on hit location is more equally distributed between the grids, perhaps making it easier for the algorithm to find relevant signals in the hit locations. Refer to Figure 3.2 for a graphical representation of how these two binning strategies differ. Also note that due to some of the background radiation in the simulation causing the detector to record photons before the actual charged particle is generated ($t = 0$ in Figure 3.2) there are hits at "negative" times.

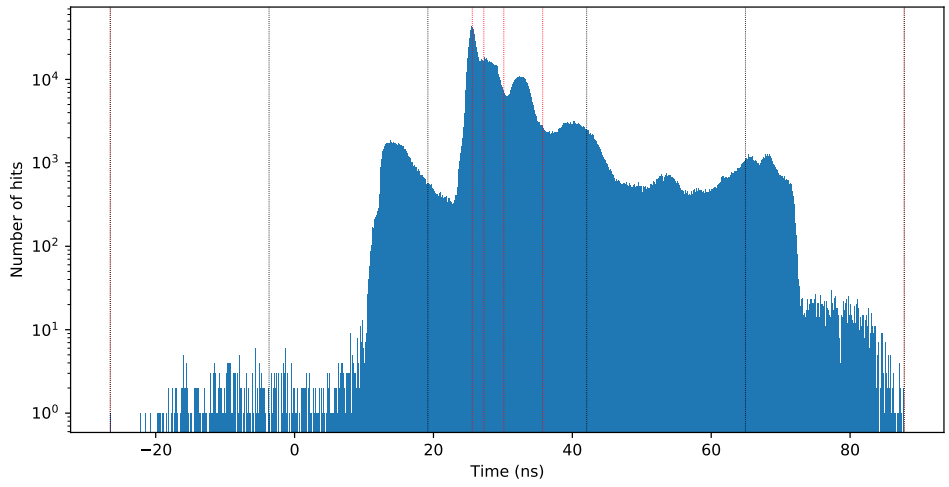

Figure 3.2: Different binning approaches for the hit-grid strategies and the amount of hits they would include. Shown in red are quantile borders and in grey are the borders based on the equidistant binning approach drawn on the training dataset for five bins.

3.2.2 Hit-triplet approach

As mentioned in the previous section, representing the data as time-binned hit-grids comes with an inherent trade-off of precision and computational efficiency. This trade-off can be avoided using a *sparse* representation, where an event is encoded as a sequence of photon hits.

To that end, each hit is represented as a vector of three values: location on the x-axis, location on the y-axis, and the time at which the hit was recorded. This approach still includes the same amount of spatial information as the previous approach, albeit in a different encoding, but allows for much more fine-grained temporal resolution. In this approach, each event can be represented as a sequence of triplets whose length depends on the number of hits recorded for the respective event.

CHAPTER 4
MODELS

In this section we describe different model architecture choices for the two data representations. All of these models are also evaluated against the previously mentioned extended likelihood method proposed by [12], the analytical approach specifically developed for the TOP PID problem.

4.1 Model architectures

4.1.1 Hit-grid models

Given the likeness of this data to regular image data, specialized architectures like CNNs or locally-connected networks are standard models for processing this data. Additionally, CNNs have the additional desirable property of local connectivity, giving them the ability to summarize spatial and temporal correlation in the data. The same CNN architecture is used on both, the quantile and uniformly spaced binning approaches for testing. To test this approach, we evaluate a relatively small CNN consisting of two convolutional layers, the first of which has 30 filters and a 4x4 kernel and the second has 10 filters and a 2x2 kernel. The resulting output tensor is the passed through a max-pooling layer with a 2x2 pooling size (halving the size of the output), followed by a one-filter, 1x1-kernel, convolutional layer for dimensionality reduction. The resulting tensor is then passed through three 512-unit dense layers and ultimately a sigmoid unit produces the final output (Figure 4.1). All hidden and convolutional layers use the rectified linear activation function $max(0, x)$ (ReLU).

The choice of the CNN architecture automatically introduces a limited amount of translational invariance into our model. Given our knowledge of the data, it is unclear whether this is a desirable attribute for our model to have. The data representation, assuming a reasonably low resolution, allows for smaller models which reduces the risk of overfitting and helps reduce the computational effort necessary for training the model.

