Hadronic Tag Sensitivity Study of $B \to K^{(*)}\nu\bar{\nu}$ and Selective Background Monte Carlo Simulation at Belle II

James Meier Samuel Kahn

München 2019

Hadronic Tag Sensitivity Study of $B \to K^{(*)}\nu\bar{\nu}$ and Selective Background Monte Carlo Simulation at Belle II

James Meier Samuel Kahn

Dissertation
an der Fakultät für Physik
der Ludwig–Maximilians–Universität
München

vorgelegt von
James Meier Samuel Kahn
aus Australien

München, den 14.02.2019

Erstgutachter: Prof. Dr. Thomas Kuhr
Zweitgutachter: Prof. Dr. Wolfgang Dünnweber
Tag der mündlichen Prüfung: 05.04.19

Zusammenfassung

Die Prozesse $B \to K^{(*)}\nu\bar{\nu}$ bieten sich als ausgezeichneter Test für das Standardmodell an, da sie Flavour-Changing Neutral Currents beinhalten und daher stark unterdrückt werden. Das Belle II Experiment am e^+e^- Beschleuniger SuperKEKB in Japan wird voraussichtlich fünfzig mal mehr Daten als das Vorgängerexperiment Belle liefern. Bei dieser Datenmenge wird erwartet, dass diese und andere bisher noch nicht beobachtete Prozesse zum ersten Mal messbar sind. Die Prozesse $B \to K^{(*)}\nu\bar{\nu}$ sind interessant aufgrund ihrer sauberen theoretischen Vorhersagen und ihrer Verbindung zu den präzise messbaren $B \to K^{(*)}\ell^+\ell^-$ Prozessen, die in Spannung zu Vorhersagen des Standardmodells stehen. In der ersten Hälfte dieser Dissertation schätze ich die Sensitivität auf $B \to K^{(*)}\nu\bar{\nu}$ ab. Mit dem Belle II Rekonstruktionsalgorithmus *Full Event Interpretation* rekonstruiere ich die volle Zerfallskette der e^+e^- Zerfallsprodukte. Anschließend entwickle ich ein Verfahren, um das Verzweigungsverhältnis zu bestimmen und schätze die Sensitivität als eine Funktion der Menge der erfassten Daten ab.

Zur Vorbereitung auf Messungen von seltenen oder schwer zu messenden Prozessen, wie beispielsweise $B \to K^{(*)}\nu\bar{\nu}$, ist eine Simulation aller physikalischen Prozesse an Belle II mindestens im Umfang des später aufgezeichneten Datenvolumens erforderlich. Mit der aktuell verfügbaren Rechenleistung ist dies jedoch effektiv nicht möglich. Ohne eine signifikante Verbesserung der Simulationsgeschwindigkeit erfordert dies daher die selektive Simulation von Kollisionsereignissen, die für derartige Messungen relevant sind. In der zweiten Hälfte dieser Arbeit entwickle ich daher ein Verfahren, um die Wahrscheinlichkeit, dass das simulierte Kollisionsereignis für die Analyse relevant ist, zu berechnen. Diese Vorhersage wird mittels eines neuronalen Netzes bereits vor den rechenintensiven Phasen des Simulationsverfahrens getroffen. Ich zeige, dass mittels dieser Methode deutliche Einsparungen bei den Simulationszeiten erreicht werden können. Das Ergebnis ist ein volsltändiges Trainings und Inferenzverfahren, das sich direkt in das bestehende Belle II Software Analysis Framework einfügt.

Abstract

The $B \to K^{(*)}\nu\bar{\nu}$ processes provide an excellent probe of the Standard Model as they involve flavour-changing neutral currents and are therefore greatly suppressed. The Belle II experiment, hosted at the e^+e^- collider SuperKEKB in Japan, is expected to collect fifty times the data of its predecessor, Belle. There, these and other not yet observed processes are expected to be measurable for the first time. $B \to K^{(*)}\nu\bar{\nu}$ are very interesting due to their clean theoretical predictions and relation to the well measured $B \to K^{(*)}\ell^+\ell^-$ processes, which are in tension with Standard Model predictions. In the first half of this thesis, I give sensitivity estimates for $B \to K^{(*)}\nu\bar{\nu}$. Using the Belle II Full Event Interpretation reconstruction algorithm, I reconstruct the entire decay chain of the e^+e^- collision products. I develop the procedure for extracting the branching fraction and estimate the sensitivity as a function of amount of data collected.

To prepare for measurements of rare or difficult to measure processes, such as $B \to K^{(*)}\nu\bar{\nu}$, a simulation of all physics processes at Belle II of at least the order of the real data volume is required. Using current resources this is computationally infeasible. Until significant improvements in the simulation speed are made, selective simulation of collision events relevant to these measurements is required. In the second half of this thesis, I develop a method of predicting how likely the simulated collision event is to be useful to the targeted analysis. The prediction is made, using a neural network, before the computationally expensive stages. I show that this method of prediction produces marked savings in simulation times. The result is a fully developed training and inference procedure that integrates directly into the existing Belle II software analysis framework.

Contents

Chapter 1

Introduction

This thesis presents two separate yet related studies, performed in preparation for the upcoming Belle II experiment. The results of the first study act in part as a motivation for the second. In both I attempt to provide the relevant background for their respective studies, with some natural overlap between each.

In the first study, I establish the procedure for performing a hadronically tagged analysis of the flavour-changing neutral-current processes $B \to K^{(*)}\nu\bar{\nu}$ at Belle II using Monte Carlo simulation data. This study, along with the corresponding semileptonic tagged study, has been performed several times before at previous experiments. None have managed to record any significant evidence of the processes so far. It is expected that at the upcoming Belle II experiment a measurement will become possible. I introduce the theoretical motivations and experimental tools necessary to perform the measurement, then detail the procedure for extracting and measuring the signal processes with a preliminary projection of sensitivities. As a result, I develop the signal extraction procedure within the context of the Belle II Analysis Software Framework (basf2) [1], ready for fast reproduction with the advent of new, larger simulation data volumes. I conclude, however, that the Monte Carlo simulations currently available are not sufficient for the full systematic analysis necessary for the analysis of real Belle II data or projections thereof. Nonetheless, this motivates the development of more intelligent Monte Carlo simulation methods, which is a technical restriction on the capacity of physics analysis preparation at Belle II.

The second study in this thesis is a technical work, in which I aim at leveraging the blossoming field of machine learning to provide one possible solution to the technical restriction outlined above. The use of machine learning, and specifically neural networks, is still a relatively new development within the high energy physics community, especially in relation to the Monte Carlo simulation process. Therefore, in performing this study I have taken inspiration from outside the physics community and introduce concepts and applications from other fields that have already found success in machine learning; namely image classification and language processing. The goal of this study is to improve the simulation efficiency in a manner that is applicable to all studies of rare physics processes at Belle II. I develop a solution that natively integrates into the basf2 simulation process in a manner adaptable to any physics study being performed.

Throughout this document charge conjugation is implied unless explicitly stated otherwise. Variables and common abbreviations, are summarised with definitions in the glossary on page 146 for convenience.

Part I

$B \to K^{(*)}\nu\bar{\nu}$ hadronic tag sensitivity study

Chapter 2

Introduction

In this half of this thesis I investigate the set of $B \to K^{(*)}\nu\bar{\nu}$ processes at Belle II in the context of the new Belle II Analysis Software Framework (basf2). My goal is to prepare a sensitivity estimate for the measurement of these processes as data from Belle II begins being recorded in 2019. Specifically, I set up the analysis procedure of the four processes: $B^+ \to K^+\nu\bar{\nu}$, $B^0 \to K^0_S\nu\bar{\nu}$, $B^+ \to K^{*+}\nu\bar{\nu}$, and $B^0 \to K^{*0}\nu\bar{\nu}$.

The $B \to K^{(*)}\nu\bar{\nu}$ processes are a subset of the class known as flavour-changing neutral currents (FCNCs), which involve the transitions of quarks to a similarly charged quark of another flavour, e.g. $b \to d$, $c \to u$, etc. The Belle II experiment is especially well equipped to investigate the FCNC transition $b \to q_d\nu\bar{\nu}$, $(q_d = s, d)$, due to the high production rate of B mesons and clean detector environment afforded by the absence of partons in the initial collision. Figure 2.1 shows the status of searches for $B \to h\nu\bar{\nu}$ processes after the most recent $B \to K^{(*)}\nu\bar{\nu}$ search [2], where h represent a meson containing either an s or a d quark. Standard model predictions of the $B \to K^{(*)}\nu\bar{\nu}$ channels are shown in grey. Currently all of the $B \to h\nu\bar{\nu}$ searches have only been able to determine upper limits on branching fractions, with statistics being the limiting factor. With the upcoming Belle II experiment, however, we can expect to make a measurement of the branching fractions in the next few years, assuming they are not significantly suppressed from the Standard Model predictions due to new physics.

The $B \to K^{(*)}\nu\bar{\nu}$ processes are of particular interest due to their close relation to the $B \to K^{(*)}\ell^+\ell^-$ processes. Both processes share Wilson coefficients in the effective theory, and recent measurements at LHCb of lepton universality via the ratio $R(K^{(*)}) \equiv \frac{\mathcal{B}(B\to K^{(*)}\mu^+\mu^-)}{\mathcal{B}(B\to K^{(*)}e^+e^-)}$ have shown indications of deviation from Standard Model predictions [3, 4]. Additionally, $B \to K^{(*)}\nu\bar{\nu}$ share the same missing energy signature as processes involving exotic final states due to the invisible final state neutrinos. For example, $B \to K^{(*)}X$ where X is an exotic final state. This makes them of great interest to new physics candidate searches.

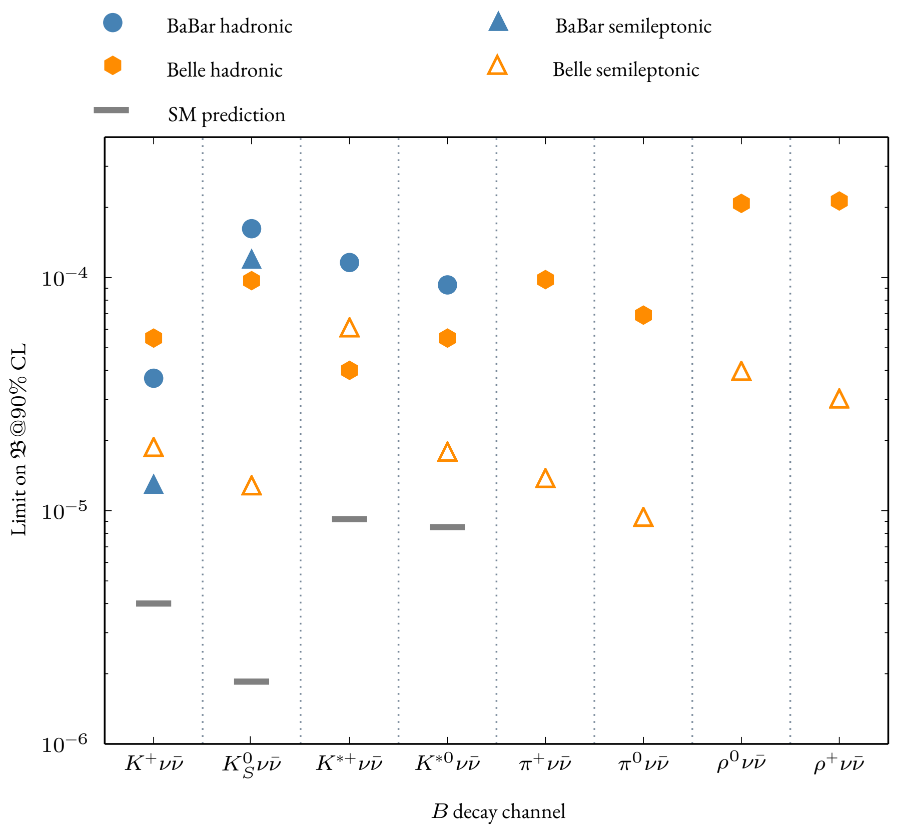
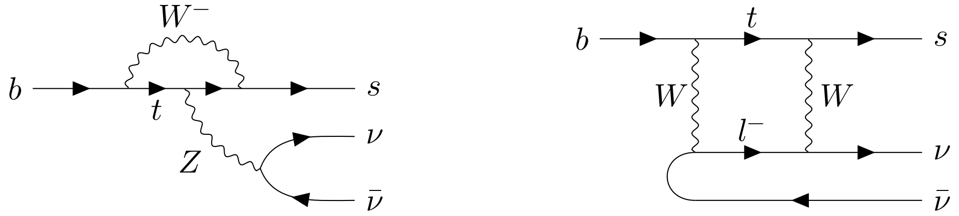


Figure 2.1: Observed limits for all $B \to h\nu\bar{\nu}$ channels in previous studies (adapted from [2]). Theoretical predictions are from [5].

Chapter 3

Theoretical framework

The $B \to K^{(*)}\nu\bar{\nu}$ processes belong to a class known as flavour-changing neutral currents (FCNCs). The quark transition involved, $b \to s$, changes flavour while preserving its electric charge (hence neutral). This transition is forbidden at tree-level within the Standard Model because neutral bosons do not take part in flavour-changing quark transitions. Instead the transition can only occur in higher order processes, involving multiple quark flavour changes. The penguin and box diagrams in figure 3.1 show the leading-order contributions to the $B \to K^{(*)}\nu\bar{\nu}$ amplitudes in the Standard Model. Only the top quark is shown inside the loops as it dominates the contributions, though as I show later in this chapter the other up-type quarks also contribute. The $B \to K^{(*)}\nu\bar{\nu}$ processes are not yet observed but very interesting due to their relation to the well-studied $B \to K^{(*)}\ell^+\ell^-$ processes which are in tension with the Standard Model. In addition, clean theoretical predictions make them excellent probes of Standard Model parameters.



(a) Penguin diagram.

(b) Box diagram.

Figure 3.1: Feynman diagrams for dominant contributions to the $b \to s\nu\bar{\nu}$ transition in the Standard Model.

3.1 Constructing the Standard Model

A key goal in high-energy physics is to find the Lagrangian density, $\mathcal{L}$. In the Lagrangian formalism this is comprised of the kinetic energy density, T, and potential, U, of the system,

$$\mathcal{L} = T - U. \tag{3.1}$$

Measuring the parameters of the Lagrangian provides a complete description of nature. To construct a Lagrangian, a model must first be hypothesised. The following components of a model must be specified before a complete Lagrangian can be constructed:

1. the gauge group of the model,
2. the representations of fields within the model,
3. the patterns of spontaneous symmetry breaking.

Once these have been specified, the Lagrangian can be constructed in its most general, renormalisable form. *General* means that all possible terms that satisfy the model components above are included, i.e. no terms can be left out based only on phenomenological/empirical arguments[1]. *Renormalisable* means that all divergences can be absorbed by a finite number of shifts (renormalisations) in the fields and couplings. In addition, Poincarè invariance is required (invariance under boost, rotation, and translation). The Lagrangian parameters must be measured as the model alone makes no specification of their values. The number of physical parameters arises from the model itself, and is finite due to renormalisability. Any exact global symmetries can either be imposed or arise as a result of the model; they are not necessarily an input requirement.

The components of the Standard Model are:

1. Gauge groups $\mathrm{SU}(3)_c \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_\mathrm{Y}$, where $\mathrm{SU}(3)_c$, $\mathrm{SU}(2)_L$, and $\mathrm{U}(1)_\mathrm{Y}$ are the colour, isospin, and hypercharge symmetry groups of the model.
2. Field (matter) representations, using the gauge notation $(colour, isospin)_{\text{hypercharge}}$,

   Left-handed quark doublets $Q_i\ (3,2)_{1/6}$

   Right-handed up quarks $U_i^c\ (\bar{3},1)_{-2/3}$

   Right-handed down quarks $D_i^c\ (\bar{3},1)_{1/3}$

   Left-handed lepton doublets $L_i\ (1,2)_{-1/2}$

[1] For example proton decay may be allowed within the theory but not something we want to include. In practice in the Standard Model this does occur and terms are neglected which have no theoretical reason not to exist, one example being CP-violation in the strong sector (strong CP problem).

Right-handed charged leptons $E_i\,(1,1)_1$

Scalar Higgs field $H\,(1,2)_{1/2}$

The index i indicates generation, in the Standard Model $i = 1, 2, 3$.

3. Spontaneous symmetry breaking, for example $\mathrm{SU}(2)_L \times \mathrm{U}(1)_\mathrm{Y} \to \mathrm{U}(1)_\mathrm{EM}$.

From these components the most general, renormalisable Lagrangian can be constructed as

$$\mathcal{L}_\mathrm{SM} = \mathcal{L}_\mathrm{kin} + \mathcal{L}_\mathrm{Higgs} + \mathcal{L}_\mathrm{Y}\,, \tag{3.2}$$

where the kinetic Lagrangian $\mathcal{L}_\mathrm{kin}$ contains the gauge interactions through the covariant derivative and non-Abelian field strengths, $\mathcal{L}_\mathrm{Higgs}$ is the Higgs potential, and $\mathcal{L}_\mathrm{Y}$ is the Yukawa Lagrangian describing the coupling of fermions to the Higgs field.

The particles of the Standard Model are shown in figure 3.2. Fermions (quarks and leptons) are matter particles; bosons, force carriers. The Standard Model contains nineteen free parameters in total: three from $\mathcal{L}_\mathrm{kin}$ defining the strength of the $\mathrm{U}(1)$, $\mathrm{SU}(2)$, and $\mathrm{SU}(3)$ interactions; two from $\mathcal{L}_\mathrm{Higgs}$ associated with the vacuum expectation value and the Higgs quartic interaction strength; thirteen from $\mathcal{L}_\mathrm{Y}$ defining lepton and quark masses, and the strength and asymmetry of flavour changing. In the Standard Model neutrinos are massless. From experiment, however, we know that the neutrinos oscillate flavour, i.e. change generation during propagation ($\nu_e \to \nu_\mu$), which requires they have a mass. An additional seven parameters describe this, three neutrino masses, three mixing angles, and one CP-violating phase[2].

3.2 Flavour changing in the Standard Model

Intergenerational quark flavour changing in the Standard Model is described by the Yukawa Lagrangian via the coupling of fermions to the Higgs field. The Yukawa Lagrangian, $\mathcal{L}_\mathrm{Y}$, contains flavour-changing interactions and can be written in terms of quark and lepton interactions as

$$\mathcal{L}_\mathrm{Y} = -Y_u^{ij}\bar{q}'_{iL}\tilde{\phi}u'_{jR} - Y_d^{ij}\bar{q}'_{iL}\phi d'_{jR} - Y_\ell^{ij}\bar{\ell}'_{iL}\phi e'_{jR} + \mathrm{h.c}\,, \tag{3.3}$$

where $Y_{u,d,\ell}^{ij}$ are the Yukawa couplings (three 3×3 matrices) between fermion families i and j, q'_L are the quark weak eigenstates with u'_R and d'_R denoting the up and down-types (L and R indicating handedness), ϕ is the Higgs doublet, and ℓ'_L and e'_R are the left handed lepton doublet and right handed charged lepton singlet.

[2] The exact number of phases depends on the nature of the neutrinos, for example Majorana neutrinos would introduce two physical phases.

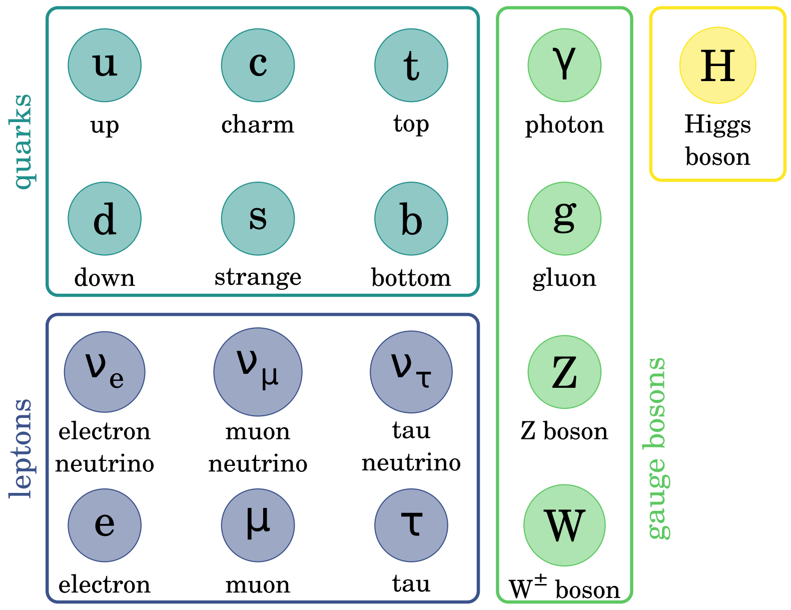


Figure 3.2: From [6], the elementary particles of the Standard Model.

There are two relevant bases involved: the mass basis, where the mass terms are diagonal and correspond to physical particles, and the weak or interaction basis, in which the weak eigenstates are diagonal. The fact that these two bases are not the same results in flavour changing within the Standard Model. The rotation between theses bases, and hence the strength of flavour changing, is described explicitly by the Cabibbo-Kobayashi-Maskawa (CKM) matrix.

The Higgs vacuum expectation value, its ground state in vacuum, is non-zero and implies spontaneous electroweak symmetry breaking,

$$\mathrm{SU}(3)_c \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_\mathrm{Y} \xrightarrow{\langle\phi\rangle\neq 0} \mathrm{SU}(3)_c \times \mathrm{U}(1)_\mathrm{EM} \,. \tag{3.4}$$

$\mathrm{U}(1)_\mathrm{EM}$ is the resulting unified electromagnetic and weak force (electroweak) after symmetry breaking. Note that the strong force, $\mathrm{SU}(3)_c$ is conserved. The resulting Yukawa Lagrangian for quarks is then

$$\mathcal{L}_\mathrm{Y} = -Y_u^{ij}\frac{v}{\sqrt{2}}\bar{u}'_{iL}u'_{jR} - Y_d^{ij}\frac{v}{\sqrt{2}}\bar{d'}_{iL}d'_{jR} + \mathrm{h.c}\,, \tag{3.5}$$

where the $\mathrm{SU}(2)_L$ quark components from equation (3.3) have been decomposed into their components

$$q'_{iL} = \begin{pmatrix} u'_{iL} \\ d'_{iL} \end{pmatrix} . \tag{3.6}$$

From this the mass terms can be identified as the matrices

$$m_q^{ij} = Y_q^{ij}\frac{v}{\sqrt{2}}\,, \quad q = u, d\,, \tag{3.7}$$

such that,

$$\mathcal{L}_{\mathrm{Y}} = -m_q^{ij}\bar{u}'_{iL}u'_{jR} - m_q^{ij}\bar{d}'_{iL}d'_{jR} + \mathrm{h.c}\,, \tag{3.8}$$

The mass matrices in the mass basis (recall we are currently working in the weak basis) are by definition diagonalisable. It is always possible to find unitary matrices V_{qL} and V_{qR} such that the mass matrix m_q is diagonalised by

$$V_{qL}m_qV_{qR}^{\dagger} = m_q^{\mathrm{diag}}\,, \tag{3.9}$$

where m_q^{diag} is diagonal and real. To achieve diagonalisation, the weak eigenstate fields, q'_{iL} and q'_{iR}, are rotated to the mass basis containing their mass eigenstates,

$$q'_{iL} = (V_{qL})_{ij}q_{jL}\,, \quad q'_{iR} = (V_{qR})_{ij}q_{jR}\,. \tag{3.10}$$

The fields q_{jL} and q_{jR} then correspond to the physical quarks.

Performing this rotation, the V_{qL} and V_{qR} matrices drop away (i.e. are trivially $\mathbb{1}$ due to unitarity) everywhere except in the quark couplings to the charged W boson,

$$\mathcal{L}^q_{W^{\pm}} = -\frac{g}{\sqrt{2}}\bar{u}_{Li}\gamma^{\mu}(V_{uL}V_{dL}^{\dagger})_{ij}d_{Lj}W_{\mu}^{+} + \mathrm{h.c}\,. \tag{3.11}$$

The resulting unitary matrix

$$V_{\mathrm{CKM}} = V_{uL}V_{dL}^{\dagger} \neq \mathbb{1}\,, \tag{3.12}$$

formed by the V_{qL} which satisfy the mass matrix diagonalisation rotation, is known as the Cabibbo-Kobayashi-Maskawa (CKM) matrix for quark mixing [7, 8]. Because V_{CKM} is not diagonal, the W bosons are able to couple to quark mass eigenstates of different generations. This is the only source of quark flavour changing within the Standard Model, with V_{CKM} describing the relative rates of flavour change.

The unitary V_{CKM} can then be written with the quark generations ordered by mass as follows

$$V_{\mathrm{CKM}} = \begin{bmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{bmatrix}\,, \tag{3.13}$$

which can be parametrised with four free parameters; three real and one imaginary. It is useful to write V_{CKM} in an explicit parametrisation to show these, one common choice from [9] is

$$V_{\mathrm{CKM}} = \begin{bmatrix} c_{12}c_{13} & s_{12}c_{13} & s_{13}e^{-i\delta_{13}} \\ -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{i\delta_{13}} & c_{12}c_{23} - s_{12}s_{23}s_{13}e^{i\delta_{13}} & s_{23}c_{13} \\ s_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta_{13}} & -c_{12}s_{23} - s_{12}c_{23}s_{13}e^{i\delta_{13}} & c_{23}c_{13} \end{bmatrix}\,, \tag{3.14}$$

where $c_{ij} = \cos\theta_{ij}$, $s_{ij} = \sin\theta_{ij}$, and the three θ_{ij} are the real mixing parameters. δ_{13} is the imaginary Kobayashi-Maskawa (charge-parity violating) phase. These four parameters are precisely those from the SM described in the previous section.