While the image-like representation of the hit grid approaches is a common way to solve this problem, there are certain shortcomings that we seek to address with this set of models. First of all, while the binning does decrease data sparsity, as mentioned in section 3.2.1, it comes at the cost of reducing temporal resolution. One of the great strengths of the iTOP detector is its stellar temporal resolution, which is largely sacrificed using a coarse time resolution. Furthermore, hyperparameter choices and data distribution can have unforeseen impacts on model performance. Depending on how bin sizes are chosen and what strategy is selected for binning, different hits can fall into different bins, drastically altering the signal. If there are differences between training and testing data, for instance in timing or how the data is distributed, hits that are close to the border

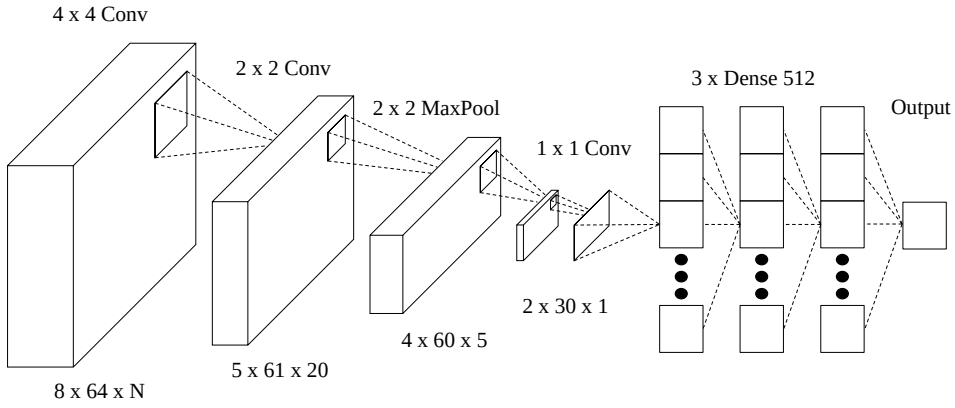

Figure 4.1: Hit-grid CNN architecture with N being the number of grids in the input representation.

between two bins might have a large impact on the final signal. These issues are addressed with the next group of models.

4.1.2 Hit-triplet models

The hit-triplet approach seeks to solve some of the previously mentioned shortcomings while also being conveniently computationally efficient compared to having to create the hit-grid representation for each event in the training data. To see how effective this representation is, a regular feed-forward neural network, a Siamese neural network, and lastly a neural network that enforces the hit-independence assumption (henceforth referred to as HINN) are evaluated using this representation.

Feed-forward neural network

The"vanilla" feed-forward neural network (vanilla NN) consists of 6 ReLU hidden layers (2x256 units followed by 4x128 units) and a standard sigmoid output layer (Figure 4.2).

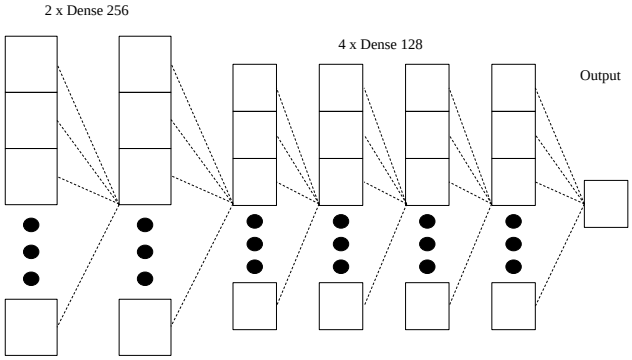

Figure 4.2: Hit-triplet feed forward neural network architecture.

This model does not use of prior knowledge about the physics data and serves as a simple baseline for this data representation. It does not fulfill any of the constraints mentioned in the introduction, aside from overcoming signal sparsity because of the nature of its input data, with the caveat of requiring padding for events with few recorded hits, which will be detailed in section 4.2.1.

Siamese neural network

Meanwhile, the Siamese neural network approach consists of two distinct parts. The first part of the network has the same structure of the previous network (2x256 + 4x128 ReLU layers); this part processes each triplet individually and is trained independently. The resulting output is a 128-dimensional encoding for all triplets in the sequence. This encoding matrix is then mean-pooled into a single 128-dimensional vector which is passed to the second part of the network: 4 more ReLU layers (3x256 + 1x128) and ultimately a sigmoid output unit to produce the final prediction for each event (Figure 4.3).

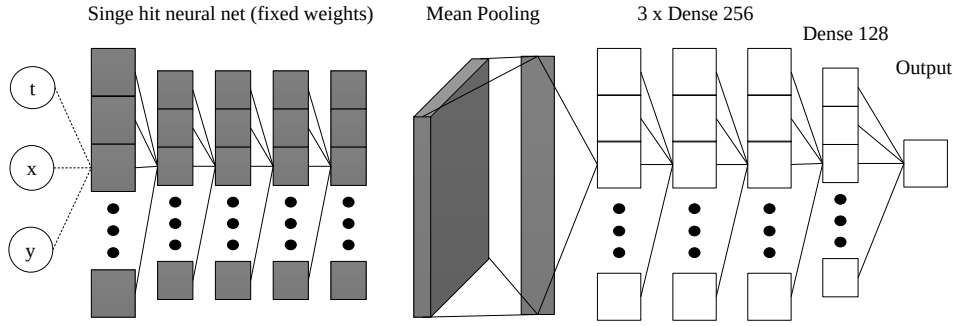

Figure 4.3: Siamese neural network architecture. Part one of the network in grey, part two in white.

Unlike the vanilla NN, this model not only has the advantage of the sparse representation, without the need for padding, but also enforces permutation invariance between the individual hits, meaning the ordering of hits is not explicitly considered in the model.

Neural network assuming hit-independence

Lastly, the HINN can be used to explicitly incorporate the physics knowledge of individual hit-locations within an event being independent. We know this assumption to be true because of the nature of the simulation data. All photons in an event are independently assigned random wavelengths (within a range defined by the quartz bar material properties) which dictates the Cherenkov angle. Combining this with the known azimuthal angle at which the photon was generated then results in the final hit location of the photon on the detector array, which is not influenced by other photons in the event. For prediction, we use the following function to correctly compute the class probability from a set of n predictions made by the single-hit neural network:

$$\begin{aligned}\text{logit}\left(p(P=1|\{H_i\}_n)\right) = \log\left(\frac{p(P=1)}{p(P=0)}\right) \\ + n\log(\lambda_1) - \lambda_1 - n\log(\lambda_0) + \lambda_0 \\ + \frac{1}{T}\sum_{i=1}^{n}\left(\text{logit}(q(P=1|H))\log\left(\frac{q(P=1)}{q(P=0)}\right)\right)\end{aligned} \tag{4.1}$$

with n being the number of hits recorded for a given event, λ_0 and λ_1 being maximum likelihood estimate Poisson parameters for the two particle classes, and T being the temperature for scaling. A detailed derivation for equation 4.1 is provided in the appendix. The same neural network model as for the first part of the Siamese network can be slightly modified with an added sigmoid output unit to produce the probability estimate $p(K|H_i, N=1)$ (Figure 4.4). The ability of the neural net to smoothly interpolate over space and time when producing this probability estimate is essential for overcoming the challenge presented by the very high temporal resolution for standard counting approaches for estimated the photon PDFs, which requires both, a great computational effort as well as vast amounts of storage to simulate enough data for informative PDF estimates.

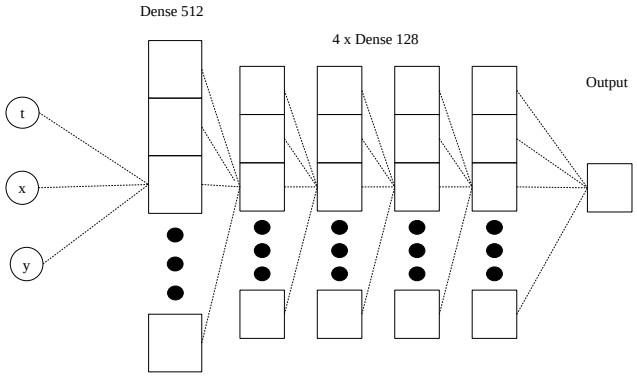

Figure 4.4: Single-hit neural network with sigmoid output.

When combining the individual probability estimates according to 4.1, this classification approach intrinsically makes an assumption of independence between the hits of an event, which is perfect for the data of which it is known that the hit locations are indeed independent. Furthermore, it is also permutation-invariant, and, like the Siamese neural network, does not require any padding of the data, making it an overall strongly-informed model.