An alternative is the Wolfenstein parametrisation [10], which defines its parameters as

$$\sin\theta_{12} = \lambda, \quad \sin\theta_{13} = A\lambda^2, \quad \sin\theta_{13}e^{i\delta} = A\lambda^3(\rho + i\eta)\,. \tag{3.15}$$

The benefit of this choice of parametrisation is that A, ρ, and η are all $\mathcal{O}(1)$, and λ can be written as a Taylor expansion up to the precision required, for example

$$V_{\text{CKM}} = \begin{bmatrix} 1-\lambda^2/2 & \lambda & A\lambda^3(\rho - i\eta) \\ -\lambda & 1-\lambda^2/2 & A\lambda^2 \\ A\lambda^3(1-\rho-i\eta) & -A\lambda^2 & 1 \end{bmatrix} + \mathcal{O}(\lambda^4)\,. \tag{3.16}$$

Current experimental values for V_{CKM} are[9]

$$V_{\text{CKM}} = \begin{bmatrix} 0.974\,46 \pm 0.000\,10 & 0.224\,52 \pm 0.000\,44 & 0.003\,65 \pm 0.000\,12 \\ 0.224\,38 \pm 0.000\,44 & 0.97359^{+0.00010}_{-0.00011} & 0.042\,14 \pm 0.000\,76 \\ 0.00896^{+0.00024}_{-0.00023} & 0.041\,33 \pm 0.000\,74 & 0.999\,105 \pm 0.000\,032 \end{bmatrix}\,. \tag{3.17}$$

3.3 Flavour-changing neutral currents

Up to this point I have shown that in mass basis, only the W boson is able to change quark flavour. Therefore flavour-changing neutral currents must proceed at the one-loop level or higher; tree-level contributions are forbidden. For loop-level processes the Glashow-Iliopoulos-Maiani (GIM) mechanism [11] describes the strength of interaction. The underlying principle of the GIM mechanism is that for a unitary matrix any pair of rows or columns are orthogonal. Inspecting the loop shown in for example the penguin diagram in figure 3.1, the CKM dependence alone is

$$\mathcal{M} \propto \sum_i V_{ib}^* V_{is} = 0\,, \tag{3.18}$$

which vanishes by unitarity. The only other sources of flavour dependence in the SM are the masses of the quarks themselves. I showed before that the flavour-changing effects occur purely due to off-diagonal couplings, therefore couplings do not change flavour if they are diagonal. It is possible, however, to change basis to produce off-diagonal couplings. Therefore this must be generalised to involve strictly universal matrices, i.e. those proportional to the identity ($\mathbb{1}$). As the mass matrices, m_i, are not universal it is then possible to write down

$$\mathcal{M} = \sum_i V_{ib}^* V_{is} f(m_i)\,, \tag{3.19}$$

where any terms independent of m_i must vanish (i.e. can be rotated away). $f(m_i)$ is the Inami-Lim function [12] which depends directly on $\frac{m_i^2}{m_W^2}$, i.e. $f(\frac{m_i^2}{m_W^2})$. Therefore heavier intermediate quark masses are expected to dominate the loop contributions, hence the transition is expected to be most sensitive to the top quark. With this mass dependence in mind, and noting the unitarity relation in equation (3.18), the amplitude can be rewritten to show the destructive interference between different flavour quark contributions explicitly,

$$\mathcal{M} = V_{tb}^* V_{ts} \left[f\left(\frac{m_t^2}{m_W^2}\right) - f\left(\frac{m_c^2}{m_W^2}\right) \right] + V_{ub}^* V_{us} \left[f\left(\frac{m_u^2}{m_W^2}\right) - f\left(\frac{m_c^2}{m_W^2}\right) \right] . \quad (3.20)$$

Therefore the closeness of quark masses of different generations directly suppresses the amplitude, specifically in the second term as $m_c - m_u \ll M_W$. This is what is referred to as GIM suppression. This also highlights an important property of high precision experiments like Belle II, that despite operating at energies lower than the mass of the top quark, the measurement of its off-shell effects on higher order processes allows indirect probing of its properties. Likewise, this is possible for any other heavy particles involved, e.g. Higgs or new physics particles.

3.4 Theoretical predictions

Having two neutrinos in the final state means that the $b \to s\nu\bar{\nu}$ subset of FCNCs are the cleanest theoretically. They don't suffer from the long-range effects experienced by the $b \to s\ell^+\ell^-$ transitions. In $b \to s\ell^+\ell^-$, the exchange of photons in the final state breaks factorisation, introducing hadronic uncertainties beyond the form factors. In $b \to s\nu\bar{\nu}$, however, the factorisation is exact, meaning it is theoretically possible to perform measurements of the form factors via branching fraction measurements alone.

From [5], the effective Hamiltonian of $b \to s\nu\bar{\nu}$ transitions in the Standard Model is

$$\mathcal{H}_{eff}^{SM} = -\frac{4G_F}{\sqrt{2}} V_{tb} V_{ts}^* C_L^{SM} \mathcal{O}_L + \text{h.c}\,, \quad (3.21)$$

where G_F is the Fermi coupling constant, $V_{tb}V_{ts}^*$ is the product of the dominant CKM elements. The effective operator, $\mathcal{O}_L$, is given by

$$\mathcal{O}_L = \frac{e^2}{16\pi^2} (\bar{s}\gamma_\mu P_L b)(\bar{\nu}\gamma^\mu(1-\gamma_5)\nu)\,. \quad (3.22)$$

with an implicit summation across all three neutrino flavours as contributions, and where P_L is the left handed chiral projection operator $\frac{1}{2}(1-\gamma^5)$. The left-handed Wilson coefficient,

C_L^{SM}, is the coupling constant of $\mathcal{O}_L$ and is known to a high precision from NLO QCD corrections [13–15] and electroweak corrections [16] to be

$$C_L^{SM} = -\frac{X_t}{s_w^2}, \quad X_t = 1.469 \pm 0.017 \pm 0.002\,, \tag{3.23}$$

where s_w is the weak-mixing angle, $\sin^2\theta_w$. As no right-handed neutrinos are allowed within the Standard Model, the complementary right-handed Wilson coefficient C_R^{SM} is trivially zero.

From this, there are three observables that can be measured from the $B \to K^{(*)}\nu\bar{\nu}$ processes [5, 17, 18],

$$\mathcal{B}(B^+ \to K^+\nu\bar{\nu})_{SM} = \tau_{B^+}3|N|^2(C_L^{SM})^2\langle\rho_K\rangle\,,, \tag{3.24}$$

$$\mathcal{B}(B^0 \to K^{*0}\nu\bar{\nu})_{SM} = \tau_{B^0}3|N|^2(C_L^{SM})^2\langle\rho_{A_1} + \rho_{A_{12}} + \rho_V\rangle\,, \tag{3.25}$$

$$F_L^{SM}(B \to K^*\nu\bar{\nu}) = \left\langle \frac{\rho_{A_{12}}}{\rho_{A_1} + \rho_{A_{12}} + \rho_V} \right\rangle\,. \tag{3.26}$$

These are the branching fractions of the unexcited and excited processes, as well as the K^* longitudinal polarisation fraction. The latter is not considered in this study but should be measured after measurements of the branching fractions have been obtained to provide additional SM constraints. Here I have given the expected SM branching fractions in terms of the expectation values of the rescaled form factors, ρ_i, integrated over the dilepton invariant mass squared, q^2. The complete form factors are defined in [5] with current expectation values given in [17]. $\tau_{B^{+/0}}$ are the B meson lifetimes. Since there is no isospin asymmetry involved, the B^0 and B^+ branching fractions are simply obtained by rescaling by the corresponding lifetime. The factor 3 in the branching fractions is to account for contributions from all three neutrino flavours, and the factor N contains the contributions from the CKM elements,

$$N = V_{tb}V_{ts}\frac{G_F\alpha}{16\pi^2}\sqrt{\frac{m_B}{3\pi}}\,, \tag{3.27}$$

where G_F and α are the normal Fermi coupling and fine-structure constants, and m_B the mass of the B meson.

The resulting Standard Model predictions are shown in table 3.1. As a final note for future iterations of this study, the recently published Flavio software package [19] provides a simple interface for obtaining up-to-date Standard Model (and new physics) predictions. The software release timing meant that it was too late to be used in this study but I strongly recommend its usage in future.

3.5 Previous studies

There have been several studies performed that have tried to measure $B \to K^{(*)}\nu\bar{\nu}$ processes in the past, though so far only upper limits have been observed for all four channels. Table 3.1

shows the current best experimental limit for each channel at 90% credibility level. In all cases the upper limit is close to the Standard Model prediction, and expected to be measurable with the increase in data expected at Belle II. [20] forecasts a measurement of branching fractions with $10-11\%$ uncertainty when the entire 50 ab^{-1} Belle II dataset is available.

The three studies which set the current best limits were:

Belle 2017 [2] This is the most recent study and used the full reconstruction [21] semileptonic tagging algorithm at Belle to measure $B \to h\nu\bar{\nu}$ processes, where $h = K^+$, K_S^0, K^{*+}, K^{*0}, π^+, π^0, ρ^+, and ρ^0. First a semileptonically decaying B meson was hierarchically reconstructed with the help of neural networks [22]. The reconstructed tag B was then combined with a single signal-like h candidate to form the fully reconstructed initial $\Upsilon(4S)$ candidate. The remaining energy in the electromagnetic calorimeter, E_{ECL}, was then fitted to directly measure the branching fractions. The upper limits set by this study on all but the K^+ and K^{*+} channels are the most stringent to date. Both this and the Belle 2013 study used the full Belle 771 fb^{-1} dataset containing approximately 772 $\times$ 10^6 $B\bar{B}$ pairs.

Belle 2013 [23, 24] This study used the same full reconstruction tagging algorithm as that of the Belle 2017, instead used to reconstruct a hadronically decaying tag-side B meson (i.e. no neutrinos in the final state). The reconstruction procedure was similar to that used in the Belle 2017 study with the candidate selection requirements changed accordingly and the same fitted observable E_{ECL} used. This study additionally measured an upper limit for the $h = \phi$ channel.

Babar 2013 [25] The 2013 study from the BaBar collaboration holds the current world best upper limit for the $B^+ \to K^+\nu\bar{\nu}$ channel. The study itself used a hadronic tagging method, like that of the Belle 2013, but the final results are combined with the semileptonic tag Babar 2010 study [26]. This study differs from the Belle analyses in that the background models were fixed from data (sidebands or via relaxing selections) and signal yields obtained from a cut-and-count (instead of a fit to signal region). A data sample of 471 $\times$ 10^6 $B\bar{B}$ pairs (429 fb^{-1}) was used.

All of the previous studies were dominated by statistical uncertainty, motivating the repeat of both the hadronic and semileptonic searches at Belle II. Projections estimate observations of the charged and K^{*0} channels at 10 ab^{-1}, and measurements with around 10% uncertainty at the full 50 ab^{-1}. The systematic uncertainties of the previous Belle studies were dominated by the background model. This was due to a lack of statistically significant background Monte Carlo events from which to obtain an accurate parametric background shape surviving the signal selection procedure, despite using MC samples five and ten times the nominal luminosity for the hadronic and semileptonic tag studies respectively. Therefore, before an improved analysis can be performed at Belle II, the volume of background Monte Carlo simulations

Table 3.1: Standard Model predictions and current best upper limits from previous studies on all signal channels [17].

Mode	$\mathcal{B}_{SM}[10^{-6}]$	$\mathcal{B}_{exp}[10^{-6}]$ (90% C.L.)
$B^+ \to K^+\nu\bar{\nu}$	4.68 ± 0.64	< 16 (BaBar 2013)
$B^0 \to K_S^0\nu\bar{\nu}$	2.17 ± 0.30	< 13 (Belle 2017)
$B^+ \to K^{*+}\nu\bar{\nu}$	10.22 ± 1.19	< 40 (Belle 2013)
$B^0 \to K^{*0}\nu\bar{\nu}$	9.48 ± 1.10	< 18 (Belle 2017)

will need to be increased significantly (this relates precisely to the second half of this thesis). The Babar study's major systematic uncertainty source was from peaking background yield and signal efficiency estimates from Monte Carlo. The signal efficiency uncertainties were due to kaon and pion selection windows used, while the peaking background uncertainty was attributed to uncertainties in the branching fractions of the contributing processes.

It is worth noting here as well that the previous studies assumed an equal probability of $\Upsilon(4S)$ decay into neutral and charged B meson pairs. This study does not, instead the cross sections are based on the results of [27] and the world average [9] which found[3] a slightly lower branching fraction of $\mathcal{B}(\Upsilon(4S) \to B^0\bar{B}^0) = 0.486$.

3.6 New physics potential

I will end the discussion of theoretical considerations with a description of how new physics (NP) could be expected to enter the $B \to K^{(*)}\nu\bar{\nu}$ processes. Working with the Standard Model effective Hamiltonian from equation (3.21), new physics can be introduced in general in the form of an additional operator (e.g. $C_R\mathcal{O}_R$ for right handed neutrinos), a modification to the existing operator (e.g. $C_L^{NP}\mathcal{O}_L^{NP}$), or a combination of both.

Looking at the first case, the assumption is made that lepton flavour universality holds and that new physics enters at scales above the mass of the B meson. Equation (3.21) can be extended to

$$\mathcal{H}_{eff}^{SM} = -\frac{4G_F}{\sqrt{2}} V_{tb}V_{ts}^*(C_L\mathcal{O}_L + C_R\mathcal{O}_R) + \text{h.c}\,, \tag{3.28}$$

where the additional $\mathcal{O}_{\mathcal{R}}$ is simply

$$\mathcal{O}_R = \frac{e^2}{16\pi^2}(\bar{s}\gamma_\mu P_R b)(\bar{\nu}\gamma^\mu(1-\gamma_5)\nu)\,. \tag{3.29}$$

[3]Assuming $\mathcal{B}(\Upsilon(4S) \to B\bar{B}) = 1$.

This change can be described by the two real quantities,

$$\epsilon = \frac{\sqrt{|C_L|^2 + |C_R|^2}}{|C_L^{SM}|}\,, \quad \eta = \frac{-\operatorname{Re}(C_L C_R^*)}{|C_L|^2 + |C_R|^2}\,, \tag{3.30}$$

with $\epsilon > 0$ and $\eta \in [\frac{-1}{2}, \frac{1}{2}]$. In the SM $\epsilon = 1$ and $\eta = 0$, with $\eta \neq 0$ indicating the presence of right handed currents in FCNCs of quarks. Relating this back to observable quantities, the following three ratios can be measured experimentally,

$$\mathcal{R}_K = (1 - 2\eta)\epsilon^2 \quad = \frac{\mathcal{B}(B \to K\nu\bar{\nu})}{\mathcal{B}(B \to K\nu\bar{\nu})_{SM}}\,, \tag{3.31}$$

$$\mathcal{R}_{K^*} = (1 + \kappa_\eta \eta)\epsilon^2 = \frac{\mathcal{B}(B \to K^*\nu\bar{\nu})}{\mathcal{B}(B \to K^*\nu\bar{\nu})_{SM}}\,, \tag{3.32}$$

$$\mathcal{R}_{F_L} = \frac{1 + 2\eta}{1 + \kappa_\eta \eta} \quad = \frac{F_L}{F_L^{SM}}\,, \tag{3.33}$$

where κ_η is dependent on the form factors from eqs. (3.24) to (3.26).

The $B \to K^{(*)}\nu\bar{\nu}$ processes receive contributions from some of the same $\mathrm{SU}(2)_L$ invariant four-fermion operators as $B \to K^{(*)}\ell^+\ell^-$. $B \to K\ell^+\ell^-$, however, contains additional operators, for example dipole operators arising from the presence of right handed currents. Recently, measurements of the ratio $R(K^{(*)}) = \frac{B \to K^{(*)}\mu^+\mu^-}{B \to K^{(*)}e^+e^-}$ has shown signs of tension with Standard Model predictions [3, 4]. Therefore, measuring $B \to K^{(*)}\nu\bar{\nu}$ processes will allow probing of the shared Wilson coefficients and isolation of the tension. It is also expected that any lepton flavour universality violation (non-universality) will be more pronounced in processes involving τ leptons. Therefore, any amplification in $B \to K^{(*)}\nu\bar{\nu}$ will be extra sensitive to the contributions from $\nu_\tau\bar{\nu}_\tau$ final states.

In the case of lepton flavour non-universality, where lepton flavour is still conserved within physics processes but the lepton generations are treated differently, the left and right-handed operators from eq. (3.29) simply obtain the index $\ell = e, \mu, \tau$ to differentiate between neutrino flavours,

$$\mathcal{O}_L^\ell = \frac{e^2}{16\pi^2}(\bar{s}\gamma_\mu P_L b)(\bar{\nu}_\ell\gamma^\mu(1 - \gamma_5)\nu_\ell)\,, \tag{3.34}$$

$$\mathcal{O}_R^\ell = \frac{e^2}{16\pi^2}(\bar{s}\gamma_\mu P_R b)(\bar{\nu}_\ell\gamma^\mu(1 - \gamma_5)\nu_\ell)\,. \tag{3.35}$$

Accordingly, ϵ and η from (3.30) become

$$\epsilon_\ell = \frac{\sqrt{|C_L^\ell|^2 + |C_R^\ell|^2}}{|C_L^{SM}|}\,, \quad \eta_\ell = \frac{-\operatorname{Re}(C_L^\ell C_R^{\ell *})}{|C_L^\ell|^2 + |C_R^\ell|^2}\,, \tag{3.36}$$

and hence

$$\mathcal{R}_K = \frac{1}{3}\sum_\ell (1-2\eta_\ell)\epsilon_\ell^2 \;= \frac{\mathcal{B}(B\to K\nu\bar{\nu})}{\mathcal{B}(B\to K\nu\bar{\nu})_{SM}}\,, \tag{3.37}$$

$$\mathcal{R}_{K^*} = \frac{1}{3}\sum_\ell (1+\kappa_\eta\eta_\ell)\epsilon_\ell^2 = \frac{\mathcal{B}(B\to K^*\nu\bar{\nu})}{\mathcal{B}(B\to K^*\nu\bar{\nu})_{SM}}\,, \tag{3.38}$$

$$\mathcal{R}_{F_L} = \frac{\sum_\ell \epsilon_\ell^2(1+2\eta_\ell)}{\sum_\ell \epsilon_\ell^2(1+\kappa_\eta\eta_\ell)} \;= \frac{F_L}{F_L^{SM}}\,. \tag{3.39}$$

Again the same measured quantities can be used to probe the new physics.

To extend this to lepton flavour violating processes, indices to distinguish neutrino flavour can be introduced,

$$\mathcal{O}_L^{ij} = \frac{e^2}{16\pi^2}(\bar{s}\gamma_\mu P_L b)(\bar{\nu}_i\gamma^\mu(1-\gamma_5)\nu_j)\,, \tag{3.40}$$

$$\mathcal{O}_R^{ij} = \frac{e^2}{16\pi^2}(\bar{s}\gamma_\mu P_R b)(\bar{\nu}_i\gamma^\mu(1-\gamma_5)\nu_j)\,, \tag{3.41}$$

where $i \neq j$, and the measured observables are modified accordingly (all neutrinos are still assumed to be left handed). A detailed description of further new physics model extensions can be found in [5].

There is even the opportunity to perform more exotic searches, such as dark matter searches in $B \to K^{(*)}X_{DM}$ [28], or $B \to K^{(*)}X_{dark}$ with X_{dark} being some dark mesons (e.g. a dark pion) [29]. Searches for intermediate vector boson resonances have also been proposed [30] in the form of $B \to K^{(*)}V(\to \chi\chi)$, where χ is a Dirac fermion that may be, for example, a sterile (right handed) neutrino. Such searches require careful consideration of kinematic restraints on the reconstructed kaon.

Overall, the NP modifications described above are intended to demonstrate the wealth of potential that $B \to K^{(*)}\nu\bar{\nu}$ hold for probing new physics. Even if agreement with SM predictions is found, the resulting new physics constraints alone will help narrow down the viable NP models and bring us a step closer to isolating precisely where the Standard Model breaks down.

Chapter 4

The Belle II experiment

The Belle II collaboration was formed in 2009 with the aim of taking everything learned from the predecessor Belle experiment and using it to build a high statistics B-physics factory sensitive enough to probe for new physics (NP). There was already a decade-long history of success at e^+e^- colliders from not only Belle, but also its companion experiment BaBar. Most notably for Belle was the confirmation of the Kobayashi-Maskawa mechanism [8] with the charge-parity (CP) asymmetries in the transitions of b-quarks which led to the Nobel prize being awarded to Kobayashi and Maskawa in 2008. Other great achievements in flavour physics, including measurements of unitarity triangle angles, time-dependent CP- violation (CPV), new resonances [31], etc., are summarised nicely in [32]. With the 50 ab^{-1} of data expected to be collected at Belle II, the goal is to give insight into some of the big questions that plague the SM. For example looking for new sources of CP-violation, as the Standard Model does not provide enough to explain the matter-antimatter asymmetry we see in the universe today. In addition, new physics in semileptonic and leptonic processes, along with other new physics areas such as lepton flavour violation, dark sector, etc., searches will be performed.

There are several unique experimental advantages to using a B-factory, as opposed to measuring B meson decays from a hadron collider (as is done at the LHCb experiment). Most prominently, direct branching fraction measurements can be performed as the number of initial B mesons produced is known within a small uncertainty. At hadron colliders only ratios of branching fractions can be measured to high precision. The environment from e^+e^- collisions is exceptionally clean: only the two B mesons are produced and their initial energy is very well known. This allows the full reconstruction of everything inside the detector and therefore measurements of processes involving missing final states (e.g. neutrinos). The precision of Belle II allows the analysis of processes that proceed only at higher order (forbidden at tree-level). This makes Belle II sensitive to particles with masses above those that can be produced directly from the collider's energy. This enables a wide range of new physics searches at higher energy scales.

The lifetime of the Belle II experiment is divided into three major phases:

Phase 1 This ran from February until June of 2016 and involved the beam commissioning with the commissioning detector BEAST II (described in [33]) used to take beam background measurements.

Phase 2 Operating from February to July of 2017, phase 2 involved the tuning of collisions at SuperKEKB. The Belle II detector was in place, without the VXD, taking calibration measurements.

Phase 3 The final phase of Belle II, set to begin in the first half of 2019, is the full physics run. For this the complete Belle II detector will be in place and recording physics data[1].

4.1 SuperKEKB

The experiment is based at the Japanese High-Energy Accelerator Research Organisation (KEK) in Tsukuba, Japan. It will use SuperKEKB (figure 4.2), the upgraded KEKB electron-positron collider. SuperKEKB is an asymmetric e^+e^- collider with a 7 Gev electron high energy ring (HER) and a 4 Gev positron low energy ring (LER) inside a 3 km circumference tunnel which has been reused from KEKB. The centre of mass energy is at the $\Upsilon(4S)$ resonance, which is almost exactly double the B meson rest mass. Figure 4.1 shows the mass energy spectrum of the $\Upsilon(nS)$ resonances, with the red line indicating the threshold for $\Upsilon(4S)$ production and the grey shaded line at 10.58 GeV, the centre of mass energy at SuperKEKB. This energy results in a very high production rate of B meson pairs which are essentially at rest in the centre of mass frame. The key changes made from KEKB are the so called nano-beam scheme being used which will squeeze the electron bunches and increase the instantaneous luminosity, and a change in the beam energies from 8 and 3.5 Gev to reduce emittance in the new scheme. The nano-beam scheme involves minimising the longitudinal size of the interaction point (IP) overlap region to effectively limit the minimum value of the beta function via the hourglass effect. The Lorentz boost factor of the centre of mass system will be $\gamma = 0.28$, two thirds of that in Belle. The trade-off of these changes made to increase the luminosity at Belle II is a significant increase expected in beam-related backgrounds [33].

4.1.1 Beam backgrounds

This section briefly describes the various beam-related background sources at SuperKEKB. A detailed description of each source can be found in [32, 33, 35].

[1] As of February 2019, only two of the twelve outer ladders of the PXD are installed. The remainder will be installed at a later time during a servicing period of the detector.

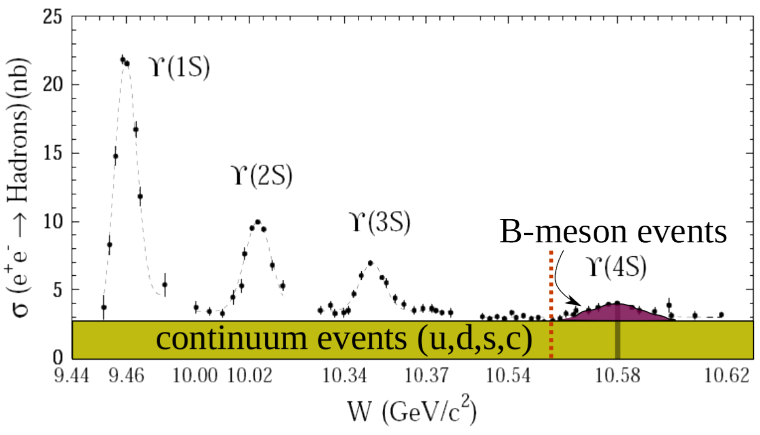


Figure 4.1: Spectrum of energy ranges covered by the $\Upsilon(nS)$ resonances and the underlying continuum background.

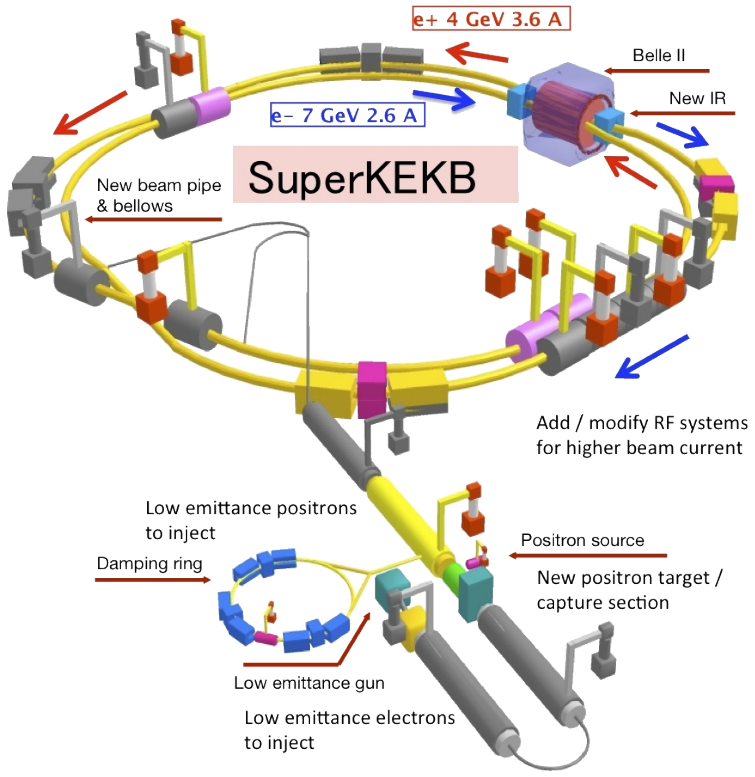


Figure 4.2: Schematic of the SuperKEKB accelerator [34]. The Belle II detector lies at the crossing point of the electron and positron beams.

Touschek scattering This involves intra-bunch Coulomb scattering of two particles. The interaction causes the involved particles' energies to deviate from the energy of the bunch: one higher and one lower. The aforementioned nano-beam scheme and changes to the beam currents at SuperKEKB are expected to results in a Touschek scattering background level ~ 20 times higher than that of KEKB [33].