4.2 Model training

All models were implemented using the standalone KERAS package in PYTHON 3.7 with the TENSORFLOW [1] backend.

4.2.1 Hit-grid models

For this representation, two separate sets of models were evaluated, one for the quantile-based bins and one for the equally-spaced bins. For both kinds of inputs, models on 10, 20, and 50 bins per event were tested, to represent a reasonable range of bin-size choices. Additionally, a base-model was trained on a single hit-graph that contained all hit information for a given event; representing an event as a single graph is identical for both binning approaches and serves as a control for the amount of improvement that can be gained by increasing resolution with different binning strategies. The models were trained for 10 epochs each, using the Adam [9] optimizer with the default parameters on the binary cross-entropy loss function. Training was done in mini-batches of 512 events.

4.2.2 Hit-triplet models

Feed-forward neural network

For the feed-forward neural network, in order to keep dimensionality constant across all events, padding was done based on the longest event observed in the training data. Thus, all events that were shorter than 979 recorded hits were zero-padded to reach that length. This padding approach has shown better performance than a combination of padding and truncating events based on, for instance, median number of recorded hits. The input vectors were concatenated into a flat 2937-dimensional vector and then fed into the model. The models were trained in mini-batches of 4096 events, again using the default-configuration Adam optimizer to minimize the binary cross-entropy loss. Training was done for 50 epochs.

Siamese neural network

The Siamese neural network had to be trained in two separate steps. Unfortunately, due to limitation of the KERAS package, end-to-end training was not possible, however, as will be discussed in the results section, this seems to be of minor impact on the performance. In the first training step, the actual weight-shared portion of the network is trained independently for individual hits. For this purpose, all recorded hit triplets are randomly shuffled and the target for each individual hit is the particle class that its origin event belonged to. This results in a training data set consisting of 16,865,660 individual hit triplets for this step. Training is done in mini-batches of 9092 hits again using the Adam optimizer with the recommended parameterization to minimize the binary cross-entropy loss. Additionally, the learning rate is halved from the initial 0.01 everytime the validation loss does not decrease for three epochs. Training is stopped either when no improvement of the validation loss is observed for ten epochs or after 100 epochs of training. In order to effectively train the network, an additional sigmoid output layer is added to the 128-dimensional final layer of this portion of the network. This temporary output layer is removed before the second training

step for this model. After this step is concluded, weights for this part of the network are locked and remain unchanged during the next training step. Here, training resumes to be on an event-level; all hit triplets for a given event are passed through the Siamese weight-shared portion of the network in parallel and then pooled and finally processed by the second part of the network. Training is done with the same hyperparameters as before, again for 100 epochs or until the validation loss does not improve for 10 epochs.

Neural network assuming hit-independence

Lastly, the HINN classification model requires less training effort. After finding the prior class probabilities of kaons and pions in the training set, which are roughly 50% each, the Siamese network trained during the first step of the previous models' training, including the sigmoid output layer, is used to assign a likelihood to each individual hit of a given event for belonging to either one of the two particle classes.

CHAPTER 5
RESULTS AND DISCUSSION

Model	Accuracy	AUROC
Baseline		
Analytical likelihood model	89.9%	0.91
Hit-grid Models		
Single-grid CNN (1 bin)	73.9%	0.81
Time-bin CNN (10 bins)	75%	0.83
Time-bin CNN (20 bins)	75.1%	0,82
Time-bin CNN (50 bins)	74.7%	0.82
Quantile-bin CNN (10 bins)	84,8%	0.92
Quantile-bin CNN (20 bin)	87.8%	0.95
Quantile-bin CNN (50 bins)	91.8%	0.97
Hit-triplet Models		
Feed-forward neural network	71.6%	0.79
Siamese neural network	**96.9%**	**0.996**
HINN model	95.6%	0.986
Single-hit models		
Siamese base model	68.7%	0.76

Table 5.1: Overview of performance of all models on testing set. The single-hit model is included for completeness but is not evaluated on the same task.

Multiple architectures ended up outperforming the baseline prediction model in both metrics, showing performance gains similar to other applications of deep learning in HEP [2, 11, 5].

Table 5.1 details the performance of all of our models with the overall strongest model being the Siamese neural network scoring 96.9% accuracy and an almost 1.0 area under the receiver operating characteristic curve (see Figure 5.1). These results show that taking advantage of the permutation invariance of the hits through a physics-informed neural network architecture is important for good performance, as it is only with these properties that we see significant performance improvements over the baseline.

The HINN classifier, similarly to the Siamese neural network also using the triplet data representation, ended up being the second-most accurate model with 95.6% testing accuracy and a 0.99 AUROC score.

The quantile-binning approach also yields strong results. It presents a clear improvement over the single hit-grid and baseline models and adding additional resolution also comes with further increased performance. The best model for this data representation is able to beat the baseline model in both accuracy (91.8%) and AUROC (0.97) score.

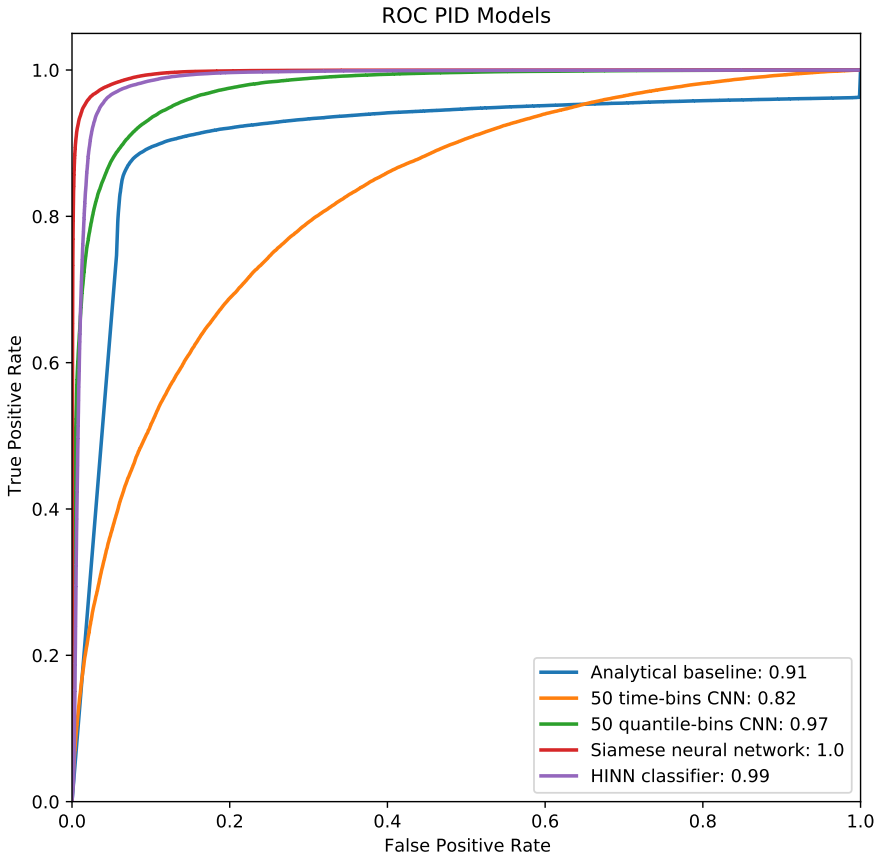

Figure 5.1: Test dataset receiver operating characteristic curve for the Siamese neural network.

5.1 Discussion

As expected, our results demonstrate that time binning has a negative effect on model performance while using sparse representation can yield improved results if known physical properties of the data like permutation invariance are accounted for.

Regarding the triplet representation, results are very encouraging. Both our strongest models utilize this representation. The performance discrepancy between the Siamese neural network and the HINN approach is also intriguing. Both models are based on the single-hit triplet neural network (which has a 68.7% accuracy in classifying the particle of origin for individual hits), with the full Siamese model using feature vectors produced by the penultimate layer, while the HINN model uses the class probabilities from the sigmoid output unit. The HINN model combines all hit predictions under an independence assumption which is in agreements with the underlying physics and the simulation parameters; the fact that the Siamese neural network nevertheless outperforms the HINN model slightly indicates that it gains an advantage by combining information from multiple hits in a way that does not assume feature independence, we suspect that this may be caused by

its ability to account for uncertainty in the single-hit neural network predictions which the HINN is unable to do.