Beam gas scattering The vacuum in the beam pipe at SuperKEKB contains a small amount of residual gas molecules unable to be removed. Scattering of beam particles off these molecules can occur in two ways: Coulomb scattering, which will change the direction of beam particles, and Bremsstrahlung scattering, which decelerates the beam particles. Bremsstrahlung scattering is expected to be negligible compared to the Touschek scattering backgrounds. The Coulomb scattering rate, however, is expected to be ~ 100 times higher than that of KEKB due to the reduced beam pipe radius.

Synchrotron radiation As synchrotron radiation scales proportional to the square of both the beam energy and magnetic field strength, therefore the high energy ring is expected to produce the majority of this type of background. During the early stages of the Belle experiment synchrotron radiation damaged the inner vertex detector, therefore for Belle II special measures have been taken to prevent any substantial levels of this background reaching the detector.

Radiative Bhabha process The radiative Bhabha process, $e^-e^+ \to e^-e^+\gamma$, results in photons travelling approximately along the beam line which interact with the iron in the accelerator magnets. This produces a large number of low energy photons as well as neutrons entering the detector. The resulting electron and positron also have lower energies after the Bhabha process, causing over-bending by the SuperKEKB focusing magnets (those directly before the Belle II detector). These then hit the walls of the magnets, sending electromagnetic showers into the detector. The rate of both of these background sources is proportional to the luminosity which will be 40 times that of KEKB.

Two-photon process This background comes from the very low momentum products of the interaction $ee \to eeee$, which can spiral within the magnetic field of the detector leaving many hits within the inner detector components.

Injection background Each time a fresh beam bunch is injected into the accelerator a surge in the beam-related backgrounds is seen. A trigger is applied to veto the time window of the surge directly after each injection to mitigate this.

The key takeaway from all of these backgrounds is that at SuperKEKB the overall background level is expected to be significantly higher than that at KEKB. Recalling that the background shape uncertainty was a major contributor to the systematic uncertainty of the previous experiments, it's clear that a competitive measurement at Belle II will require a strong understanding of the beam-related backgrounds in order for them to be adequately suppressed.

4.2 The Belle II detector

The Belle II detector is the centrepiece of the experiment. It is a 7×7.5 m full-solid-angle detector with many sub-detector layers surrounding the interaction point. The detector is built as mostly a refurbished Belle detector, with the goal of maintaining the performance of the Belle detector in the presence of the considerably higher background levels. A labelled 3D cross-section of the Belle II detector is shown in figure 4.3. The detector is comprised of the following sub-detectors: Vertex detector (VXD), Central drift chamber (CDC), Particle identification (PID), Electromagnetic calorimeter (ECL), K-Long and muon detector (KLM).

The key changes from the Belle detector are [32]:

- The beam pipe radius at the interaction point has been reduced from 15 to 10 mm, allowing the vertex detector to be closer to the interaction point.
- The old silicon strip detector immediately outside the beam pipe has been replaced with a two-layer pixel detector.
- The remaining silicon strip detector has been extended to have a larger radius than in Belle.
- The CDC has a larger volume and smaller cell sizes than in Belle.
- Particle identification is performed by entirely new devices using Čerenkov imaging, with faster read-outs than in Belle.
- The end-cap scintillator crystals (CsI(Tl)) in the ECL have been replaced with faster, more radiation tolerant pure CsI crystals, and new electronics will be used.
- The end-cap and inner layers of the KLM have been replaced with scintillators.

4.3 Vertex detector

The vertex detection module is comprised of two sub-detectors: a pixel detector (PXD) and a silicon vertex detector (SVD). The PXD contains two layers of the DEPleted p-channel Field Effect Transistor (DEPFET). The SVD is made of four layers of Double Sided Strip Detectors (DSSD). The primary purpose of the vertex detection system is to measure the vertices of the

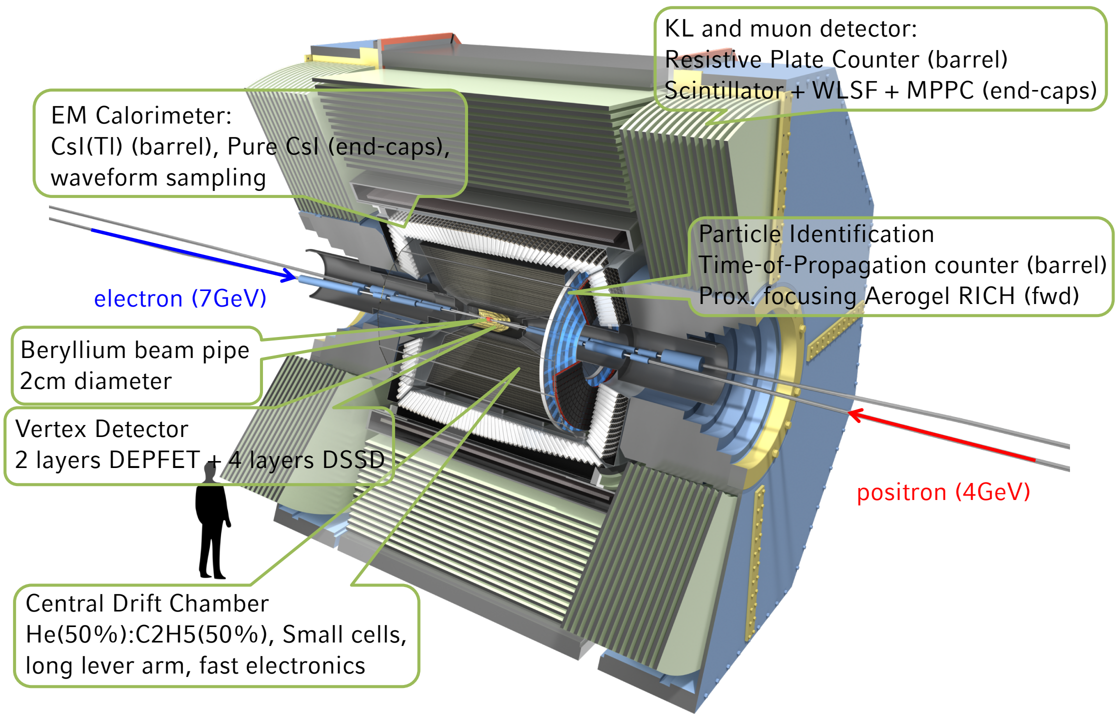

KL and muon detector:
Resistive Plate Counter (barrel)
Scintillator + WLSF + MPPC (end-caps)
EM Calorimeter:
CsI(Tl) (barrel), Pure CsI (end-caps),
waveform sampling
electron (7GeV)
Particle Identification
Time-of-Propagation counter (barrel)
Prox. focusing Aerogel RICH (fwd)
Beryllium beam pipe
2cm diameter
Vertex Detector
2 layers DEPFET + 4 layers DSSD
positron (4GeV)
Central Drift Chamber
He(50%):C2H5(50%), Small cells,
long lever arm, fast electronics

Figure 4.3: Cross-section animation of the Belle II detector [34].

two B meson decays for mixing-induced CPV measurements and the vertices of D meson and τ lepton decays. Given the lower centre of mass boost in SuperKEKB, the two B meson decay vertices will have a smaller separation than in Belle. Nevertheless, the smaller beam pipe width at the interaction region and the larger radius of the SVD are expected to allow Belle II to have as good or better vertexing performance than Belle.

4.3.1 Pixel detector

At the high luminosity of SuperKEKB, the detector components closest to the beam pipe will experience incredibly high hit rates coming from the beam-related backgrounds and low-momentum-transfer QED processes (e.g. photon-photon interactions). The beam pipe radius at the IP is only 10 mm, and the background experienced increases in proportion to the inverse square of the radius. Therefore strip detectors are no longer used for the innermost layer (as was the case in Belle) due to the increased occupancy. Instead, pixel detectors are used for the two innermost layers of the vertex detector. The two layers of the PXD are at radii 14 mm and 22 mm from the beam line. Pixel detectors have already been successfully used in detectors at the LHC [36, 37]. The lower energy of SuperKEKB, however, means that thinner sensors need to be used. DEPFET technology has been used and to allow for sensors as thin as 50 microns, which only require air cooling and can be engineered to be radiation hard enough to

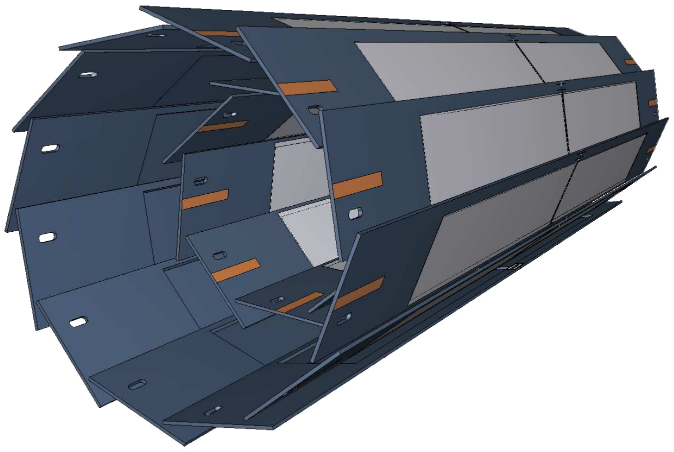


Figure 4.4: Ladder structure of pixel detector modules [32]. The light grey rectangles are the arrays of pixels themselves with the remainder containing readout electronics.

withstand that expected in Belle II. A schematic of the DEPFET sensor layout in the PXD can be seen in figure 4.4, note the ladder structure to ensure full coverage of the acceptance region. The inner layer of the pixel detector contains 8 modules (ladders) with 3.072 M pixels. The outer layer will have 12 modules with 4.608 M pixels (only two are installed currently). The expected spatial resolution is $\sim$ 10 m [32].

4.3.2 Silicon vertex detector

The silicon vertex detector (SVD) comprises the outer four layers of the vertex detection subdetector at radii 38, 80, 115, 140 mm [38]. Comparing this with Belle, which had its outermost SVD layer at a radius of only 88 mm, the reconstruction efficiency of low-momentum particles and long-lived particles is expected to improve greatly. The SVD in Belle II covers the full detector acceptance range of $17° < \theta < 150°$. The hit occupancy is required to be less than 10% to ensure hits in the SVD are correctly associated with tracks in the CDC, and the expected maximum average trigger rate is 30 kHz. The double-sided silicon microstrip detectors used in the SVD provide excellent timing resolution ($\sim$ 23 ns), which complement the excellent spatial resolution of the PXD.

4.4 Central drift chamber

The central drift chamber (CDC) is the main tracker for charged particles in Belle II. It is larger than in Belle with a smaller cell size, and contains 14, 336 sense wires and 42, 240 field wires. A superconducting solenoid coil surrounds the ECL to supply a 1.5 T magnetic field for the CDC. The CDC has three key roles in Belle II: to reconstruct charged tracks with precision momentum measurements, to provide particle identification information using measurements of energy loss (e.g. for identifying low momentum tracks which do not reach the PID detector), and to provide efficient and reliable trigger signals for charged particles. Given the success of the CDC design in Belle the same design structure is used in Belle II. A comparison of the major parameters in Belle and Belle II can be seen in table 4.1.

The key changes to the CDC design are:

- New readout electronics are used to handle the expected higher trigger rates with less deadtime.
- The CDC inner radius and outer radii have been increased to avoid the high backgrounds near the IP and to make room for the new, larger VXD.
- The CDC will generate 3D trigger information using a z-direction trigger.

Table 4.1: Comparison of main CDC parameters in Belle and Belle II.

	Belle	Belle II
Radius of inner cylinder (mm)	77	160
Radius of outer cylinder (mm)	880	1130
Radius of innermost sense wire (mm)	88	168
Radius of outermost sense wire (mm)	863	1111.4
Number of layers	50	56
Number of sense wires	$8,400$	$14,336$
Gas	$He - C_2H_6$	$He - C_2H_6$
Diameter of sense wire (μm)	30	30

4.5 Particle identification

The particle identification (PID) sub-detector is completely new in Belle II and contains two components: a time Of propagation (TOP) detector, and an aerogel ring imaging Čerenkov (ARICH) detector. The TOP detector is used for particle identification in the barrel region of Belle II. The ARICH detector performs particle identification in the forward end-cap region.

4.5.1 Time of propagation detector

The time-of-flight and Čerenkov detector in the barrel region of Belle has been replaced by a time of propagation (TOP) counter in Belle II. The goal of the TOP is to improve

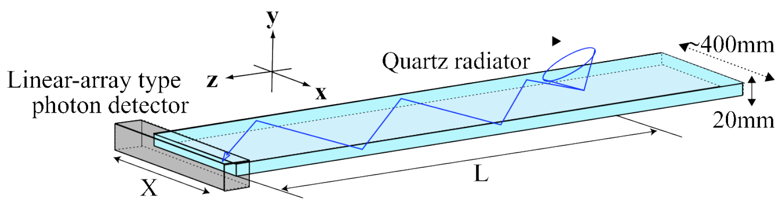


Figure 4.5: Example of a single TOP module collecting a Čerenkov photon. Position information is collected for the $x - y$ directions and timing information is collected in the z direction to reconstruct the Čerenkov ring (blue cone) [32].

K/π separation while coping with the increased backgrounds expected from SuperKEKB. Overall the TOP contains sixteen modules with each module comprised of: two 2.7 m long quartz bars, a spherical mirror on one end, and an expansion prism on the other with an array of photo-detectors. It uses micro-channel-plate photomultiplier tubes (MCP-PMTs) and waveform sampling electronics for high precision position and timing measurements ($O(100\,\text{ps})$ resolution). The Čerenkov ring, as shown in figure 4.5, is reconstructed in three-dimensions from the detected time and the $x - y$ position of the Čerenkov photon hits on the MCP-PMTs.

4.5.2 Aerogel ring imaging Čerenkov detector

The Aerogel ring imaging Čerenkov detector (ARICH) detector is used for particle identification in the forward end-cap. Each detector module contains two layers of aerogel together ($20 + 20$ mm thick) separated by an expansion volume (200 mm) from an array of 420 Hybrid Avalanche Photo Detectors (HAPD). The two layers of aerogel have differing refractive indices to provide overlapping of the Čerenkov rings for a better photon yield. The focusing of the ARICH has been constructed to separate kaon Čerenkov photons from pion Čerenkov photons across most of their momentum range, while also discriminating between pions, muons, and electrons in the momentum range below 1 GeV/c. This is exceptionally useful for a B-factory, as most processes have an abundance of pions and kaons either in the primary decay or as secondary particles to reconstruct. An example of how kaons and pions can be discriminated between is shown in figure 4.6.

4.6 Electromagnetic calorimeter

Following the success of the electromagnetic calorimeter (ECL) in Belle, the Belle II detector has reused the ECL design with upgrades to handle the higher backgrounds expected. The material from Belle has been reused in both the barrel and end-cap, containing Thallium doped CsI(Tl) scintillating crystals. New electronics will also be used with bias filtering and waveform

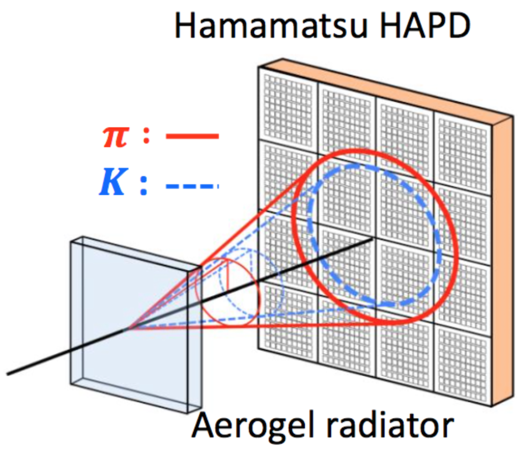


Figure 4.6: ARICH setup demonstrating how kaons and pions with the same momentum can be discriminated between [39].

sampling for faster readouts. The coverage of the ECL is $12.4° < \theta < 155.1°$, excluding two 1° gaps where the barrel and end-caps join. The key roles of the ECL are to: detect photons with precision measurements, identify electrons, take on-line and off-line luminosity measurements, and help detect K_L^0 together with the KLM. The energy resolution of the ECL in Belle was $\frac{\sigma_E}{E} = 4\%$ at 100 MeV and 1.6% at 8 GeV, with an angular resolution of 13(3)mrad at low (high) energies [35]. It is expected achieve similar performance in Belle II, with the upgrades compensating for the increased backgrounds.

4.7 K_L and μ detector

The K-long and muon detector (KLM) in Belle II is made of alternating layers of 470 mm thick iron plates and detector components. The resistive plate chambers (RPCs) that were used throughout the entire Belle KLM to detect charged particles will not be efficient enough to handle the ambient hit rate expected in the Belle II end-caps and barrel inner-layers. Instead scintillators are being used in the entire end-cap and first two layers of the barrel section, with RPCs used for the remaining barrel layers. In the barrel there are fifteen detector components and fourteen iron plates. In the forward (backward) end-cap there are fourteen (twelve) detector layers and iron plates. The iron plates also serves as a magnetic flux return for the solenoid and provide interaction material in which K_L^0 mesons can shower hadronically. The total coverage of the KLM (barrel + end-caps) is $20° < \theta < 155°$. The barrel KLM was the first sub-detector to be installed in Belle II in 2013. The end-caps were installed in 2014.

Chapter 5

Experimental tools

This study is performed within the context of the Belle II Analysis Software Framework (basf2) [1, 40], with two key multivariate analysis (MVA) tools utilised throughout. In this chapter I describe the two tools: fast boosted decision trees (section 5.1) and full event interpretation (section 5.2). For a description of the operation and fundamental principles of the software framework see section 10.2 of part II.

5.1 Fast boosted decision trees

Fast boosted decision trees (fBDTs) are an extension of traditional decision tree classifiers. Decision trees (DTs) are simple classifiers that allow the dividing of a given phase space into subsets relevant to the categories being classified. Figure 5.1 shows an example of a three level decision tree, where the task is to identify signal (red) in the presence of some background (blue) given some input variables x, y, and z. The values of the thresholds applied at each node are adjusted during training, in which a sample of signal and background events are fed into the tree with known labels (signal or background). The adjustments attempt to maximise the separation gain between signal and background for each given sample. However, simple decision trees are prone to overfitting of the training data and struggle to extrapolate well when applied to new data. To combat this, in particular within the Belle II implementation, [41] introduced a boosted component to the training procedure. A boosted decision tree (BDT) constructs a series of shallow decision trees during the training stage. The restriction on the depth of each shallow decision tree prevents overfitting by ensuring each individual DT is only able to weakly classify the inputs. By combining many weak classifiers a robust, strongly classifying decision tree can be constructed.

The *fast* in fast boosted decision tree simply refers to the CPU cache friendly implementation of fBDT which performs the training of the decision trees on each individual level

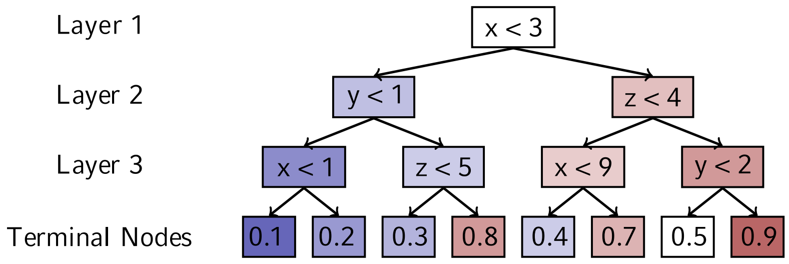


Figure 5.1: A three layer decision tree using thresholds to classify sections of the input phase space. Each level performs a binary decision until a final terminal node is reached [41]. The numbers shown in the terminal nodes is the fraction of input data points that reach that node which are signal, e.g. 0.9 means 90% of all inputs to reach that node are signal events.

sequentially, making use of the memory access patterns of modern CPUs. Additional pre-processing and instruction-level parallelism is performed to provide further speedups and are described in detail in [41].

5.2 Full event interpretation

A key part of missing energy studies at Belle II, those containing neutrinos in the final state, is the ability to fully reconstruct the $\Upsilon(4S)$ decay within the detector while also having knowledge of the initial state of the $\Upsilon(4S)$. This is what makes the Belle II experiment invaluable in high energy physics research and distinguishes it from B physics studies performed at hadron colliders (e.g. LHCb).

The full event interpretation (FEI) [42] software is the tool used within the Belle II software to perform this full reconstruction of the $\Upsilon(4S)$. The full FEI procedure is performed in six sequential stages, shown in figure 5.2. Beginning with the reconstructed tracks and clusters, FEI builds final state particle candidates. These are then combined in the following stages to construct intermediate particles until finally a B meson candidate is produced. FEI employs a series of fast Boosted Decision Tree (fBDT) (section 5.1) to identify the likelihood of each reconstructed candidate based on its properties and the previous stage's output. To this effect FEI is able to hierarchically reconstruct a candidate B meson via a bottom-up approach.

Each stage in the FEI training is configurable along with the hyperparameters of the fBDTs used. Two modes of training exist: generic and specific. Generic is the training of FEI on simulated generic B meson decays (*Charged* and *Mixed* channels shown later in table 6.1). This has the advantage that the training is agnostic to the signal process being searched for in a particular analysis and can therefore be used by anyone needing FEI. For this purpose two FEI generic trainings are performed centrally by the collaboration with each Monte Carlo campaign released (see section 10.3). Specific FEI training is the training of FEI on a particular signal channel, with the signal side B meson being already reconstructed. The benefit here is a training that is able to learn specifically the kinematics and subtle features of that particular signal mode, making for a higher tag side reconstruction efficiency and purity. However, this requires a large ($(O)(100M)$ events) signal Monte Carlo sample and must be retrained for every signal channel required. Given that each stage of the FEI training depends on the output of the previous stage, training requires $\sim$ 1 week to complete, excluding validation, on a specifically commissioned computing resource.

In addition to the generic and specific training modes, FEI may be further separated into hadronic and semileptonic reconstructions. Hadronic reconstruction considers only the B meson decay chains containing no leptons, excluding the case of decays involving a $J\psi$ which is detected via its $\ell^+\ell^-(\ell = \mu, e)$ daughters. Since all final state particles of the B decay are detectable and reconstructed its four-momentum is measured allowing for a very pure sample to be extracted. Hadronic B decays, however, have only a branching fraction of $\mathcal{O}(10^{-3})$ and so this FEI sample suffers from a low reconstruction efficiency. Semileptonic reconstruction deals with only the $B \to D\ell\nu$ and $B \to D^*\ell\nu$ decay channels. These decays have a higher branching fraction and the lepton daughter of the B has in general a high momentum allowing for a higher overall reconstruction efficiency than the hadronic FEI. The trade-off for the higher efficiency is the loss of purity, with the presence of an undetectable neutrino in the final states meaning the four-momentum of the B is not able to be fully reconstructed and therefore signal decays better selected for.

Overall the FEI is able to reconstruct over 100 individual sub-decay chains, resulting in a combined $\mathcal{O}(10^4)$ possible complete B meson decays. The output of the FEI is a list of reconstructed B meson candidates for each input event. Each candidate has an associated signal probability (sigProb) which indicates the predicted likelihood that the associated B meson was correctly reconstructed.

For this study the generic training of hadronically tagged B^0 and B^+, trained centrally by the Belle II collaboration on 10×10^8 events each. At the time of writing this is the latest available training, the use of updated and specific trainings is discussed in section 8.1. The correct reconstruction efficiencies of these trainings are 0.42% for B^+ decays, and 0.24% for B^0. The difference between neutral and charged reconstruction efficiency is due to the difference in coverage of decay modes between the B^0 and B^+ in FEI. The total branching fraction coverage of the neutral B^0, when taking into account decay modes of the daughters of the B mesons, is only about 60% that of the reconstructed charged B^+.

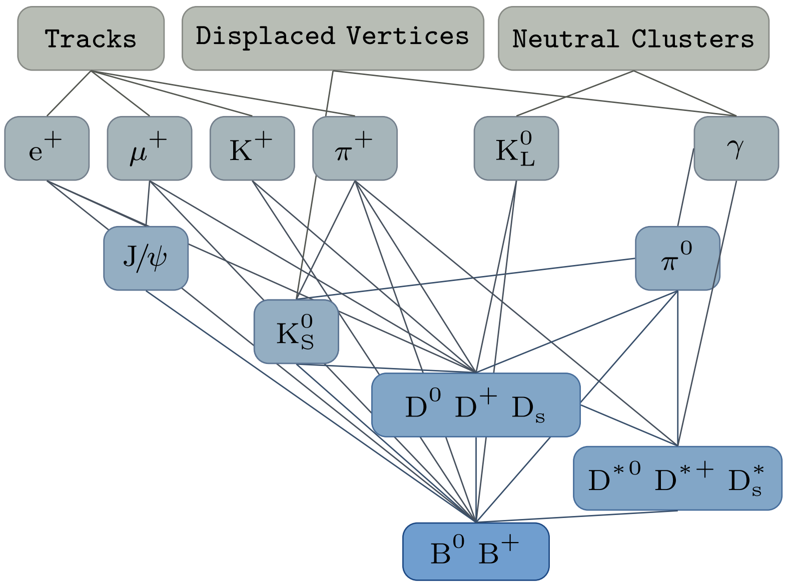


Figure 5.2: Hierarchical reconstruction approach used by FEI [42]. Reconstruction runs from top to bottom sequentially to create B meson candidates from a range of combinations of intermediate particles.

Chapter 6

Experimental Procedure

In order for a branching fraction to be measured, the signal $B \to K^{(*)}\nu\bar{\nu}$ processes must be identified amongst all e^+e^- collision events. In this study I use Monte Carlo simulations to tune the analysis for identifying and isolating the signal and rejecting the non-signal (background) events. To achieve this a large simulation sample is required. The two invisible neutrinos in the final state necessitates using information about of the second B meson in the $e^+e^- \to \Upsilon(4S) \to B\bar{B}$ process.

6.1 Data Sample

The bulk of Monte Carlo (MC) samples used in this study are generated centrally by the Belle II collaboration as part of the so-called ninth MC production campaign (MC9). This includes all of the background categories described in table 6.1, as well as the two K^* signal modes: $B \to K^{*+}\nu\bar{\nu}$ and $B \to K^{*0}\nu\bar{\nu}$. The K^{*+} decays generically in simulation, with roughly 66.6% going to $K^0\pi^+$ and 33.2% to $K^+\pi^0$[1]. The K^{*0}, however, is restricted to decay exclusively to $K^+\pi^-$, which makes up only 66.57% of the total expected K^{*0} standard model decays. I account for this factor in the branching fraction upper limit calculations in section 7.1.2. The two unexcited K signal modes, $B \to K^+\nu\bar{\nu}$ and $B \to K^0_S\nu\bar{\nu}$, I simulated using local computing resources. All of the background channels used in this study correspond to 0.8 ab^{-1} integrated luminosity equivalent, with the exception being the mixed backgrounds, which at the time of this study contained only 0.672 ab^{-1} worth of events available. This missing quantity I account for in the efficiency estimate performed later in this chapter. It is convenient to group the mixed and charged backgrounds together under the name BB Background, and the uubar, ddbar, ccbar, and ssbar together as continuum. This is due to these two groups' components having similar topological decay features. All events in this thesis are simulated using the Belle II Analysis Software Framework (basf2) version *release-00-09-01*.

See section 10.2 for a detailed explanation of the software.

It is instructive at this point to make a quick calculation of the approximate number of signal events that can be expected to be produced at Belle II if I assume Standard Model branching fractions. This will provide some context for the final reconstruction efficiencies at the end of this chapter in section 6.2.5). Using the values from table 3.1 along with the number of $B\bar{B}$ pairs expected per ab^{-1} from table 6.1, I obtain the per ab^{-1} mean values shown in the latter table. The result is a $\mathcal{O}(10^6)$ difference between signal and background events present, and indicates the level of the difference between these categories' efficiencies to aim for.

A crucial difference between the study I perform in this thesis and the two previous Belle studies (section 3.5) is the discrepancy in Monte Carlo simulation volumes used. The Belle 2013 hadronic tag study used five times the nominal data luminosity to prepare the analysis, while the Belle 2017 semileptonic tag study used ten (six for continuum). The background shape uncertainty, determined from simulation, was a major contribution to the systematic uncertainty in these studies. The statistical uncertainty, however, dominated the signal yield measurements. The increased data volume at Belle II is expected to reduce these statistical uncertainties such that the systematic uncertainties will become relevant. Therefore, I expect a significant increase in the Monte Carlo background simulation to be needed in order for this study at Belle II to be performed at a comparable or improved level to the previous. Indeed in the conclusion of this half of the thesis I find exactly that, that the currently available Monte Carlo simulation volume is simply insufficient.

Finally, tree-level $B^+ \to \tau^+\nu_\tau(\tau^+ \to K^+\bar{\nu}_\tau)$ processes [43, 44] have the same final state signature as $B^+ \to K^+\nu\bar{\nu}$. These are not considered explicitly in this study as they are not expected to contribute significantly to current backgrounds; the overall branching fraction is of the order 10^{-7}. Instead they are included in the background simulation and I model them as such. A discussion of when they might be relevant and how they may be considered in future is included in section 8.1. There I also explained why they are of greater significance to other $B \to h\nu\bar{\nu}$ channel searches. x

6.2 Signal Event Selection

I perform the process of selecting signal events in several logically separate stages: first I reconstruct generically decaying B mesons, labelled B_{tag}, using the Full Event Interpretation (FEI) software [42], then I reconstruct a signal-side B meson, labelled B_{sig}, pair it with the generic candidates to produce an $\Upsilon(4S)$. Finally I apply a series of selections to reduce the background events while retaining as many signal events as possible. As the Full Event Interpretation software requires by a significant margin the largest computing time and has the lowest single retention rate ($\mathcal{O}(1\%)$), it is performed first. The reconstructed B_{tag} candidates are

[1]The remaining 0.2% decays to $K^+\gamma$ and is ignored as negligible in the reconstruction in this study.

Table 6.1: MC production categories, expected number of event per ab^{-1} integrated luminosity, and corresponding luminosity equivalents of events simulated. An X in the simulated decay indicates any allowed Standard Model decay products (generic decay).

Channel	Simulated Decay	Expected per ab^{-1}	Simulated ab^{-1}
Signal (K^+)	$B^+ \to K^+ \nu\bar{\nu}$	5.29×10^3	9.43×10^2
Signal (K^0_S)	$B^0 \to K^0_S \nu\bar{\nu}$	2.32×10^3	4.2×10^3
Signal (K^{*+})	$B^+ \to K^{*+} \nu\bar{\nu}$	11.56×10^3	3.46×10^3
Signal (K^{*0})	$B^0 \to K^{*0} \nu\bar{\nu}$	10.14×10^3	3.94×10^3
Charged	$\Upsilon(4S) \to B^+B^- \to X$	565.4×10^6	0.8
Mixed	$\Upsilon(4S) \to B^0\bar{B}^0 \to X$	534.6×10^6	0.672
uubar	$e^+e^- \to u\bar{u}$	1605×10^6	0.8
ddbar	$e^+e^- \to d\bar{d}$	401×10^6	0.8
ccbar	$e^+e^- \to c\bar{c}$	1329×10^6	0.8
ssbar	$e^+e^- \to s\bar{s}$	383×10^6	0.8

saved as output with only loose selection criteria enforced (sec 6.2.1). This process is referred to as skimming within Belle II, and for the data samples produced as part of MC9 it is performed centrally by a skimming coordination group within the collaboration.

Throughout the remainder of this study I divide the reconstructed signal event candidates into two categories: signal and self-crossfeed (SCF). Candidates reconstructed and displayed as signal correspond to correctly reconstructed signal B mesons. For these the reconstructed particles have been cross-checked with the Monte Carlo simulation information to confirm that the reconstructed particles and their combination into parent particle candidates are correct. Self-crossfeed I use to refer to those candidates reconstructed from signal events incorrectly, either through the misidentification of a final state particle, for example the K^+ in $B^+ \to K^+\nu\bar{\nu}$ in fact being a misidentified π^+, or the incorrect combination of particles when reconstructing decaying candidates, for example the kaon in the aforementioned example actually originating from the tag-side B meson but falsely attributed to the signal side B decay.

6.2.1 Hadronic tagging

In the case of this study, where the signal process contains at most two detectable final state particles (FSPs), I inspect what remaining particles have been detected as a means of verifying that the event did indeed contain a $B \to K^{(*)}\nu\bar{\nu}$ signal process. The goal is to look for events which, after the entire $\Upsilon(4S)$ decay chain has been reconstructed, contain nothing else within the detector aside from beam-related backgrounds. To do this I utilise the Full Event Interpretation (FEI) software package [42] included in the Belle II Analysis Software Framework (basf2), described in detail in section 5.2. The FEI software attempts to perform a hierarchical reconstruction of a generically decaying B meson from the bottom up via $\mathcal{O}(1000)$ possible combined decay channels. The goal being to cover the largest possible total branching fraction of neutral and charged B decays, providing the largest possible reconstruction efficiency.

Given the vast volume of MC data and that many analyses make use of FEI, it is practical to commission a single global processing of background MC with loose selections applied to produce the subset of output files containing reconstructed B_{tag} candidates. In the aforementioned commissioned skims used in this study a pre-trained FEI network is used. The training was performed on the previous Monte Carlo production campaign, known as MC7, which was simulated with an earlier basf2 version (*release-00-07-02*). The commissioned FEI training and selections are applied identically to the locally simulated K^+ and K_S^0 signal channels as well. For both the charged (B^+) and neutral (B^0) hadronic reconstruction channels, requirements were placed on the following observables (see glossary for details)

nTracks ≤ 12

$M_{\mathbf{bc}}$ $> 5.24\,\text{GeV/c}^2$

$|\Delta(E)|$ $< 0.2\,\text{GeV}$

sigProb > 0.001

where nTracks is the total number of reconstructed charged particles in the event, M_{bc} is the beam-constrained mass of the reconstructed B_{tag} defined as $M_{bc} = \sqrt{E_{beam}^2 - p_{B_{\text{tag}}}^2}$, $\Delta(E)$ is the reconstructed energy difference $\Delta(E) = E_{B_{\text{tag}}} - E_{beam}$, and sigProb is the correct-reconstruction probability output by FEI. The selections are included in the commissioned skims. Figure 6.1 shows an example of the characteristic shapes of the M_{bc}, $\Delta(E)$, and sigProb for correctly and incorrectly reconstructed B_{tag} candidates. The skim selections applied exclude regions outside those dominated by correctly reconstructed B_{tag} candidates. Note that no best-candidate selection, i.e. selection of at most a single signal candidate per event, has been made at this stage of this analysis. The variables used in these skim selections are optimised further in section 6.2.5 where appropriate.

The resulting event-level efficiencies for the given selections used in each FEI skim channel are shown in table 6.2. These include events with incorrectly reconstructed candidates. The simulated signal channels in general have a lower multiplicity (lower total number of particles present), which translates to a lower number of possible combinations that can be used to make up B_{tag} candidates. The reconstructed B^0 efficiencies are lower than the B^+ due to the lower branching coverage of B^0 in FEI.

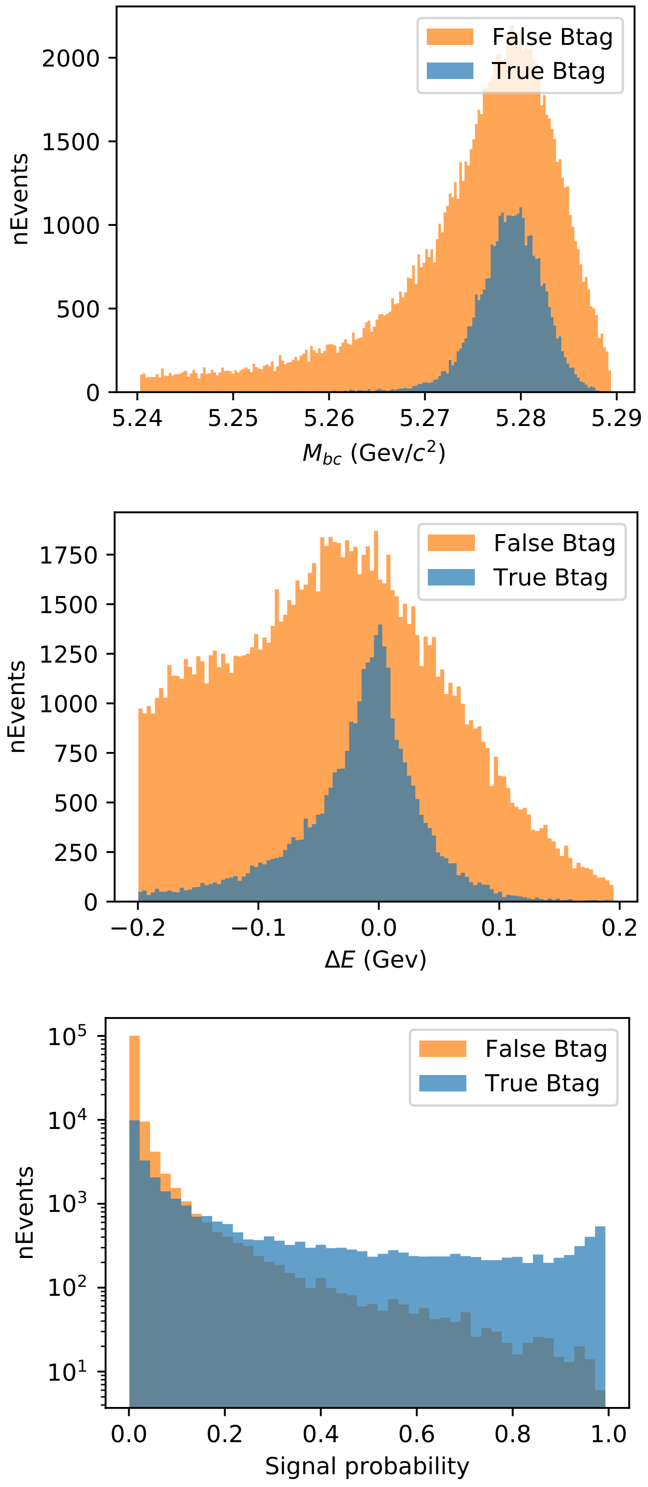


Figure 6.1: Example of characteristic kinematic distributions of correctly (True Btag) and incorrectly (False Btag) used to motivate FEI skims (not stacked). Events shown are reconstructed from hadronic FEI B^+ skims.

Table 6.2: FEI skim event retention rates of each channel reconstructed, before signal-side reconstruction (sec. 6.2.2) has been performed.

Channel	Had B^+	Had B^0
K^+	3.10%	–
K^0	–	1.30%
K^{*+}	3.26%	–
K^{*0}	–	1.27%
Mixed	5.62%	4.25%
Charged	8.35%	3.82%
$u\bar{u}$	6.86%	3.78%
$d\bar{d}$	7.20%	3.39%
$c\bar{c}$	12.0%	5.73%
$s\bar{s}$	6.13%	2.95%

6.2.2 Signal side reconstruction

The next step in signal selection is to identify the signal-side B meson candidates. In all four signal channels of this study the B meson has only a single detectable daughter. In the case of the two excited kaons and the K_S^0, the kaons themselves have two daughters that must first be reconstructed. Therefore I begin the signal side reconstruction with the four kaon reconstructions. I reconstruct the K^+ with requirements on the particle ID and χ^2 probability such that they have a 95% reconstruction efficiency. The particle ID utilises particle likelihood information, e.g. from Čerenkov rings in the ARICH detector (section 4.5.2), to distinguish between particle types and determine how likely the reconstructed particle is indeed a kaon.. The χ^2 track fit likelihood allows suppression of incorrectly reconstructed charged tracks. K_S^0 are reconstructed during the reconstruction stage of Monte Carlo simulation (Chapter 10) from charged pion pairs and are provided as-is by basf2. This process is known in Belle II as V^0*-like particle reconstruction*, details of how it is performed are given in [35]. I require reconstructed K_S^0 to be within 50 MeV of the nominal K_S^0 mass and have daughters which form a valid vertex (i.e. share the same origin). I reconstruct K^{*+} candidates from a combination of either $K^+\pi^0$ or $K_S^0\pi^+$ pairs, with the mass 80 MeV required to be within the nominal mass. K^{*0} candidates I reconstruct solely from a $K^+\pi^-$ pair, with a mass required to be within 150 MeV of the nominal K^{*0} mass. For the kaon daughters of the K^* modes, I apply the same selection criteria as those used to reconstruct the unexcited signal modes. I select π^+ daughters with particle likelihoods and χ^2 requirements such that a 95% reconstruction efficiency is achieved. I reconstruct the π^0 daughters from photon pairs, with restrictions on the reconstructed mass and individual photon daughter energies such that the π^0 have a 40% reconstruction efficiency. Finally, each reconstructed kaon I attribute to a corresponding parent B meson with matching charge.

6.2.3 $\Upsilon(4S)$ reconstruction

The final step in the signal reconstruction is to produce an $\Upsilon(4S)$ candidate for each event. To do this, I pair every signal-side B meson candidate with every tag-side B candidate to produce an $\Upsilon(4S)$ candidate, with the pairing procedure ensuring that the paired signal and tag-side B mesons contain no overlapping daughter particles. The handling of event involving multiple $\Upsilon(4S)$ candidates is described in section 6.2.5.

What remains in the detector, excepting the $\Upsilon(4S)$ candidate, I reconstruct in the form of remaining charged tracks and ECL hits and referred to as the rest of event (ROE). I require charged tracks in the ROE to have a point of closest approach within the interaction region, with $dr < 2\,\text{cm}$ and $|dz| < 4\,\text{cm}$ (see glossary on page 146 for details). These track requirements remove secondary particles originating from long-lived particle decays (which may result in double counting of particles) or detector interactions. The remaining tracks should originate from the primary B meson decays. This does, however, risk excluding V^0 particles (K_S^0, λ^0, converted γ). At the time of writing work is underway within the Belle II collaboration to optimise the ROE track filtering with the use of trained decision trees and if possible should be included in future iterations of this analysis. I place restrictions on the ECL clusters considered in the ROE, outlined below. The outcome of this is the extraction of the fit variable E_{ECL}, defined as

$$E_{\text{ECL}} = E_{\text{obs}} - E_{\text{rec}}, \tag{6.1}$$

where E_{obs} is the total energy observed in the ECL, and E_{rec} is the energy used to reconstruct the $\Upsilon(4S)$ candidate. For a signal event, since I have reconstructed every detectable particle that took part in the decay, there should be nothing left in the detector except particle originating from beam-related backgrounds. For a background event, I expect additional particles remaining on top of those from beam background or a reconstructed $\Upsilon(4S)$ signature not like that of a signal event. Therefore the reconstructed rest of event should have restrictions such that the beam backgrounds are excluded as best as possible, and only particles originating from the initial e^+e^- collision (primary particles) remain. The resulting E_{ECL} should then show signal events peaking sharply at 0 GeV and backgrounds having a non-zero-peaking structure[2]. The E_{ECL} energy in this study is specifically the neutral extra energy. I place track restrictions on the ROE during the selection optimisation stage that discard any signal candidates containing charged particles not associates with the reconstructed $\Upsilon(4S)$. An example of the characteristic shape of E_{ECL} is shown in figure 6.2. Here the expected peaking signal in the lowest energy bin can be seen. Further information about the ROE useful for signal-background discrimination is also obtained, with the relevant variables introduced in section 6.2.5.

To find the optimal ROE requirements for ECL clusters I tested five combinations of varying ECL cluster timing, ECL cluster error timing, and cluster energy, E, against a baseline

[2]Due to the high statistical uncertainty of the previous experiments' measurements there is no clear structure for background E_{ECL} distributions.

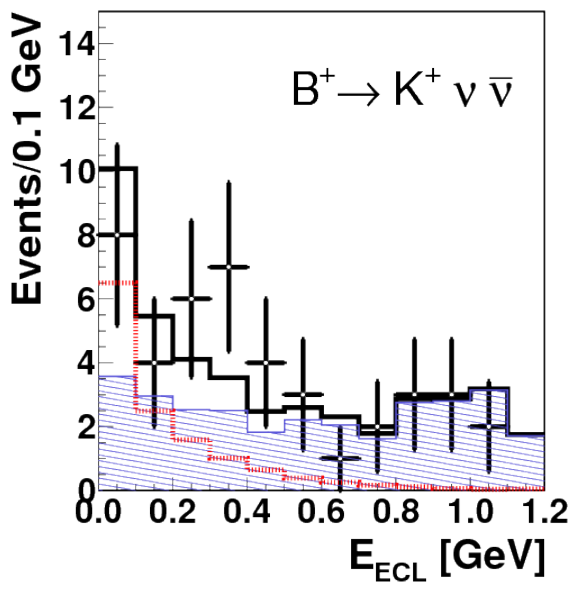


Figure 6.2: Observed E_{ECL} from previous Belle hadronically tagged study [23] for the $B^+ \rightarrow K^+\nu\bar{\nu}$ signal channel. The red line is the fitted signal and the blue shaded region the background fit. The black data points are the observed events from data and the black line is the combined fit.

photon requirement. ECL cluster timing is the difference in nano-seconds between the bunch crossing (e^+e^- collision) and ECl hit. As the photons from beam-backgrounds are not associated with the bunch crossing I expect them to show a relatively uniform distribution across the ECL cluster timing measurements. ECL cluster error timing is the ECL cluster's timing uncertainty range that contains 99% of true photons, and E is simply the total energy deposited in the ECL cluster. I fix the requirement on charged tracks to the track origin requirements outlined above. I obtain the baseline requirements for ECL hits from the physics photon reconstruction requirements (see the neutral particle identification section of [35]) which are as follows:

- The ECL cluster is hypothesised during the reconstruction stage to come from a photon, i.e. no charged tracks are associate with the cluster as is the case for electrons.
- The cluster is contained entirely within the acceptance region of the tracking detectors (deg $17 < \theta <$ deg 150). This allows detection of clusters originating from charged particles.
- The E1E9 energy concentration ratio is required to be greater than 0.4 or the total energy of the photon greater than 75 MeV. This is the ratio of energy deposited in the central (highest energy) crystal in a cluster to the square of 9 crystals centred around that central crystal. Photons originating from the primary decay are expected to have a

narrower spread of energy than those from beam-related background photon showers [35].

- The total energy of the photon is greater than 50 MeV if it is detected in the barrel or backward end-cap, and greater than 75 MeV if in the forward end-cap. Photons originating from the primary physics event are expected to have in general a higher energy spectrum than those from beam-backgrounds. The asymmetric energy of SuperKEKB means that beam-backgrounds are expected to have a larger impact on the forward region of the detector, hence the higher minimum energy threshold.

This baseline filter is labelled as *UpsilonROE*_0 for the remainder of this thesis.

To motivate the various ROE requirements explored I use a test data sample of charged generic B^+B^- pair decays. Figure 6.3 shows the relation between ECL cluster timing and ECL cluster error timing for ECL clusters reconstructed as photons. Blue points are primary photons originating from the products of the e^+e^- collision and red are the backgrounds I want to filter out. From inspection, a linear function can be used to separate the two classes of reconstructed photons. The three linear filters of |ECL cluster timing| $< a \times$ ECL cluster error timing are shown on the same figure, with $a = 1$ (yellow), 0.5 (green), and 0.1 (orange). Figure 6.4 shows the reconstructed energies, E, for the same two classes of photons. Comparing this with the ECL cluster timing in figure 6.5, it can be seen that the background clusters that can be filtered using ECL cluster timing are predominantly in the lower energy region. Therefore I also test a conditional threshold on the energy, keeping all clusters above 0.1 GeV. This is shown on the same figure with the green line.

The different filters tested, all added as additional requirements to the baseline, and their associated names are:

UpsilonROE_1	|ECL cluster timing| $<$ ECL cluster error timing
UpsilonROE_2	|ECL cluster timing| $< 0.5 \times$ ECL cluster error timing
UpsilonROE_3	|ECL cluster timing| $<$ ECL cluster error timing or E > 0.1 GeV
UpsilonROE_4	|ECL cluster timing| $< 0.5 \times$ ECL cluster error timing or E > 0.1 GeV
UpsilonROE_5	|ECL cluster timing| $< 0.1 \times$ ECL cluster error timing or E > 0.1 GeV

Using the figure of merit (FOM) $\frac{S}{\sqrt{B}}$, where S is the number of reconstructed primary photons, and B the number of photons from beam-backgrounds, I obtain the values for each filter shown in table 6.3. The ROE requirement category *UpsilonROE*_2 shows the best ratio of retained good clusters to rejected background clusters. Therefore for the remainder of this study I apply this ROE filter.

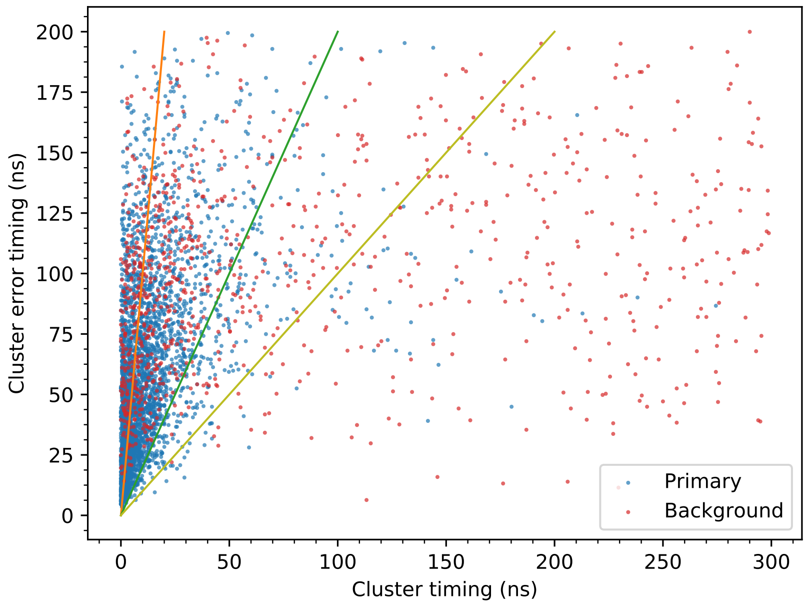


Figure 6.3: Comparison of ECL cluster timing to ECL cluster error timing for primary (blue) and background (red) photons. The lines show the $a = 1$ (yellow), 0.5 (green), and 0.1 (orange) linear filter gradients as described in the text.

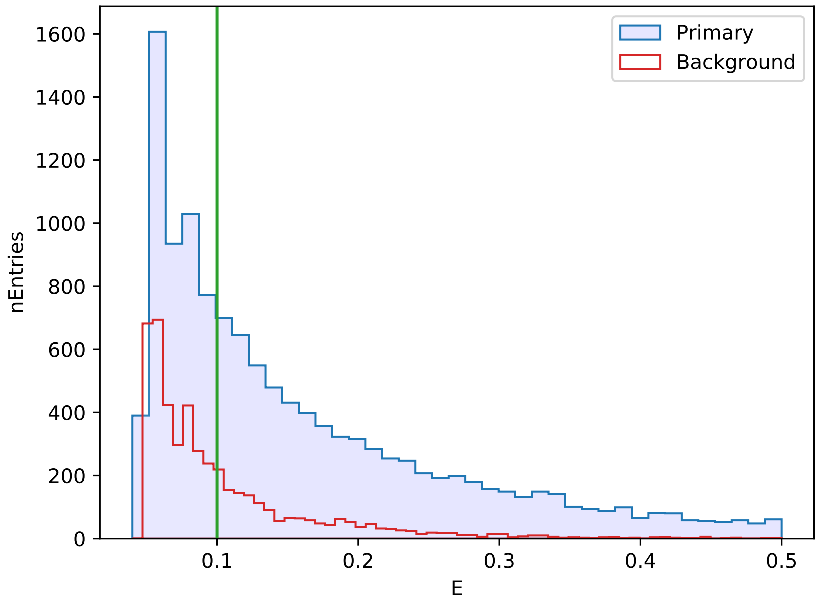


Figure 6.4: Distribution of ECL cluster energies for primary (blue) and background (red) photons. The green line shows the conditional threshold at 0.1 GeV applied to several tested ROE filters.

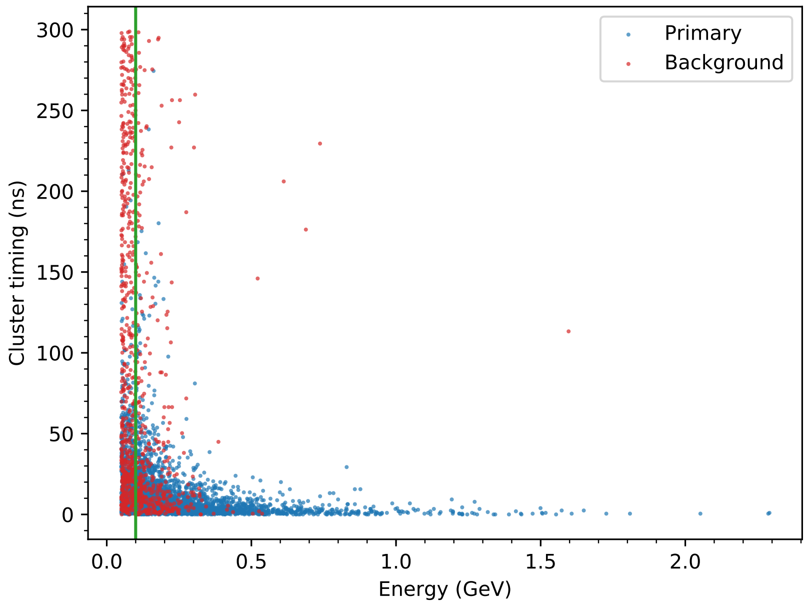


Figure 6.5: Comparison of ECL cluster energy to ECL cluster timing for primary (blue) and background (red) photons. The green line shows the conditional threshold at 0.1 GeV applied to several tested ROE filters.