While these results show deep learning performing better than the analytical method for this particular use case, analysis in the real detector needs to be able to handle particle tracks from multiple directions and energies. The next step is to extend the data set to cover a larger portion of the phase space which is the ultimate challenge for integrating these PID models into the Belle II detector. This means lifting some restrictions of particle track direction and momentum currently in place for the particle gun simulation and sample events from the entire phase space. Since we have demonstrated the ability of a neural network to interpolate over the spatiotemporal dimensions for a fixed point in phase space. With future analysis, we hope to show that is can also interpolate over phase space dimensions. The initial findings presented here will serve as a foundation to narrow down the models to test for the broader dataset. If performance in future testing remains similarly strong, these algorithms could become viable options improve the analysis pipeline in the Belle II experiment.

APPENDIX A
DERIVATION OF HIT INDEPENDENT NEURAL NETWORK CLASSIFIER(HINN)

A.1 Hit Independence

The hit independence assumption is that the locations of the hits are independent conditioned on particle ID and the number of hits. Let random variables N be the number of hits and H_i be the location and time of the i-th hit (according to some arbitrary labeling of the electrons). Then, letting $P \in \{0, 1\}$ be the particle type,

$$p(H_1, H_2, ..., H_n|P) = p(N = n|P)p(H_1, H_2, ..., H_N|P, N) \tag{A.1}$$

$$= p(N = n|P) \prod_{i=1}^{n} p(H_1 = H_i|P, N = 1) \qquad \text{(Independence)} \tag{A.2}$$

Because the ordering of the photons was arbitrary, the probability of observing the unordered *set* of locations and times $\{H_i\}_n$ can be obtained by multiplying by the number of permutations $n!$.

$$p(\{H_i\}_n|P) = n!p(N = n|P) \prod_{i=1}^{n} p(H_1 = H_i|P, N = 1) \tag{A.3}$$

A.2 Estimating Hit Count Distribution

Because the hits are assumed to be independent conditioned on the particle type, the class-conditional distribution of hit count $p(N = n|P)$ is a Poisson distribution. For each particle type, the maximum likelihood estimate of the Poisson parameter λ is computed, which is just the average hit count per event. The p.m.f. $p(N = n|P)$ is then given by

$$p(n|P = 1) = \frac{\lambda_1^n e^{-\lambda_1}}{n!} \tag{A.4}$$

Later, we will use the relationship

$$\log\left(\frac{p(N = n|P = 1)}{p(N = n|P = 0)}\right) = \log\left(\frac{\lambda_1^n e^{-\lambda_1}}{\lambda_0^n e^{-\lambda_0}}\right) \tag{A.5}$$

$$= n\log(\lambda_1) - \lambda_1 - n\log(\lambda_0) + \lambda_0 \tag{A.6}$$

A.3 Single Hit Classifier

Now, we can train a model to classify particles from a single *hit* by sampling uniformly from the set of all hits from all events in the training set, the single-hit neural network. This model will approximate the probability of a hit being from a certain particle type. Because we sample *hits* rather than *events*, we denote this probability as $q(P = 1|H)$. The *logit* can be written as

$$\begin{aligned}\text{logit}(q(P=1|H)) &= \log\left(\frac{q(P=1|H)}{1-q(P=1|H)}\right) \quad \text{Definition} &\text{(A.7)}\\ &= \log\left(\frac{q(H|P=1)q(P=1)}{q(H|P=0)q(P=0)}\right) \quad \text{Bayes'Rule} &\text{(A.8)}\end{aligned}$$

The prior $q(P)$ can be estimated from the fraction of total hits from each particle type, and $q(H|P) = p(H_1|P, N = 1)$. Thus we can compute the quantity

$$\begin{aligned}\log\left(\frac{p(H_1|P=1,N=1)}{p(H_1|P=0,N=1)}\right) &= \log\left(\frac{q(H|P)}{q(H|P)}\right) &\text{(A.9)}\\ &= \text{logit}(q(P=1|H)) - \log\left(\frac{q(P=1)}{q(P=0)}\right) &\text{(A.10)}\end{aligned}$$