Table 6.3: Resulting figure of merit ($\frac{S}{\sqrt{B}}$) for each rest of event filter tested in section 6.2.3. UpsilonROE_0 is the benchmark, with UpsilonROE_2 obtaining the best results.

Filter	Figure of merit
UpsilonROE_0	147
UpsilonROE_1	218
UpsilonROE_2	223
UpsilonROE_3	146
UpsilonROE_4	190
UpsilonROE_5	197

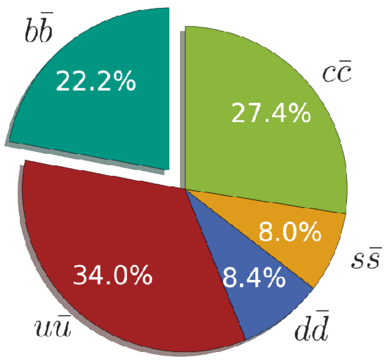
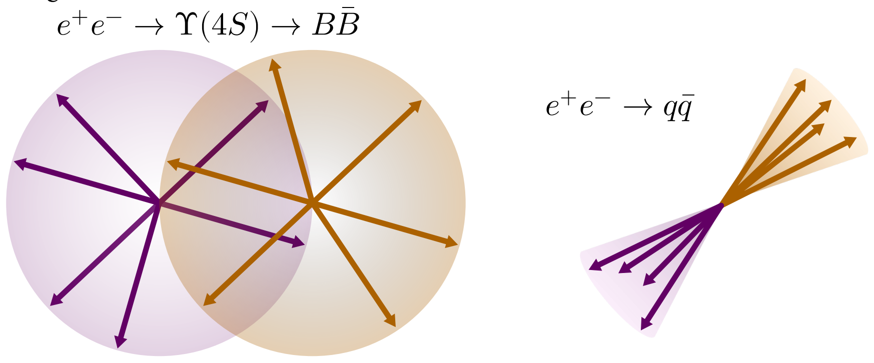


Figure 6.6: Relative cross sections of $e^+e^- \to q\bar{q}$ transitions at Belle II.



Spherical Jet-like

Figure 6.7: Typical $B\bar{B}$ and continuum decay shapes.

6.2.4 Continuum suppression

Continuum represents the dominant background at Belle II due to the significant cross section of $e^+e^- \to q\bar{q}$ transitions, where q is one of u, d, c, s. Figure 6.6 shows the relative cross sections of each quark pair production channel. It is named as such because of its continuous presence across the entire energy range explored at SuperKEKB. Figure 4.1 shows the Υ mass energy spectrum. The red dotted line indicates the $\Upsilon(4S)$ production threshold, the purple peak the $\Upsilon(4S)$ events which are the target of this study, and the grey shaded line the centre of mass (CoM) energy of SuperKEKB at 10.58 GeV. This CoM energy is roughly double the B meson mass, therefore the production and hadronisation of two quarks lighter than the b quark results in a relatively large excess kinematic energy in the decay products. This translates to a jet-like decay shape of the $q\bar{q}$ in the centre of mass reference frame, an example of which is shown in figure 6.7. The spherical decay of the B meson pair can be used to distinguish between continuum and $\Upsilon(4S)$ events.

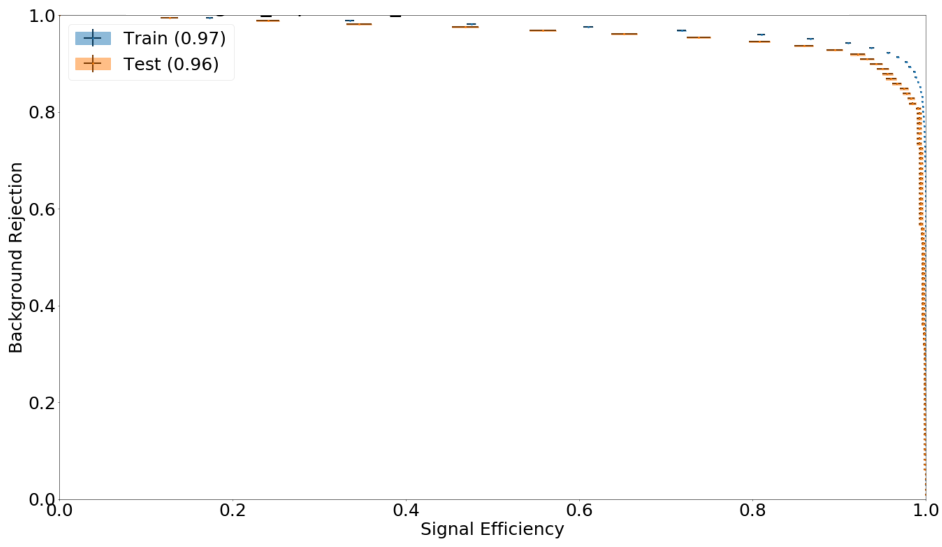
To perform this distinction, I train a fast Boosted Decision Tree (fBDT) [41] on a series of kinematic variables calculated from both the B_{sig} and B_{tag} daughters of the $\Upsilon(4S)$. The variables are the CleoCones [45], Kakuno-Super-Fox-wolfram moments [46], and thrust vectors [47]. I pass the set of all of these variables for a given signal candidate into the fBDT and a probability of the event not being continuum is returned. I perform the training on approximately $\mathcal{O}(10^5)$ signal and $\mathcal{O}(10^6)$ background events. I then evaluate the trained fBDTs on an independent sample containing roughly 10% of the number of training events. Figures 6.8 and 6.9 show the receiver operating characteristic (ROC) curves for each of the four signal channels' trainings. The area under curve (AUC) shown in each ROC figure provides a metric for quantifying the performance. All channels achieve excellent AUC scores, indicating strong signal-continuum discrimination. I also check the trainings for signs of overfitting. The ROC curve for the K^0 channel appears to have a difference in classification ability between training (blue) and validation (orange). Figures 6.10 and 6.11 shows the difference between the fBDT prediction outputs for the training data set (light shaded) and the independent validation set (dark crosses). Here the blue is for signal events and orange for continuum. Underneath each plot is the difference between the training and validation outputs. I observe no overfitting, with almost all outputs showing less than 1% difference. In the K^0 channel several bins show a difference of 2%, with the remainder less than 1%. Therefore, I accept this difference, and in future iterations of this study when more Monte Carlo simulations are available these trainings can be repeated on larger data sets to further reduce this difference.

Even though the input variables of the fBDT are calculated from both the B_{sig} and B_{tag}, only information from the B_{tag} provides an accurate measure of the sphericity of signal decays. This is because the reconstructed B_{sig} has only one daughter (with at most two final state particles) from which to calculate the kinematics. Therefore, I expect the training to also be sensitive to $B\bar{B}$ backgrounds as these will almost always produce an apparently non-spherical B_{tag} decay shape. An example of the predictive output of the fBDT is shown in figure 6.12. There not only does the continuum peak at zero as expected, but also both channels of the $B\bar{B}$ background. The signal and self-cross-feed (SCF) both show similar shapes, with the correctly reconstructed signal showing a smaller tail-end peak at zero, as expected. The other signal channels show similar distributions.

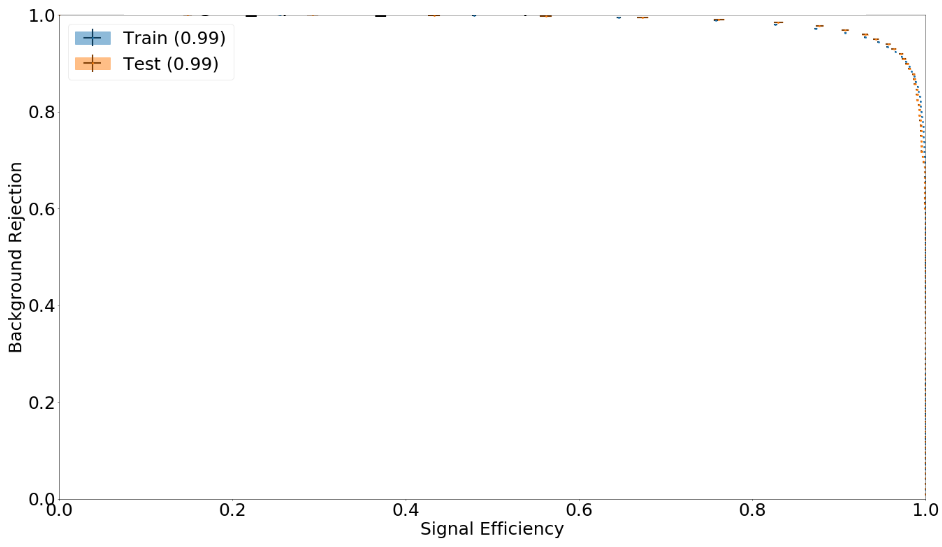
6.2.5 Selection optimisation

In this study I perform a cut-based selection. I place upper and lower bounds on individual variables useful for discriminating between signal and background events. The selection I optimise after all of the previous reconstruction stages are performed. This allows simultaneous optimisation of each cut to ensure maximum signal purity is achieved.

In total I select eight variables for optimisation. They describe the single kaon daughter of the reconstructed B_{sig}, the B_{tag}, and the $\Upsilon(4S)$ candidate individually. The variables

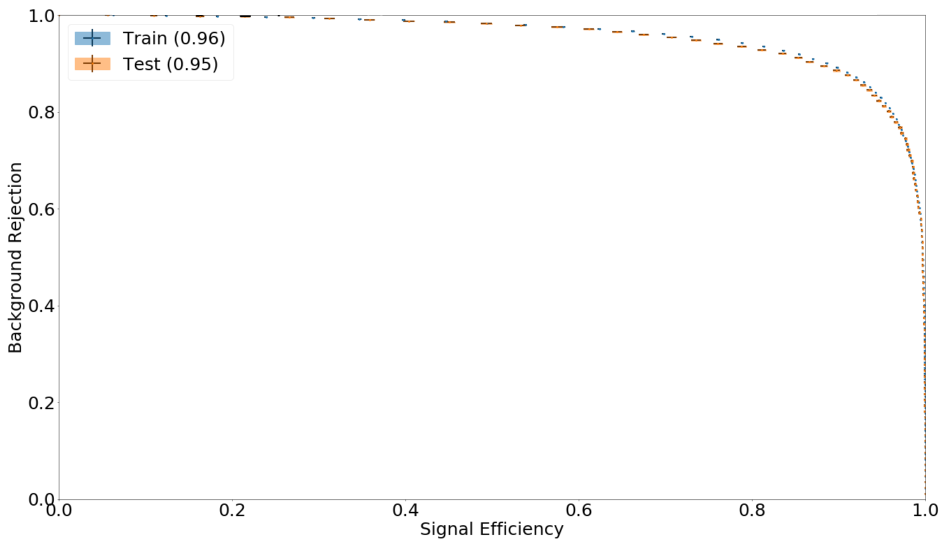


(a) K^0

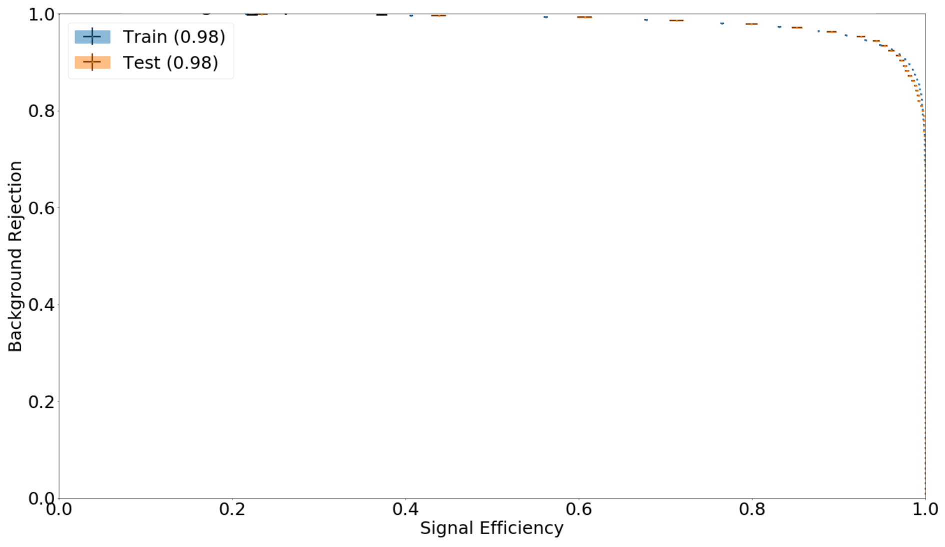


(b) K^+

Figure 6.8: Receiver operating characteristic (ROC) curves for fast boosted decision tree (fBDT) trainings of K^0 and K^+ channels in continuum suppression training. The blue curve shows the fBDT performance on the data used to train it and the orange curve shows the performance on independent validation data. The values shown in the legend are the area under curve (AUC) of both samples.

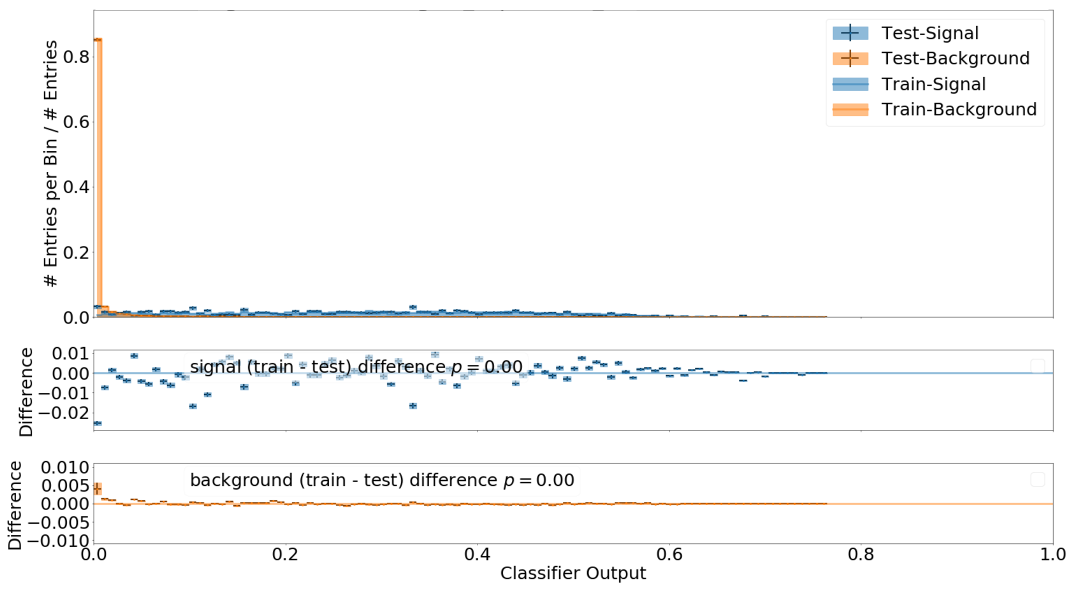


(a) K^{*0}

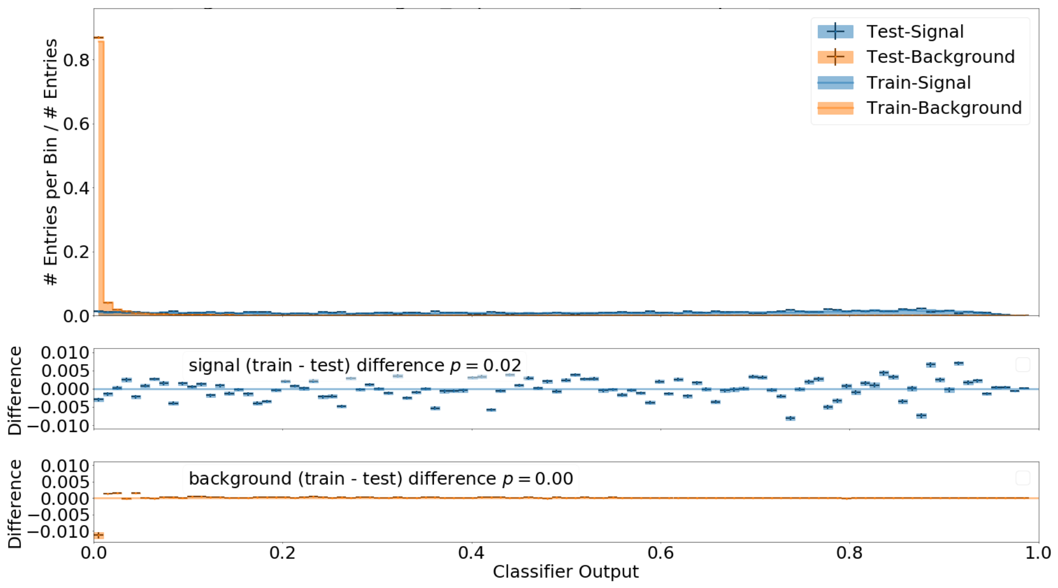


(b) K^{*+}

Figure 6.9: Receiver operating characteristic (ROC) curves for fast boosted decision tree (fBDT) trainings of K^{*0} and K^{*+} channels in continuum suppression training. The blue curve shows the fBDT performance on the data used to train it and the orange curve shows the performance on independent validation data. The values shown in the legend are the area under curve (AUC) of both samples.

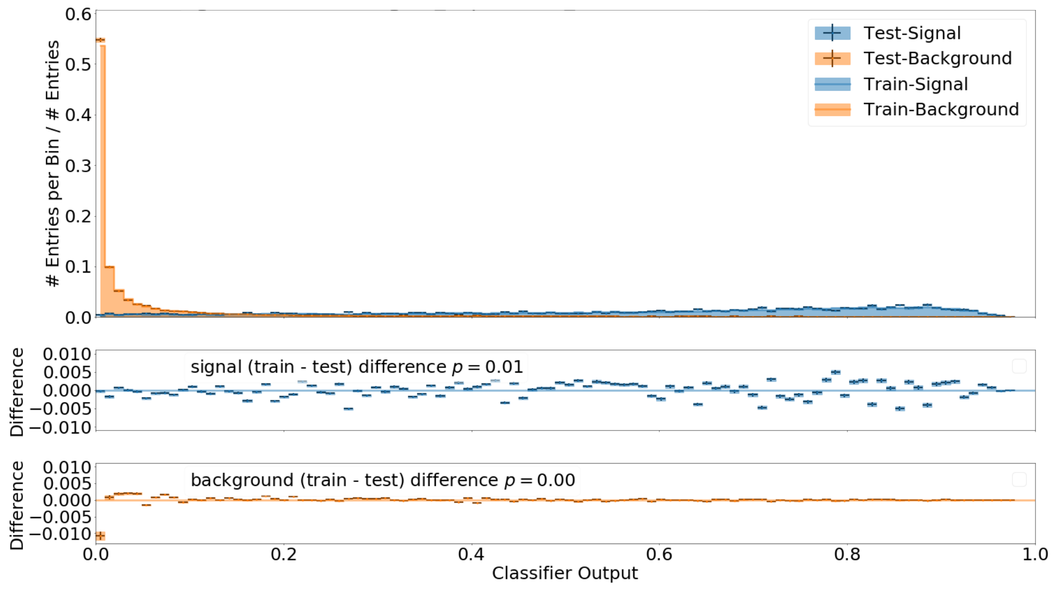


(a) K^0

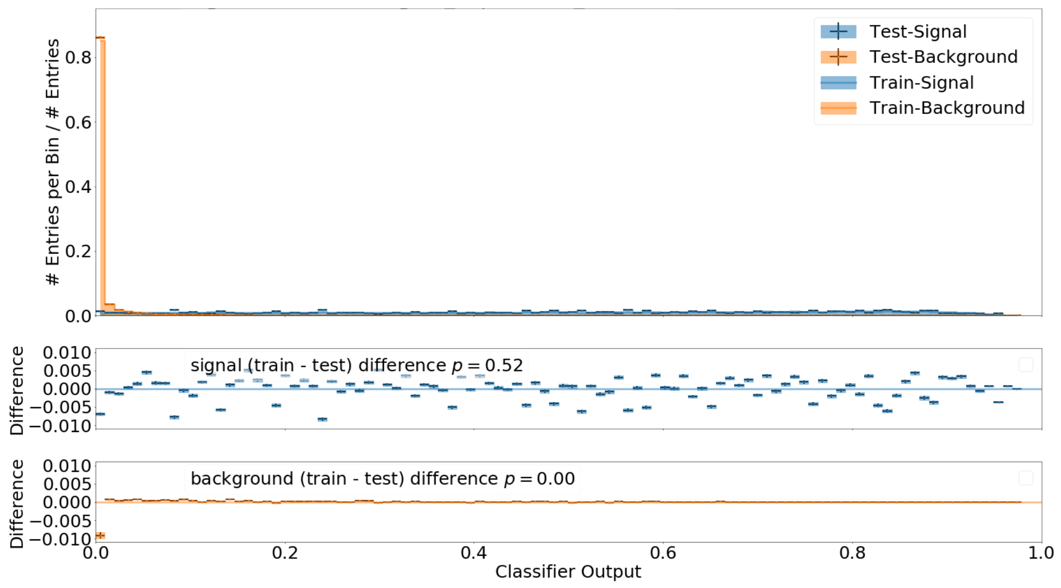


(b) K^+

Figure 6.10: Corresponding overtraining checks for trainings shown in figure 6.8. The blue shows the fBDT performance on signal data and the orange shows the performance on background.



(a) K^{*0}



(b) K^{*+}

Figure 6.11: Corresponding overtraining checks for trainings shown in figure 6.9. The blue shows the fBDT performance on signal data and the orange shows the performance on background.

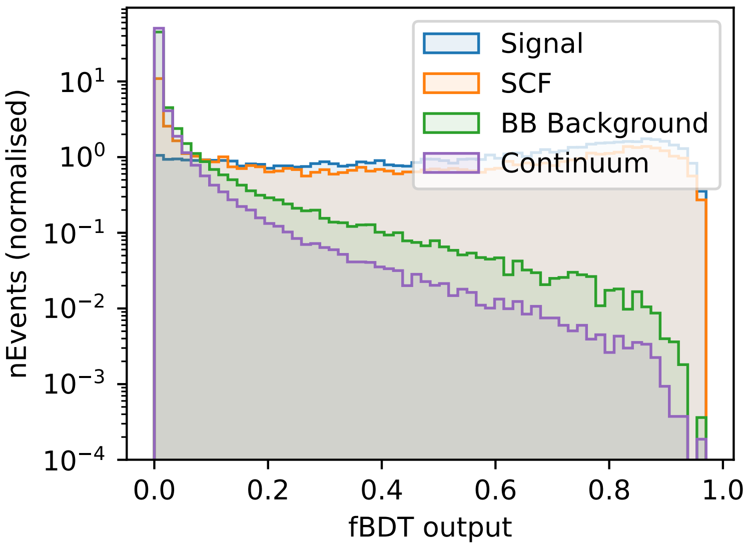


Figure 6.12: Example fBDT prediction output when applied to $B^+ \to K^+\nu\bar{\nu}$ reconstructed candidates. The values shown here are for candidates with the preliminary selections outlined in section 6.2.5 applied.

describing the B_{sig} daughter are:

p_{CMS} The reconstructed kaon momentum in the centre-of-mass (CoM) frame.
I expect Kaons originating from signal decays to be more energetic than those coming from other B meson decays, e.g. sub-decays involving a D meson. Therefore I search for a cut on the low-end.

Kdaughter angle The cosine of the angle between the signal-side kaon daughters.
For correctly reconstructed kaons I expect this to peak at one, as the daughters will be boosted in the flight direction of the parent kaon. For incorrectly reconstructed kaons, I expect the daughter angles to be distributed more uniformly from -1 to 1.

ROE(E) Energy remaining in the detector once the B_{sig} is removed.
This is the same as the rest of event described in section 6.2.3 but applied to the B_{sig} only instead. I apply the same restrictions used in the optimised UpsilonROE_2 filter. For signal events this should be physically limited to the rest mass of the B meson, i.e. everything else in the detector comes from the B_{tag}. In practice the ROE filter is not perfect and the optimal upper limit can be slightly higher.

Those describing the B_{tag} candidate are:

$M_{\mathbf{bc}}$ Beam constrained mass of the reconstructed B.
Defined as $M_{bc} = \sqrt{E_{beam}^2 - p_B^2}$, where $E_{beam} = \frac{\sqrt{s}}{2}$ is half the initial beam energy, and p_B is the reconstructed three-momentum in the CoM frame. This allows an invariant mass measurement of the reconstructed B meson with the known energy of the beam substituted for the measured energy of the B. For correctly reconstructed B mesons this will peak at the nominal B mass.

sigProb The signal probability given by the FEI reconstruction.
As this is a score by FEI of how likely it believes the B_{tag} was correctly reconstructed this is a natural addition to optimise selection for.

Finally, the variables describing the reconstructed $\Upsilon(4S)$ are:

E Reconstructed energy of the $\Upsilon(4S)$.
I expect this to be higher for a correctly reconstructed $\Upsilon(4S)$, with only the energy of the two signal-side neutrinos missing. For misreconstructions, for example missing muons which escaped the detector, the total energy should be lower.

$fBDT$ Fast Boosted Decision Tree (fBDT) calculated probability of an event originating from continuum or not.

$E_{\mathbf{miss}} + P_{\mathbf{miss}}$ The sum of the missing energy and three-momentum of the $\Upsilon(4S)$ in the centre of mass frame.
The E_{miss} (P_{miss}) is the difference between initial and reconstructed energy (momentum). This is expected to be less correlated to the $\nu\bar{\nu}$ mass than E_{miss} or P_{miss} alone [35] which makes it a suitable selection for model independent analyses.

Note that Kdaughter angle is not available for K^+ channel. The choice of these variables is motivated in part by those used in previous analyses' selections as well as those proposed in $B \to K^{(*)}\nu\bar{\nu}$ Belle II prospects estimates in [35]. Unlike the previous hadronic tagging study performed at Belle, here I have not included selection cuts on $\Delta(E)$ as tightening those applied to the skims in section 6.2.1 appears to provide no benefit.

Examples of the characteristic shapes of each variable before selection optimisation are shown in figs. 6.14 and 6.15 for the K^0 channel. The shapes are similar for all other channels and included in Appendix A (figs. A.1 to A.6) for completeness. The data used to generate these shapes and used to perform the cut search discussed below includes the following preliminary selections to reduce processing times:

- No remaining charged tracks present after $\Upsilon(4S)$ extraction.
  This utilises the rest of event charge track requirement from section 6.2.3 to ensure all particles from the primary physics event were used in the $\Upsilon(4S)$ reconstruction.

- $M_{\mathrm{bc}} > 5.265\,\mathrm{GeV/c^2}$
  Recalling the M_{bc} distribution from figure 6.1, the correctly reconstructed B_{tag} candidates lie almost entirely in this region.

- sigProb > 0.005
  Similar to the M_{bc} motivation, this requirement excludes the false B_{tag} dominated region of sigProb.

Additionally, I only consider the low-energy region of the fit variable, $E_{\mathrm{ECL}} < 1.0\,\mathrm{GeV}$, as this will ultimately be the fitted region in Chapter 7. Overall these preliminary selections keep approximately 75% of correctly reconstructed signal events from the initial FEI skims and remove roughly 99.5% of background events.

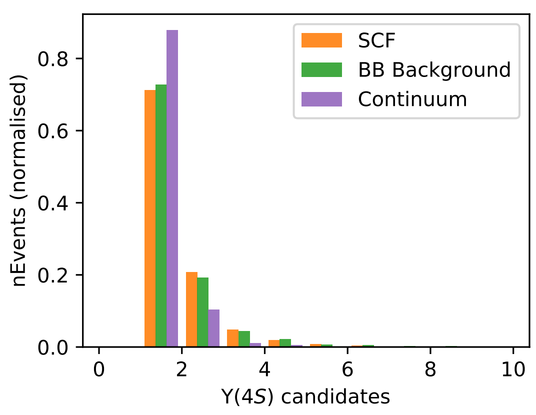
The resulting distributions of number of $\Upsilon(4S)$ candidates per event for each signal channel after these preliminary selections is shown in figure 6.13. I have truncated the number of candidates at ten for clearer visualisation. Signal (correctly reconstructed) is not shown as there is trivially only one candidate at most per event. To handle events with multiple $\Upsilon(4S)$ candidates I perform a best candidate selection. I choose the candidate remaining *after* all cuts have been applied with the highest FEI signal probability (sigProb) as the final signal candidate. The sigProb ranking is performed by FEI such that only one candidate per rank exists, i.e. two $\Upsilon(4S)$ candidates with the same B_{tag} daughter (and different B_{sig} daughters), are given adjacent ranks randomly. The cut on $fBDT$ output ensures this is possible without the B_{sig} candidate being incorrectly chosen at a high rate. The result is no signal Monte Carlo $K^{*+} \to K^+\pi^0$ events falsely chosen as $B^+ \to K^+$ signal candidates, and $K^{*0} \to K^0_S\pi^0$ misreconstructed as $B^0 \to K^0_S\nu\bar{\nu}$ only 8% of the time. The restriction placed on remaining charged tracks removes the potential for overlap of B_{sig} candidates with only charged final state particles. Therefore no further B_{sig} candidate selection is required.

One key veto present in the previous semileptonic tagging study at Belle [2] which I have not applied is the discarding of events containing a K^0_L. At the time of this study no usable K^0_L identification information has been developed for Belle II. This feature is under active development and expected to be available by the beginning of phase 3 data taking in early 2019 [35].

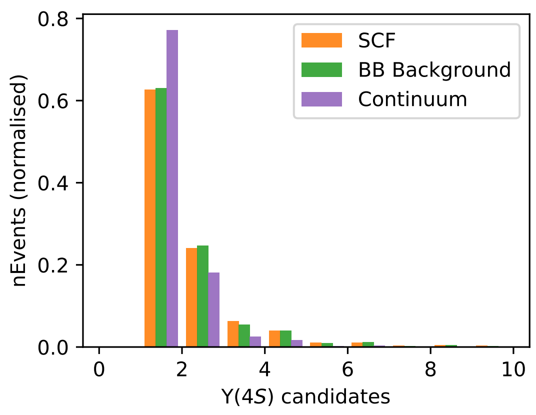
I optimise the cut selections using a random search to maximise a figure of merit (FOM). The FOM is calculated as

$$\mathrm{FOM} = \frac{S}{\sqrt{B}}, \tag{6.2}$$

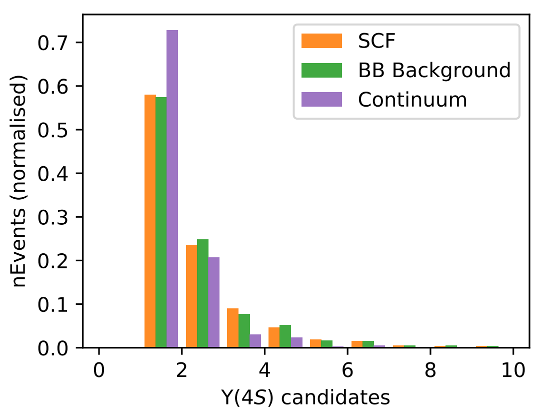
where S is the number of signal events and B is the number of background events. I perform the best candidate selection described above after each trialled cut and before the FOM calculation in each step of the cut search. I perform the random cut search in eight iterations of 4096 randomly selected sets of cuts, with each iteration reducing the available cut parameter space to focus in on the FOM maximum. I repeat the cut search several times to reduce the chance of



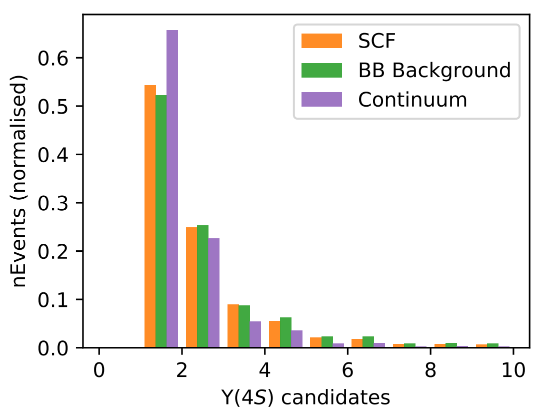
(a) K^0



(b) K^{*0}



(c) K^+



(d) K^{*+}

Figure 6.13: Normalised distribution of number of $\Upsilon(4S)$ candidates per event for each signal channel after preliminary selections described in section 6.2.5 have been applied.

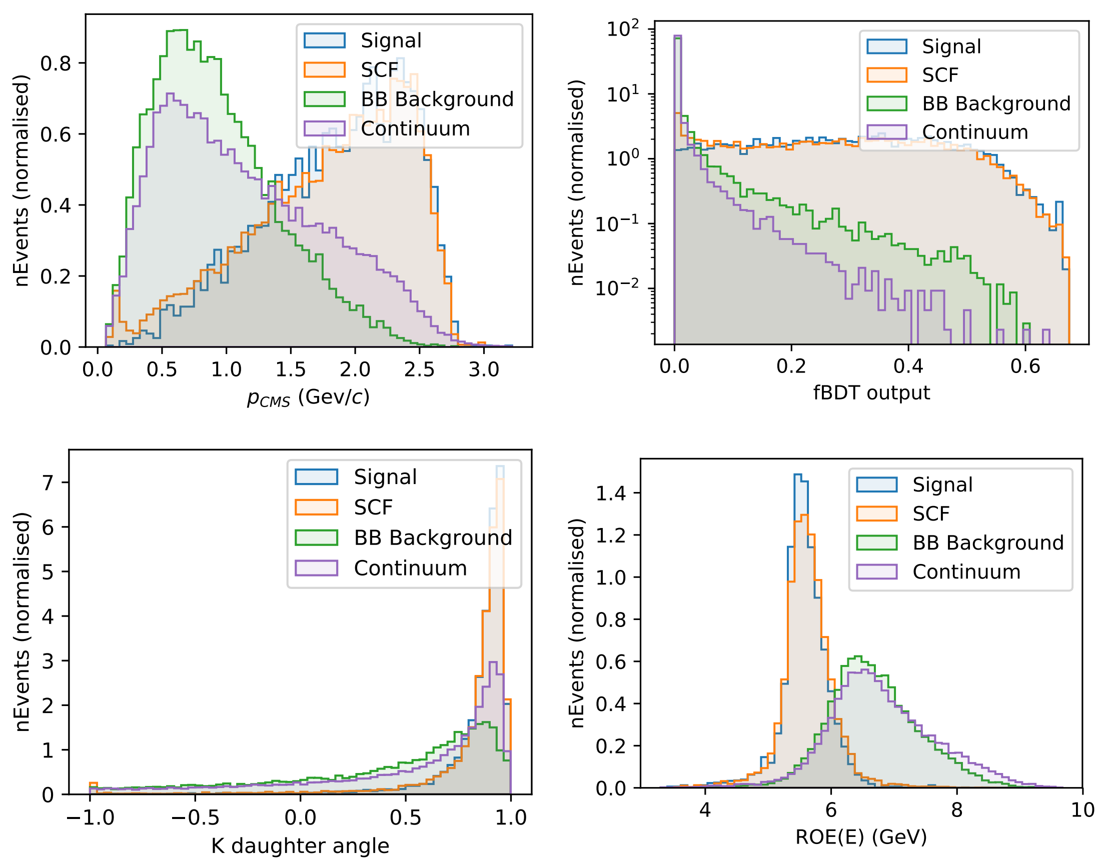


Figure 6.14: K^0 hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots have been normalised to show relative number of events across the full range.

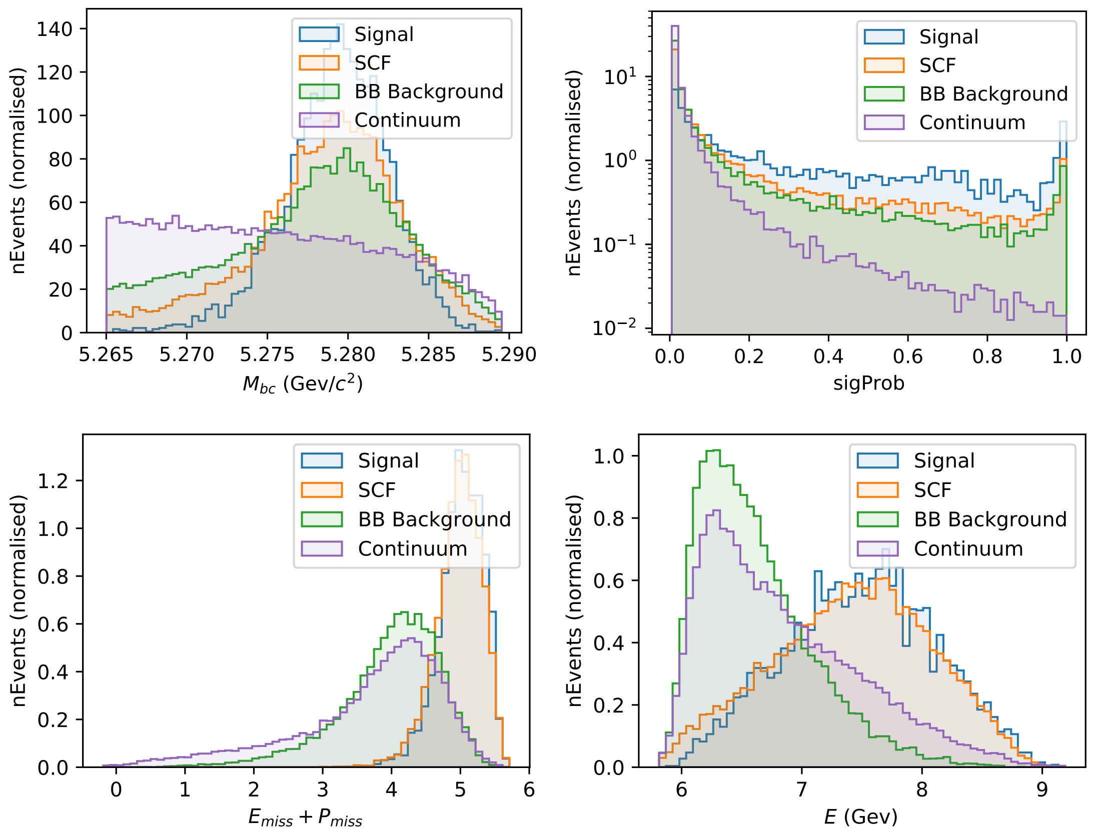


Figure 6.15: K^0 hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots have been normalised to show relative number of events across the full range.

Table 6.4: Efficiency of each channel and mode after optimised cut selections and best candidate selection in section 6.2.5 have been applied.

Channel	Signal	SCF	Background	Continuum
K^+	8.71×10^{-4}	3.98×10^{-4}	2.31×10^{-7}	3.19×10^{-8}
K^0	2.15×10^{-4}	3.34×10^{-4}	2.96×10^{-8}	3.36×10^{-10}
K^{*+}	1.05×10^{-4}	4.78×10^{-5}	6.48×10^{-8}	1.11×10^{-8}
K^{*0}	3.17×10^{-4}	1.11×10^{-4}	1.39×10^{-7}	1.75×10^{-8}

Table 6.5: Belle 2013 hadronic tag [23] efficiencies. As no official numbers are given for the background efficiencies I have estimated them from the total number of reconstructed events minus the mean of the reconstructed signal events.

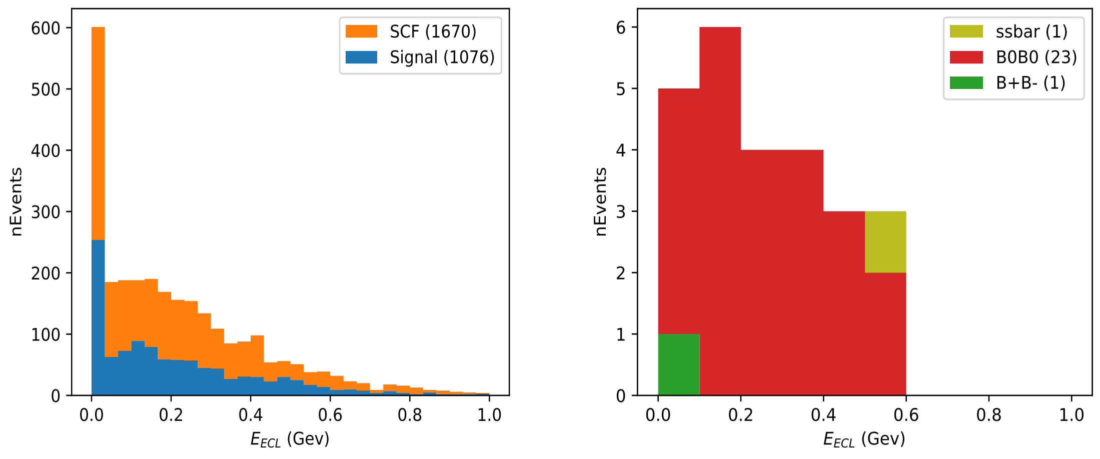
Channel	Signal	All backgrounds
K^+	5.68×10^{-4}	3.92×10^{-8}
K^0	0.84×10^{-4}	2.85×10^{-9}
K^{*+}	1.47×10^{-4}	2.72×10^{-8}
K^{*0}	1.44×10^{-4}	1.30×10^{-8}

identifying a local FOM maximum and the resulting efficiencies (after best candidate selection) are shown in table 6.4. The efficiencies of the previous $B \to K^{(*)}\nu\bar{\nu}$ hadronic tag study performed at Belle [23] are shown in table 6.5 for comparison. This provides a sanity check that the FEI and cuts are performing at a reasonable efficiency level, recalling the signal-background event volume difference discussed in section 6.1. As no official efficiencies for the backgrounds were given from the previous study, I have estimated the combined background efficiency values shown in table 6.5 from the total number of reconstructed events minus the mean of the reconstructed number of signal events. The optimised cut values for each signal channel are shown in Appendix B.

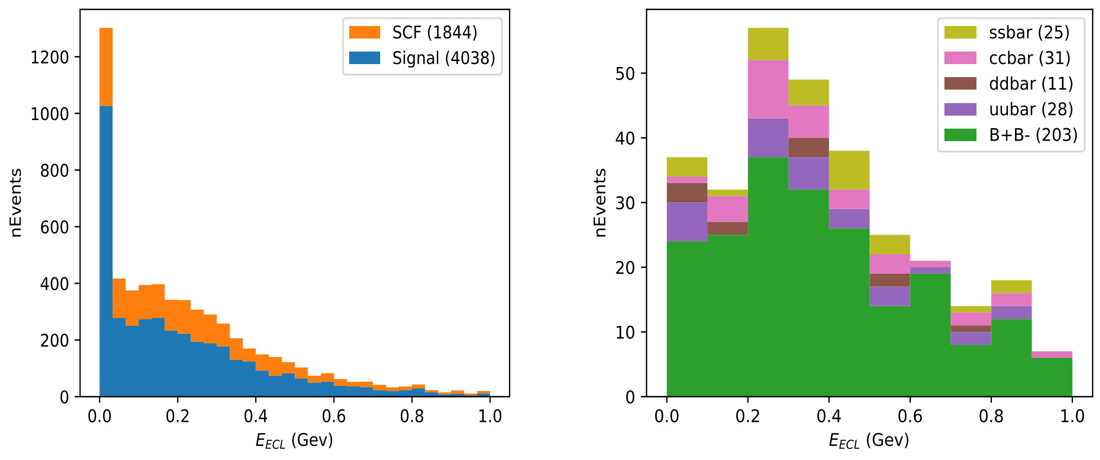
Comparing the efficiencies for both signal and background, this study is performing at a comparable level to the previous. The efficiencies of this study are all marginally higher than that of the previous, likely due to the significantly increased beam-related backgrounds counterbalancing the improved reconstruction efficiency of the FEI tagging algorithm. The fit results in Chapter 7 give a more quantitatively meaningful comparison as the efficiencies do not account for the distributions of the fitted E_{ECL}.

Applying the optimised cuts to the full set of Monte Carlo events, the resulting E_{ECL} distributions are shown in figs. 6.16 and 6.17. Each channel in the legends show the number of candidates remaining from the original event volumes simulated in table 6.1. Inspecting the shapes of each channel in E_{ECL} I can verify that they match expectations. The signal shows the characteristic peak at zero, with the non-zero tail due to imperfect requirements placed on the rest of event ECL clusters. Due to the requirement that no tracks remain in the event after the $\Upsilon(4S)$ has been reconstructed, it is reasonable that the ROE signature of the self-crossfeed will be similar to that of the signal. They are still signal events and so by definition are cases in which the signal side kaon was falsely attributed to the the B_{tag}, and correspondingly a kaon from the B_{tag} was falsely attributed to the B_{sig}. This type of misreconstruction of the SCF is independent of the rest of event contents. The background shapes all appear to have non-zero-peaking structures as expected. The low number of remaining events, however, makes the interpretation of any parametrisable shape difficult. All previous studies suffered from a similar problem, despite having significantly larger Monte Carlo simulation sample sizes, and opted for histogram templates to describe the background shapes. The K^0 channel has no background events remaining in the higher energy region of E_{ECL}. This is due in part to the lower volume of Monte Carlo mixed background simulation available (the dominating BB background for the neutral signal channels), as well as the K_S^0 channel having a much better background suppression than all other channels. A similar outcome regarding the latter point was seen in the previous hadronic tag study at Belle, with the K_S^0 channel having both the lowest signal and background efficiencies. I believe this is due to a combination of the FEI hadronic B^0 reconstruction having lowest retention rate, the K_S^0 being only reconstructed from two charged pions with $\mathcal{B}(K_S^0 \to \pi^+\pi^-) = 69.2\%$, and K_S^0 having a flight time which is used to check for a valid decay vertex during its reconstruction. The last point reduces the number of combinations of charged pion daughters that can produce a K_S^0 candidate.

To check the impact that the choice of rest of event filter has on E_{ECL} I inspect the change in E_{ECL} when the baseline UpsilonROE_0 and second-best performing UpsilonROE_1 ROE filters from section 6.2.3 are applied. Figure 6.18 shows the changes to E_{ECL} for signal and background events separately in the K^+ channel, which can be compared with figure 6.16b. All channels show a shift towards a higher measured E_{ECL} as the ROE restrictions are relaxed. This indicates that the measurement will be highly sensitive to which filter is applied, or conversely, the modelling of E_{ECL} from Monte Carlo relies heavily on the simulation accurately describing the beam-related backgrounds at Belle II. Given the extent of background underestimation found in [33], it's likely they will be even higher still than expected in phase 3.

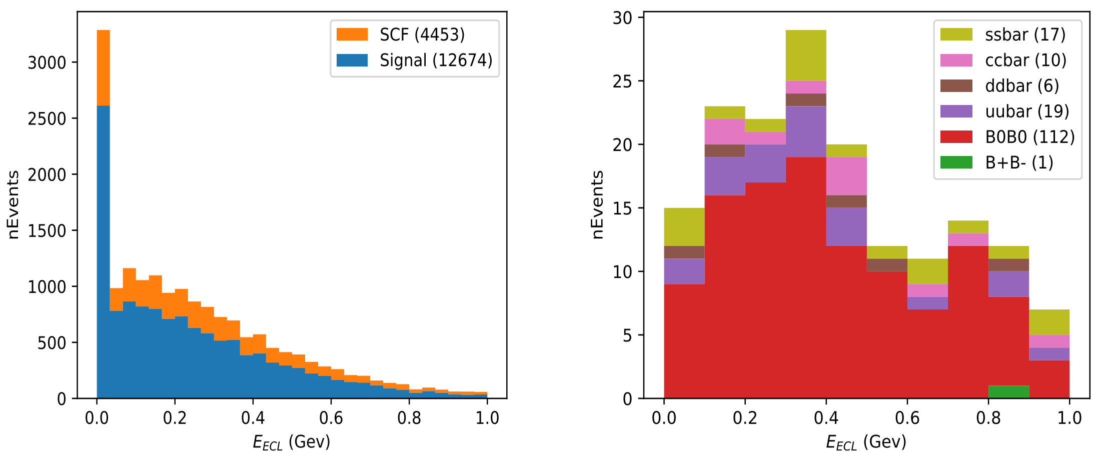


(a) K^0

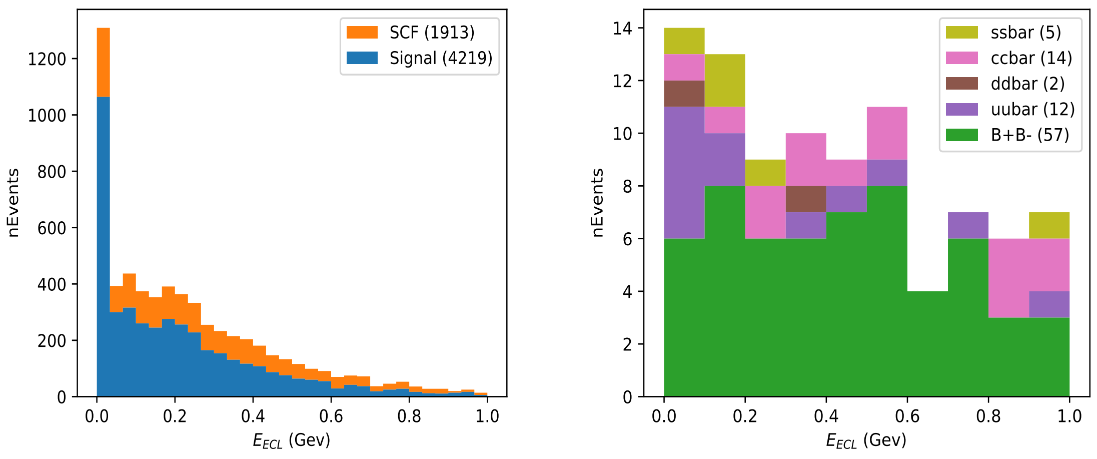


(b) K^+

Figure 6.16: Resulting E_{ECL} shapes (stacked) after cuts maximising the FOM $\frac{S}{\sqrt{B}}$ have been applied. Binning is selected to best demonstrate the distributions based on remaining number of events.

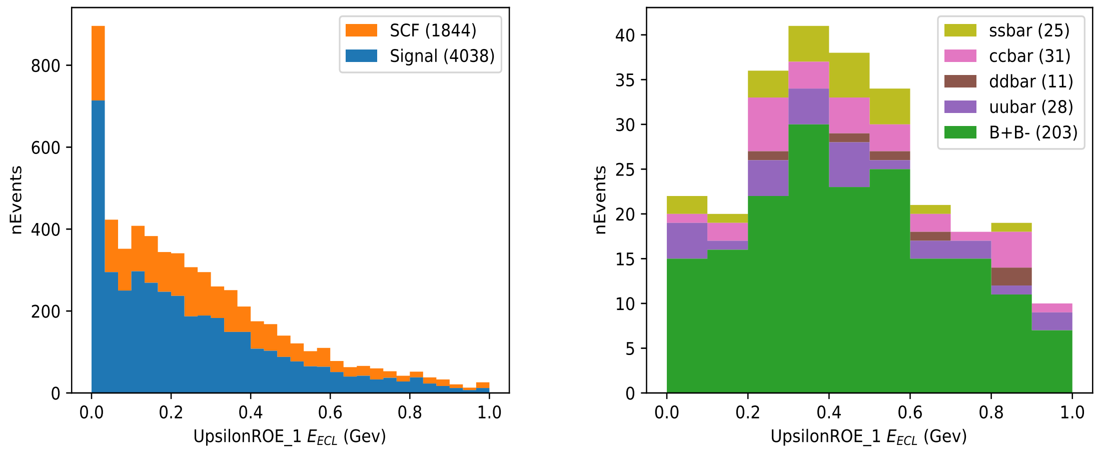


(a) K^{*0}

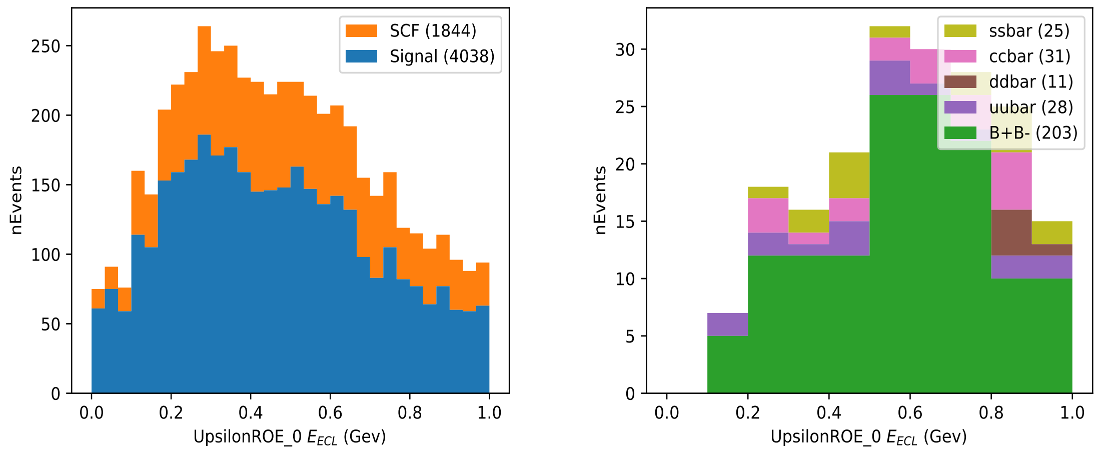


(b) K^{*+}

Figure 6.17: Resulting E_{ECL} shapes (stacked) after cuts maximising the FOM $\frac{S}{\sqrt{B}}$ have been applied. Binning is selected to best demonstrate the distributions based on remaining number of events.