A.3.1 Putting it all together

The ultimate goal is to estimate the probability that $P = 1$ given a set of n hits from a single event. Bayes Rule gives

$$\begin{aligned}p(P=1|\{H_i\}_n) &= \frac{p(\{H_i\}_n|P=1)p(P=1)}{p(\{H_i\}_n)} \qquad \text{(Bayes)} &\text{(A.11)}\\ &= \frac{1}{1+\frac{p(\{H_i\}_n|P=0)p(P=0)}{p(\{H_i\}_n|P=1)p(P=1)}} &\text{(A.12)}\\ &= \frac{1}{1+\frac{p(P=0)}{p(P=1)}\frac{p(n|P=0)}{p(n|P=1)}\prod_{i=1}^{n}\frac{p(H_i|P=0,N=1)}{p(H_i|P=1,N=1)}} \qquad \text{(Independence)} &\text{(A.13)}\end{aligned}$$

Here, $p(P)$ is the event-level prior, $p(n|P)$ is the distribution of number of hits conditioned on particle type, and terms in the product can be estimated using single-hit neural network. For computational stability, we compute the logit,

$$\text{logit}\left(p(P=1|\{H_i\}_n)\right) = \log\left(\frac{p(P=1)}{p(P=0)}\frac{p(n|P=1)}{p(n|P=0)}\prod_{i=1}^{n}\frac{p(H_i|P=1,N=1)}{p(H_i|P=0,N=1)}\right) \tag{A.14}$$

$$\begin{aligned}
&= \log\left(\frac{p(P=1)}{p(P=0)}\right) \\
&\quad + n\log(\lambda_1) - \lambda_1 - n\log(\lambda_0) + \lambda_0 \\
&\quad + \sum_{i=1}^{n}\text{logit}(q(P=1|H)) \\
&\quad - n\log\left(\frac{q(P=1)}{q(P=0)}\right)
\end{aligned} \tag{A.15}$$

A.3.2 Temperature Scaling

A common regularization method when combining probabilities is to "temperature scale" the probabilities before combining them, which moves them away from extreme values. In this case, we can scale the predictions for the individual hits. One problem that arises is that the predictions for the individual hits are not performed on a balanced training set, thus temperature scaling should be performed with the prior information included. This can be done easily in logit space using the following transformation where T is the temperature, and $T = 1$ involves no scaling.

$$\begin{aligned}
\text{logit}\left(p(P=1|\{H_i\}_n)\right) &= \log\left(\frac{p(P=1)}{p(P=0)}\right) \\
&\quad + n\log(\lambda_1) - \lambda_1 - n\log(\lambda_0) + \lambda_0 \\
&\quad + \frac{1}{T}\sum_{i=1}^{n}\left(\text{logit}(q(P=1|H))\log\left(\frac{q(P=1)}{q(P=0)}\right)\right)
\end{aligned} \tag{A.16}$$

BIBLIOGRAPHY

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016.

[2] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1), Jul 2014.

[3] Pierre Baldi, Kevin Bauer, Clara Eng, Peter Sadowski, and Daniel Whiteson. Jet substructure classification in high-energy physics with deep neural networks. *Physical Review D*, 93(9), May 2016.

[4] Matthew Barrett. Particle identification with the iTOP detector at Belle-II. In *The Meeting of the American Physical Society Division of Particles and Fields, DPF 2013, Santa Cruz, CA, USA, August 13–17, 2013, Conference Proceedings*, 2013.

[5] Dimitri Bourilkov. Machine and deep learning applications in particle physics. *International Journal of Modern Physics A*, 34(35):1930019, Dec 2019.

[6] Belle II collaboration. Belle II experiment homepage.

[7] CMS Collaboration. A new boson with a mass of 125 gev observed with the cms experiment at the large hadron collider. *Science*, 338(6114):1569–1575, 2012.

[8] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2020.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[10] K. Nishimura. The time-of-propagation counter for belle II. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 639(1):177–180, May 2011.

[11] Peter Sadowski and Pierre Baldi. *Deep Learning in the Natural Sciences: Applications to Physics*, pages 269–297. Springer International Publishing, Cham, 2018.

[12] M. Starič. Pattern recognition for the time-of-propagation counter. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 639(1):252 – 255, 2011. Proceedings of the Seventh International Workshop on Ring Imaging Cherenkov Detectors.

[13] M. Starič, K. Inami, Peter Križan, and Toru Iijima. Likelihood analysis of patterns in a time-of-propagation (top) counter. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, pages 252–255, 09 2008.