(a) UpsilonROE_1



(b) UpsilonROE_0

Figure 6.18: Example of changes to E_{ECL} in the K^+ channel for different rest of event restrictions in the signal region ($E_{\text{ECL}} < 1\,\text{GeV}$). All histograms shown here are stacked. The top row shows the second best performing ROE filter from the tests in section 6.2.3. The bottom row shows the baseline filter. The left column shows the change to signal and self-crossfeed, and the right to the individual background channels.

Chapter 7

Fitting

The branching fraction of each $B \to K^{(*)}\nu\bar{\nu}$ channel is obtained by performing a fit to signal and background in E_{ECL}. The goal of the fit is to extract the number of signal events present in the sample, known as the signal *yield*. To do so I first create probability distribution functions (PDFs) that model the signal and background contributions to E_{ECL} individually (section 7.1). I then combine the PDFs with all shape parameters fixed except for a scaling factor of each component that represents the individual yields (section 7.1.1). After this I validate the combined PDFs in a series of toy MC test (section 7.2), and finally I estimate the expected fit sensitivity on real data (section 7.2.2).

As no phase 3 (physics run) data for Belle II exists at the time of writing, and due to the lack of enough background Monte Carlo to draw statistically significant sensitivity estimates, this fit is only able to provide a framework for preparing the final $B \to K^{(*)}\nu\bar{\nu}$ measurement when the volume of background Monte Carlo increases and the beam backgrounds due to the increased luminosity of Belle II are better understood.

7.1 Fit model

I expect E_{ECL} to peak at zero for signal and SCF candidates and show a flat or non-zero peak for background. Given the indistinguishable shapes of signal and SCF, I fit both together with a single parametrisation. Similarly, I combine all background channels into a single fit to combat the high statistical uncertainty that comes from fitting each individually. Previous studies [2, 23] fit both the background and signal with histogram templates of MC, therefore for this study I first try histogram templates as a fit method. Figure 7.1 shows the individual histogram templates obtained from MC for each of the signal channels. The blue shaded regions show the background templates, and the red shows signal scaled to a comparable size for clarity. The black line shows the combined fit to the data points, which contain the entire background

MC sample after selection and a comparably sized random subset of signal MC events. I also test the use of parameterised PDFs, using the expected shape of the signal to guide the PDF selection. Figure 7.2 shows the resulting PDFs; for signal, I used an exponential function; for background, I fit a kernel density estimator (KDE) as I don't know what structure to expect. The individual parametric fit components are included in Appendix D.

7.1.1 Yield fit

To extract the signal yield from observed events I perform a combined fit of both signal and background. I fix each components fit parameters and add the individual PDFs together with a scaling factor for each representing the yields,

$$P_{\text{fit}} = Y_{\text{sig}} P_{\text{sig}} + Y_{\text{bkg}} P_{\text{bkg}} \,, \tag{7.1}$$

where Y are the individual yields and P the PDFs. To find the Y_{sig} and Y_{bkg} parameters I perform a binned extended maximum likelihood fit, with the likelihood given by

$$\mathcal{L}(\lambda) = \frac{\lambda^N e^{-\lambda}}{N!} \prod_{i=1}^{N} P_{\text{fit}} \,, \tag{7.2}$$

where λ is the sum of events predicted in each category, N is the total number of observed events.

To calculate the significance of the fitted signal yield, I perform the toyMC experiments as described in section 7.2 with the simulated number of signal events set to zero and the repeat the fits. Using Wilk's theorem [48] with one degree of freedom, I extract the significance from the square root of the difference in log likelihoods of the fits to samples with and without simulated signal,

$$\text{Significance} \equiv \sqrt{\Delta(-\log(L))} = \sqrt{2 \log \frac{\mathcal{L}_{S+B}}{\mathcal{L}_B}} \,. \tag{7.3}$$

7.1.2 Upper limit calculation

As none of the $B \to K^{(*)} \nu\bar{\nu}$ transitions have been observed yet, it is reasonable to expect that in the first years of operation of Belle II no significant signal will be observed. Therefore I include a calculation of the signal yield upper limit at 90% confidence level (C.L.).

Following the test statistic prescribed by [49], I constructed a profile likelihood as the ratio

$$q_{Y_{\text{sig}}} = -2 \ln \left(\frac{\mathcal{L}(E_{\text{ECL}}; Y_{\text{sig}}, \theta'(Y_{\text{sig}}))}{\mathcal{L}(E_{\text{ECL}}; \hat{Y}_{\text{sig}}, \hat{\theta}(Y_{\text{sig}}))} \right) \,. \tag{7.4}$$

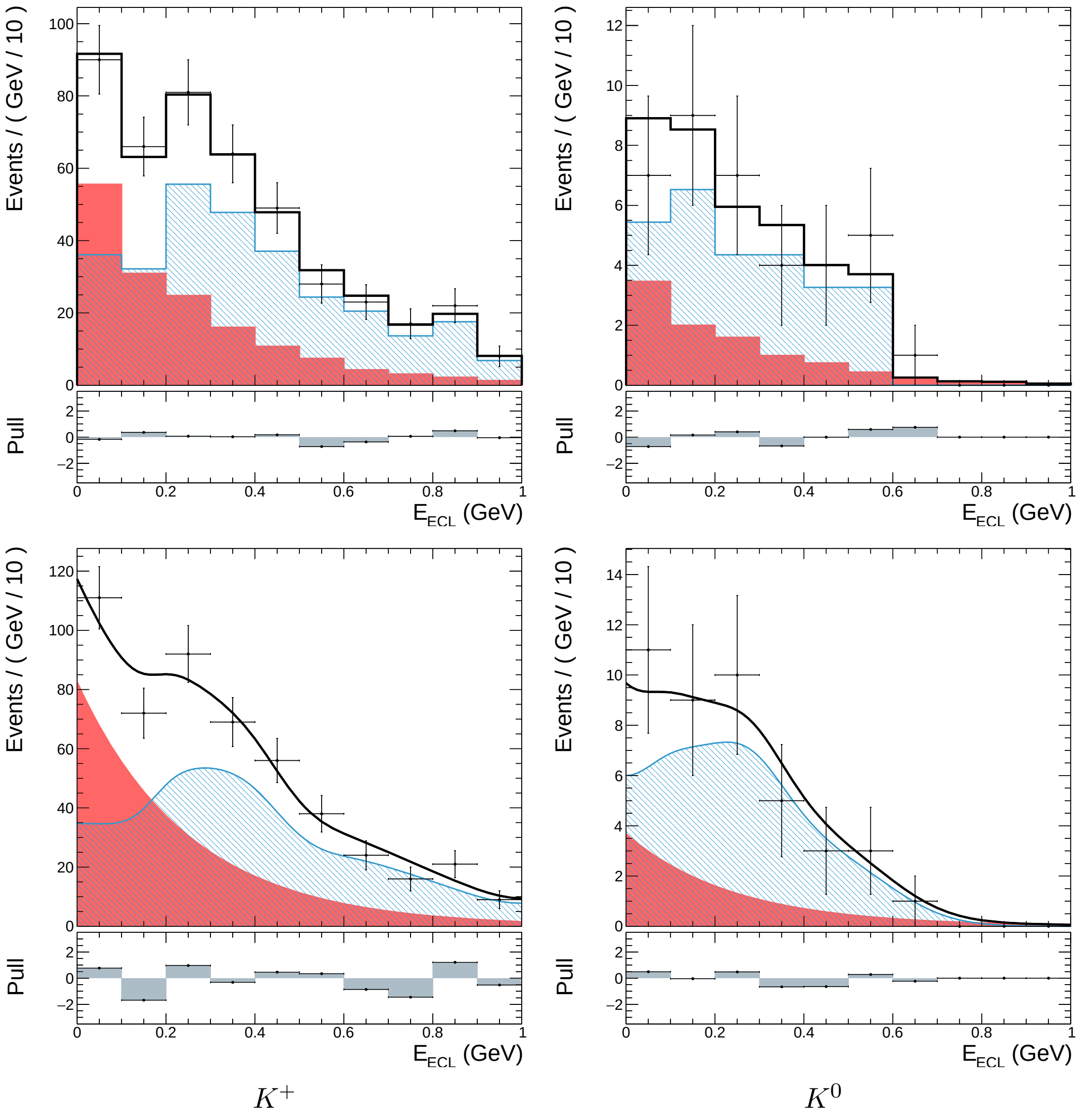


Figure 7.1: Histogram templates (top) and parametric fits (bottom) obtained individually from signal and background MC. Red shows the signal fits, blue the background. The solid black line shows an example of a combined fit to a sample of the MC (data points), along with the pull of the fit below each.

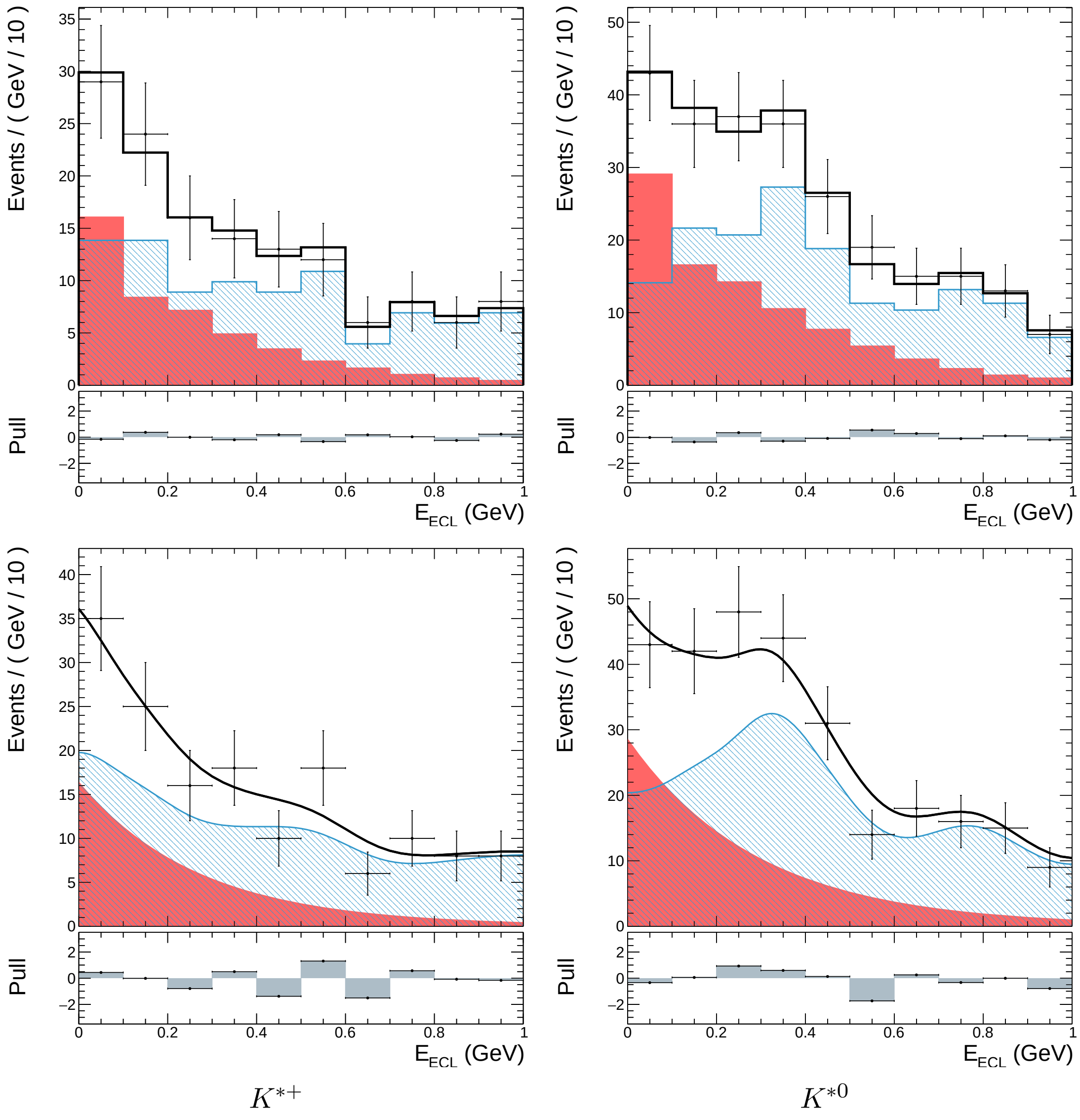


Figure 7.2: Histogram templates (top) and parametric fits (bottom) obtained individually from signal and background MC. Red shows the signal fits, blue the background. The solid black line shows an example of a combined fit to a sample of the MC (data points), along with the pull of the fit below each.

Y_{sig} is the upper limit on the signal yield being searched for, and θ represents the all other parameters given Y_{sig} (nuisance parameters). In the denominator Y_{sig} and θ are fit as $\widehat{Y}_{\text{sig}}$ and $\widehat{\theta}$ simultaneously to maximise their respective $\mathcal{L}$. $\theta'(Y_{\text{sig}})$ is the conditional maximum likelihood estimator, which maximises $\mathcal{L}$ in the numerator for a given Y_{sig}. The ratio approaches one if the hypothesised Y_{sig} agrees well with the data. The PDF $f\left(q_{Y_{\text{sig}}}\middle|Y_{\text{sig}}\right)$ of $q_{Y_{\text{sig}}}$ can be used to compute a p-value from

$$p_{Y_{\text{sig}}} = \int_{q_{Y_{\text{sig,obs}}}}^{\infty} f\left(q_{Y_{\text{sig}}}\middle|Y_{\text{sig}}\right) dq_{Y_{\text{sig}}}\,, \tag{7.5}$$

where $Y_{\text{sig,obs}}$ is the actual number of fitted signal events from data. From Wilks [48] and Wald [50], for a sufficient test statistic $f\left(q_{Y_{\text{sig}}}\middle|Y_{\text{sig}}\right)$ follows a χ^2 distribution, in this case for one degree of freedom. I then compute the p-value such that the 90% C.L. is set at the highest fitted Y_{sig} for which the p-value is not less than 0.10.

Once the upper limit on signal yield has been calculated I convert it to an upper limit on the branching fraction to allow direct comparison with current upper limits and the Standard Model predictions. The conversion is calculated as

$$\mathcal{B}_{UL} = \frac{Y_{\text{UL}}}{\epsilon_{\text{sig}} N_{B\bar{B}} N_{\text{lum}}}\,, \tag{7.6}$$

where $\mathcal{B}_{\text{UL}}$ is the branching fraction upper limit given observed signal yield N_{UL} at integrated luminosity N_{lum}, ϵ_{sig} is the signal reconstruction efficiency obtained in section 6.2.5, and $N_{B\bar{B}}$ is the number of simulated $B\bar{B}$ pairs.

7.2 ToyMC tests

To probe the upper limit and yield fit significance I perform a series of toyMC experiments (sometimes referred to as pseudo-experiments) using the two fit models from section 7.1. The ToyMC experiments involve the repeated simulation and fitting of the fit variable, E_{ECL}, based on the individual signal and background fit models obtained from Monte Carlo. The individual yields of each component are simulated according to a Poisson distribution with expectation values set according to standard model predictions. This allows the uncertainty in the expected number of signal and background events to be included in the toyMC experiments.

The goal here is two-fold:

1. Check that the fit models chosen are robust enough to handle expected fluctuations in the data from the simulation.

2. Obtain an estimate of the fit sensitivity at varying luminosities of Belle II based on the expected number of events assuming standard model (table 6.1).

For each signal channel I simulated 5000 toyMC experiments at each integer integrated luminosity equivalent ranging form 1 - 25 ab^{-1}. From these the pull distribution for each channel was calculated to verify the fits. Following the success of these the branching fraction upper limits and signal yield significance was calculated.

7.2.1 Fit robustness tests

To verify the robustness of the combined fit model, and check for fit biases, the pull distributions of the fitted signal and background yields are recorded. The pull is defined as

$$\text{Pull} = \frac{x - \mu}{\sigma}, \tag{7.7}$$

where x in this case is the measured (fitted) value of Y_{sig}, simulated with mean μ and standard deviation σ. In the case of no biases the pull will correspond to a Gaussian distribution with a mean of zero and a standard distribution of one.

Figures 7.3 and 7.4 show the pull distributions resulting from fits to the toyMC experiments at 1 ab^{-1} using histogram templates, and Figures 7.5 and 7.6 show the same using the parametric models. All channels in the histogram fits except the K^0 show a reasonable agreement with a zero mean and unit standard deviation, with the agreement only increasing at higher simulated luminosity equivalents. Figures E.1 to E.4 in Appendix E show the corresponding histogram template pull distributions at 2 and 5 ab^{-1} for demonstration (K^0 has been neglected as the fit is unstable). The highest deviation appears to be in the signal yield pull of the K^{*0} channel, with a -0.067 mean. This corresponds to a consistent signal yield underestimation by 6.7% of the standard deviation of the expected signal yield. This is a relatively small effect and can reasonably be expected to diminish as Monte Carlo volumes are increased, therefore I make no further corrections to it.

In the parametric fits it appears as though there is a consistent systematic bias towards underestimation of the signal yield and a corresponding compensating overestimation of the background yield in all channels. The minimum in signal being -0.11 in the K^+ channel. Therefore I include the parametric fits in the upper limit test purely for completeness. Until a better understanding of the expected background shape can be obtained from increased Monte Carlo background simulations, however, the parametric fits don't appear to provide any benefit over histogram templates.

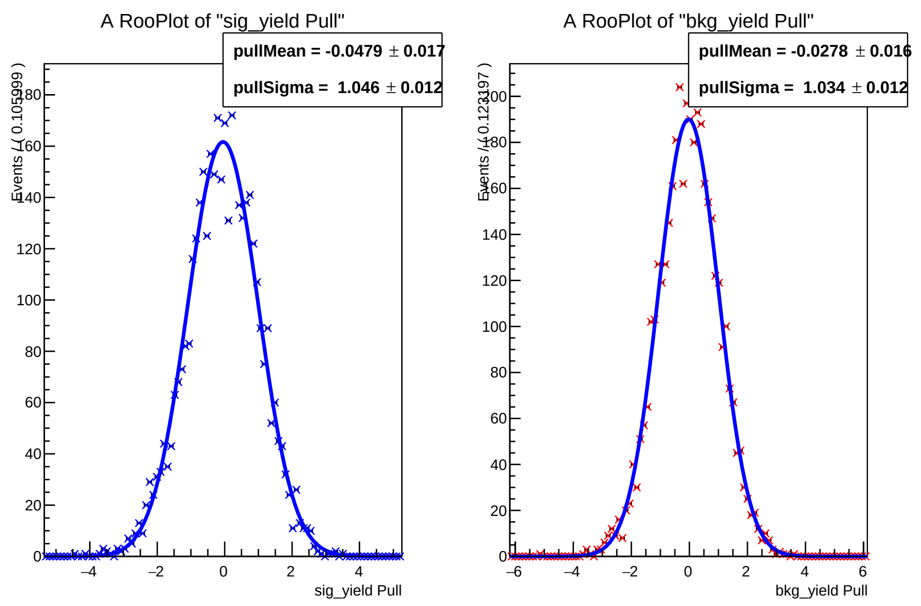
The K^0 histogram template appears to never produce a stable fit, regardless of the integrated luminosity simulated or the fit shapes used. My estimation is that this is an issue arising entirely from the volume of background Monte Carlo simulations available for the hadronically reconstructed neutral B mesons. Relative to the K^{*0} signal channel, the K^0 channel has a significantly lower background efficiency, i.e. a much higher purity resulting from the selection optimisation in section 6.2.5. Therefore not only is there no information about the contributions from continuum backgrounds, but the 24 $B\bar{B}$ background MC events remaining provide too little information to fit a statistically significant shape to. Inspecting fig. 6.16a, there is no information remaining in the bins above 0.6 GeV, resulting in a background fit dependent heavily on the signal populated region of E_{ECL}.

It is apparent at this point that the low volume of Monte Carlo simulated events is the restricting factor for continuing preparation of this analysis. Without a good statistical understanding of the background contributions to E_{ECL}, a stable fit, and hence branching fraction limit, cannot be extracted. For completeness I have implemented and performed the upper limit and signal yield significance scaling tests in the following section, however this study provides primarily a strong motivation for significant increases in the Monte Carlo simulation volume capacity of the Belle II collaboration. This point is discussed further in the Summary (Chapter 8).

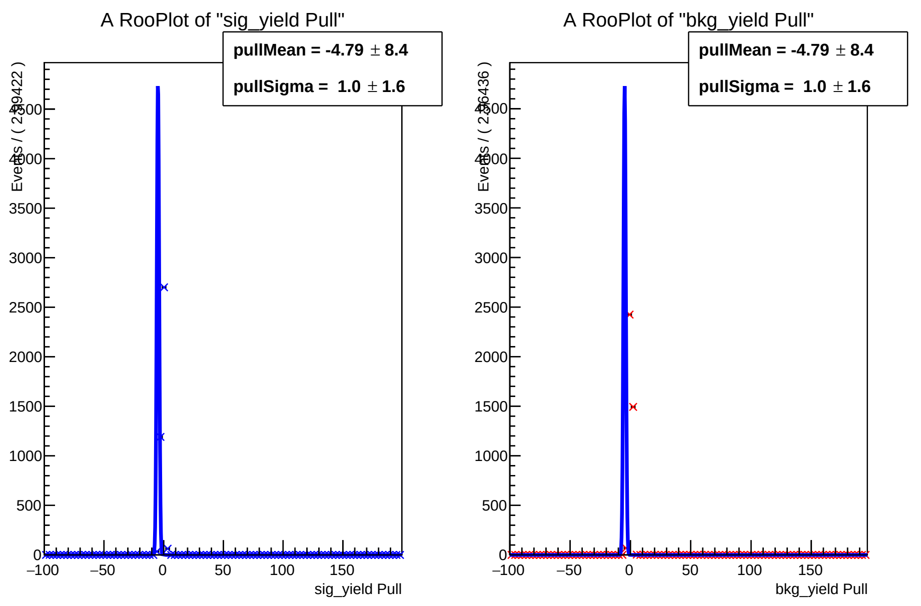
7.2.2 Upper limit scaling tests

The final tests in the first part of this thesis demonstrate the prediction of branching fraction upper limits and signal significance scaling with integrated luminosity. To test how the sensitivity scales a series of upper limit measurements can be made on the toyMC studies across a range of integrated luminosities. In particular there is great interest in whether the updated tag-side reconstruction algorithm is enough to combat the expected increase in backgrounds due to the higher luminosity of Belle II compared with Belle.

Figures 7.7 to 7.9 show the results of the calculated upper limits from fits (blue lines) to batches of toyMC experiments at integrated luminosities ranging from 1 ab^{-1} to 25 ab^{-1}, as well as the square of the yield fit significance (orange lines). This range is of course only covering half of the data volume expected from Belle II, however a refined repeat of this sensitivity study will be performed before measurement of real data when more Monte Carlo simulations and a better understanding of the backgrounds present in Belle II from results of calibration tests become available. Hence the ranges shown here are simply chosen to demonstrate that the upper limit and signal yield scale as expected with the data volume. The green lines show the current best limits, and the red the best limit from previous hadronic tag experiments. The shaded regions show the uncertainty due to statistical uncertainties, which at 1 ab^{-1} are $\mathcal{O}(1)$ relative to the branching fraction uncertainty. The resulting values of all scaling tests are included in tables F.1 to F.3 in Appendix F. From the large uncertainties it is not possible to determine whether the analysis performs better or worse than the previous hadronic tag. It does appear to be within the same order of magnitude at 1 ab^{-1} (recall the previous hadronic tag studies used less than 1 ab^{-1}), which indicates the overall procedure is working as expected.

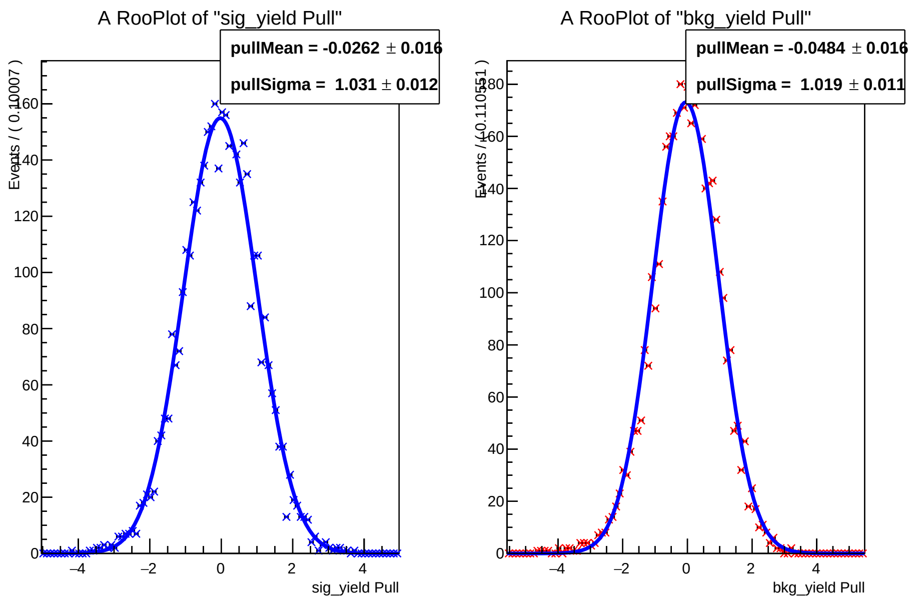


(a) K^+

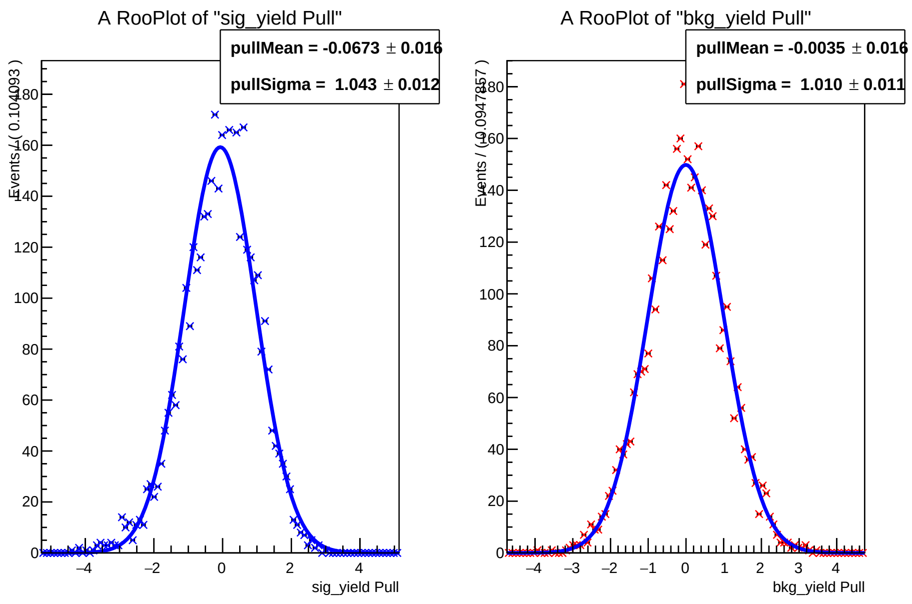


(b) K^0

Figure 7.3: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 1 ab^{-1} luminosity equivalent using histogram template models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits.

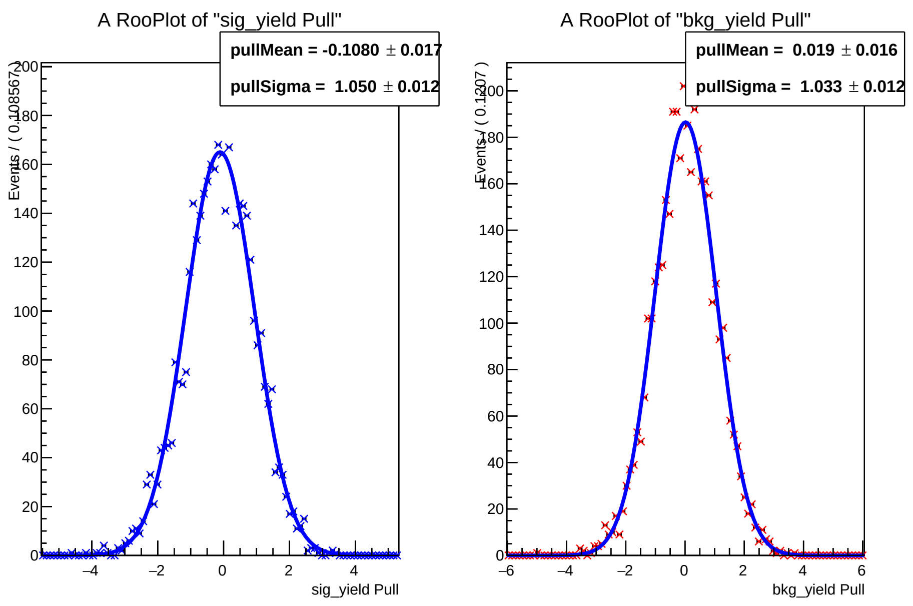


(a) K^{*+}

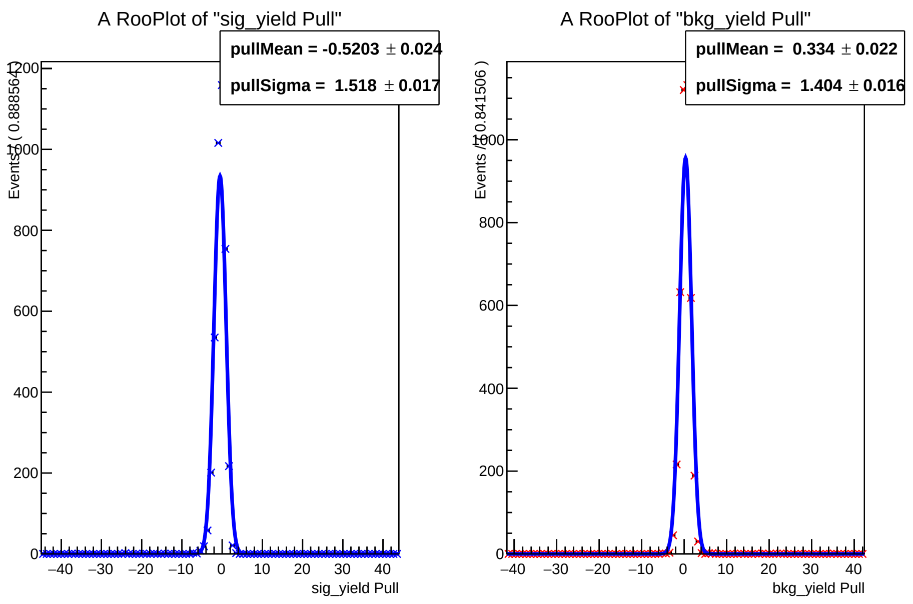


(b) K^{*0}

Figure 7.4: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 1 ab^{-1} luminosity equivalent using histogram template models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits.

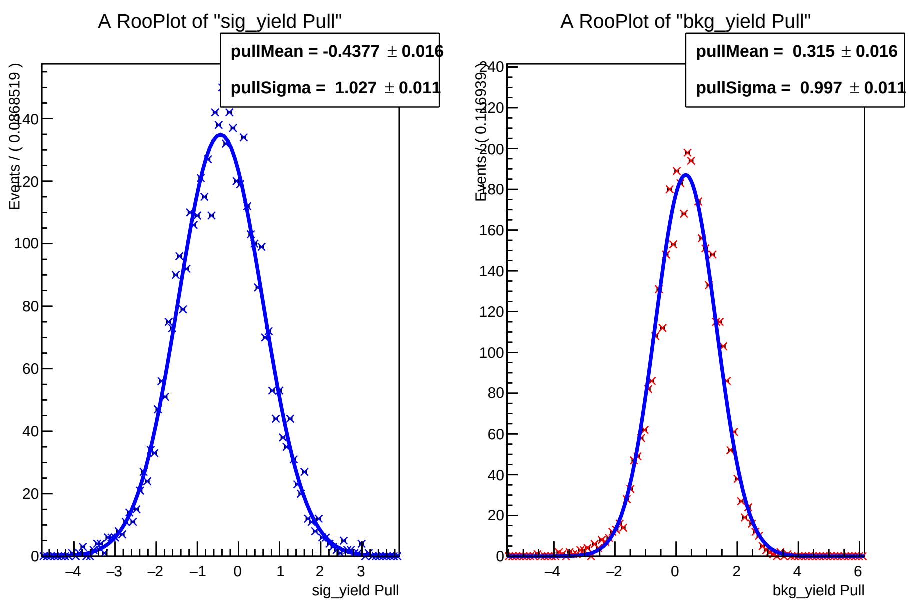


(a) K^+

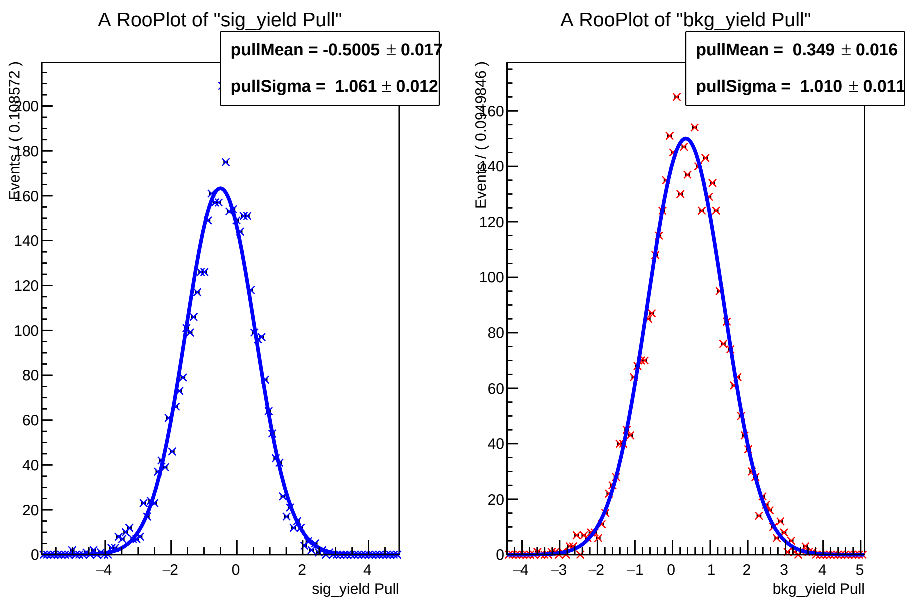


(b) K^0

Figure 7.5: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 1 ab^{-1} luminosity equivalent using parametric fit models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits. The results here are for the parametric fits only.

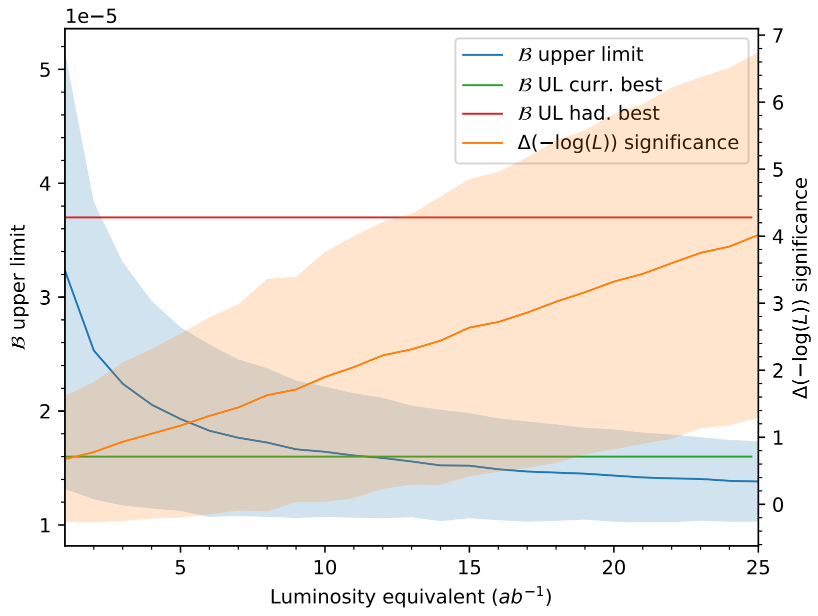


(a) K^{*+}

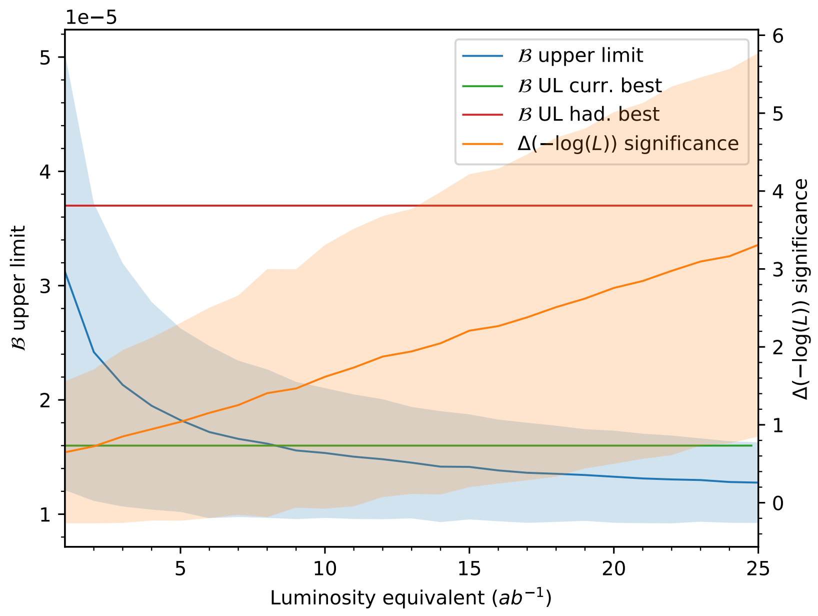


(b) K^{*0}

Figure 7.6: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 1 ab^{-1} luminosity equivalent using parametric fit models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits.

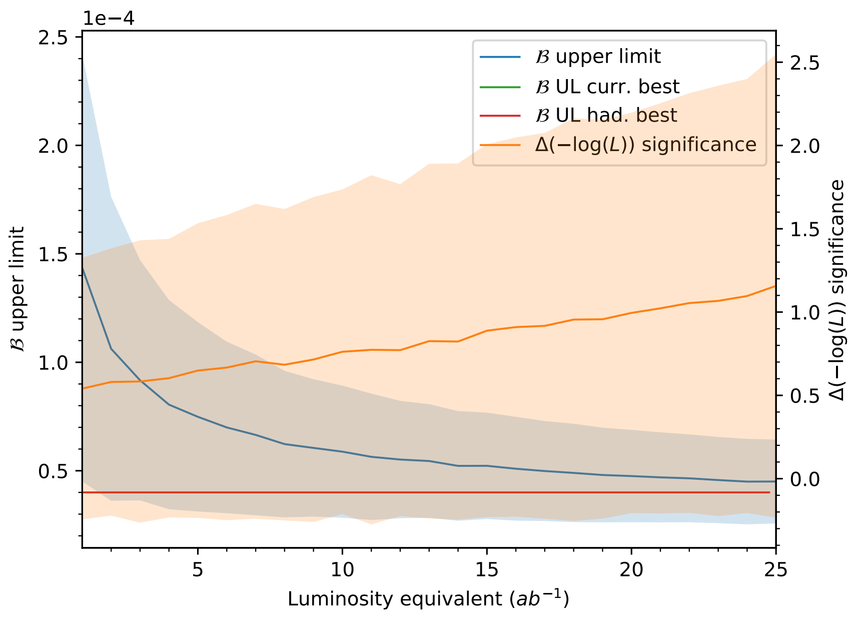


(a) Histogram template fits

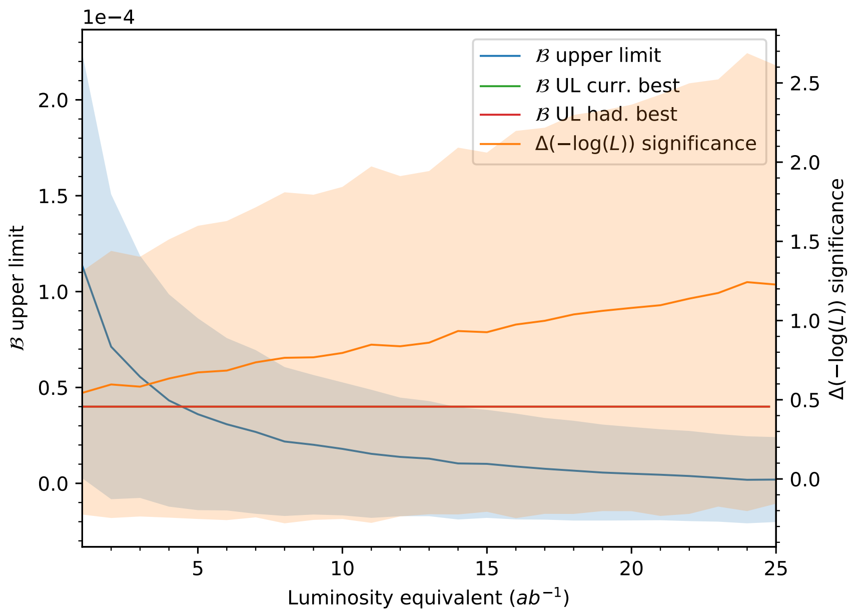


(b) Parametric (exponential + KDE) fits

Figure 7.7: Expected branching fraction upper limits at 90% C.L. obtained from toyMC studies performed at varying luminosity equivalents for the K^+ signal mode. The blue line shows the mean of the fitted upper limits from MC, red the best upper limit given by previous hadronic tag studies, green the current world best upper limit from either hadronic or semileptonic tagging studies, and orange the mean significance squared of the observed signal yield. Shadings indicate one standard deviation due to statistical uncertainties alone.

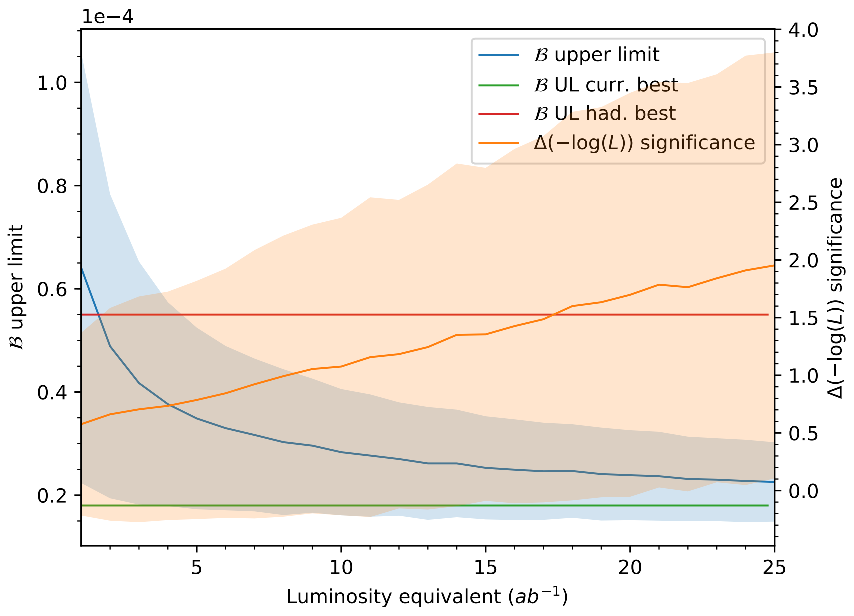


(a) Histogram template fits

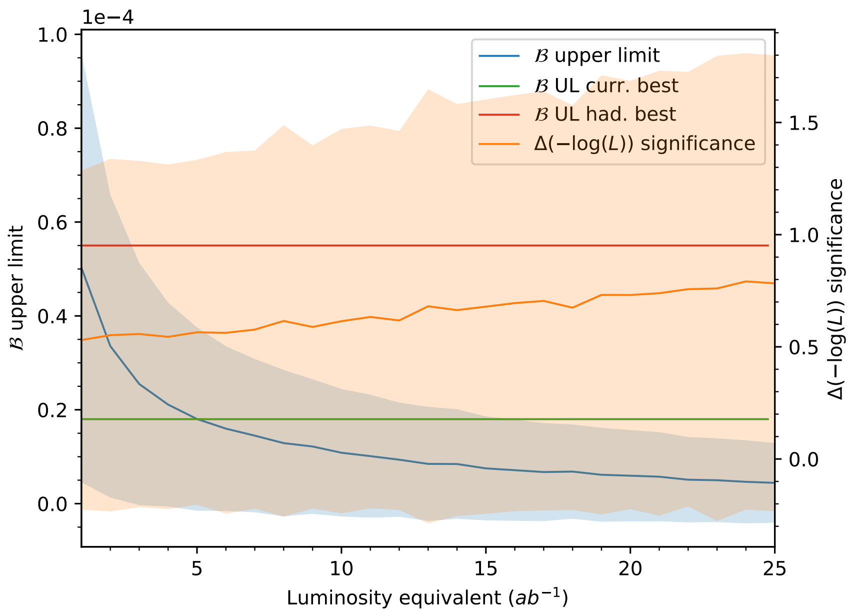


(b) Parametric (exponential + KDE) fits

Figure 7.8: Expected branching fraction upper limits at 90% C.L. obtained from toyMC studies performed at varying luminosity equivalents for the K^{*+} signal mode. The blue line shows the mean of the fitted upper limits from MC, red the best upper limit given by previous hadronic tag studies, green the current world best upper limit from either hadronic or semileptonic tagging studies, and orange the mean significance squared of the observed signal yield. Shadings indicate one standard deviation due to statistical uncertainties alone.



(a) Histogram template fits



(b) Parametric (exponential + KDE) fits

Figure 7.9: Expected branching fraction upper limits at 90% C.L. obtained from toyMC studies performed at varying luminosity equivalents for the K^{*0} signal mode. The blue line shows the mean of the fitted upper limits from MC, red the best upper limit given by previous hadronic tag studies, green the current world best upper limit from either hadronic or semileptonic tagging studies, and orange the mean significance squared of the observed signal yield. Shadings indicate one standard deviation due to statistical uncertainties alone.

Chapter 8

Summary

In this study I developed the procedure for measuring the sensitivity to the flavour-changing neutral current (FCNC) processes $B \to K^{(*)}\nu\bar{\nu}$. I utilised the reconstruction software Full Event Interpretation (FEI) to perform a hadronically tagged search within the new Belle II Software Analysis Framework. I demonstrated the procedure for extracting upper limit and observation significance sensitivities and made an estimation from currently available background Monte Carlo simulations.

FCNC processes have long been of interest due to their suppression at tree-level making them excellent probes of the Standard Model. The high level of suppression due to the Glashow-Iliopoulos-Maiani (GIM) mechanism means FCNCs in general are highly sensitive to new physics contributions. The final state neutrinos in $b \to q_d\nu\bar{\nu}$, $q_d = d, s$ makes this particular subset of FCNCs the theoretically cleanest, allowing direct measurement of the corresponding form factors. In $B \to K^{(*)}\nu\bar{\nu}$ processes in particular, the final state kaons allow for a direct comparison of Wilson coefficients of operators shared with the well measured $B \to K^{(*)}\ell^+\ell^-$.

I have added to the techniques employed in previous studies by optimising the selection procedure prior to the full event reconstruction, along with the use of the improved tagging algorithm FEI. I employed a specifically targeted rest-of-event filter aimed at handling the increased beam-related backgrounds expected at Belle II. This will require further fine-tuning as new information about the background levels at Belle II becomes available from calibration tests performed during phase 2 of operation.

I performed the final signal yield extraction via a fit to the neutral extra energy remaining in the electromagnetic calorimeter, E_{ECL}. For correctly reconstructed $B \to K^{(*)}\nu\bar{\nu}$ events this is sharply peaking at 0 GeV, and for backgrounds is distributed roughly evenly across higher energies. The signal region I investigated in this study was the range 0 GeV–1 GeV.

Due to the low statistical significance of the Monte Carlo simulation volume available at the time of this study, I was not able to ascertained a significant sensitivity estimate. The entire procedure, however, from reconstruction to selection optimisation and finally fitting and branching fraction calculations, has been made available to the Belle II collaboration with a focus on fast reproducibility. This will allow fast, updated estimates to be made as more Monte Carlo simulated data becomes available in future.

The result of this study leads directly into part II of this thesis. The need for significantly more Monte Carlo simulations, of which are currently very computationally expensive to produce, motivates the development of more intelligent simulation techniques. This is precisely what I investigate in the second half of this thesis, with the work performed within the context of the study of rare, fully reconstructed decay mode searches.

8.1 Outlook

The biggest improvement to this study will come from increasing the volume of background Monte Carlo simulations. To produce a sensitivity estimate of equivalent significance to the previous experiments, an integrated luminosity equivalent of at least $5\,\mathrm{ab}^{-1}$ should be used. For accurate estimates of the latter half of the Belle II lifetime, significantly higher simulation volumes will be required. This will be especially important as improvements in the full reconstruction algorithm can be reasonably expected which will reduce the background efficiency further.

The signal selection in this work used a simple cut-based approach. Even with the selection optimisation performed this is always limited in its discriminating power as it does not consider correlations between selection variables. Extending the selection to use a decision tree or neural network instead would provide additional background suppression and warrants investigation. This would require validation on a control sample containing a kinematically similar process to the signal. The control could be used to ensure the decision tree or network is not susceptible to differences between simulation and data.

A technique used in the BaBar 2013 analysis [25] to determine the background fit shapes in the signal region was to use a region in M_{bc} containing no signal events. Since M_{bc} is uncorrelated with E_{ECL} it allows a determination of the background shape from data directly, removing some of the dependence on Monte Carlo simulation volumes.

The Standard Model predictions used in this study are based on 2015 predictions [17]. The Flavio [19] package contains up to date Standard Model parameters which can be used to produce updated branching fraction predictions. Using this will allow more accurate Standard Model sensitivity estimates.

With the recently finished phase 2 calibration run of Belle II, updated information about the beam-related backgrounds is available and included in the latest commissioned Monte Carlo simulations. Therefore the rest of event filter should be updated to account for any changes. Additionally, investigations within the Belle II collaboration are being made into the use of decision trees for optimisation of a generic rest of event filter. This work should be monitored and implemented when ready.

As discussed in the introduction to Chapter 3, the process $B^+ \to \tau^+\nu_\tau(\tau^+ \to K^+\bar{\nu_\tau})$ contains the same final state missing energy signature as some of the $B \to K^{(*)}\nu\bar{\nu}$ process. The τ^- decays into a single charged K^- or to a charged-neutral pair, e.g. $K^-\pi^0$, which may produce a false K^{*-} candidate, would contribute with a signal-like structure in E_{ECL}. These potential background channels, however, have a branching fraction of order $\mathcal{O}(10^{-3})$. Combining this with the current experimental world average for $\mathcal{B}(B^+ \to \tau^+\nu_\tau) = (1.09\pm0.24)\times10^{-4}$ [9], and the maximal contribution is at the $\mathcal{O}(10^{-7})$ level. This is before accounting for the reconstruction efficiencies, which for the previous $B^+ \to \tau^+\nu_\tau$ hadronic tag study was at the $\mathcal{O}(10^{-4})$ level. Therefore this appears to be a negligible background contribution until significant improvements in the current tagging and background suppression techniques are made. Furthermore, if the sub-decay involving $\tau^+ \to K^+\bar{\nu_\tau}$ did eventuate as a non-negligible background, it could be kinematically vetoed as it comprises of multiple two-body decays (assuming this does not interfere with q^2 regions of interest in the analysis). As the current experimental results show agreement with standard model predictions, the full set of $B^+ \to \tau^-\nu_\tau$ background contributions to the fitted observable E_{ECL} can also be modelled with reliable accuracy from Monte Carlo simulation and included in the fit. Since the τ^+ decays relevant to $B \to h\nu\bar{\nu}$ signal channel studies are dominated by processes involving one or more pions, this background is one that will be of greater significance to channels involving $h = \pi, \rho$ for example.

Part II

Selective background Monte Carlo simulation at Belle II

Chapter 9

Introduction

The context of the problem that I address in this study lies in the drive to push the boundaries of our understanding of the fundamental properties of nature. The field of high energy physics (HEP) has two key frontiers which allow it to probe deeper into the current Standard Model of particle physics: the energy frontier, which allows creation of new, heavy particles directly via a brute force approach, and the precision frontier, which relies on the detailed knowledge of a clean collision environment to observe the effects of the contributions of intermediate particles to physics processes. Figure 9.1 shows the progression of HEP experiments through time as they have pushed the boundaries of both. The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is at the forefront of the energy frontier, and SuperKEKB where the Belle II experiment is hosted will lead the high luminosity precision frontier.

The goal of Belle II is to obtain an integrated luminosity of $50\,\mathrm{ab}^{-1}$ worth of data, or roughly fifty times that of the Belle experiment. Performing measurements on this data requires modelling of the experiment as best as possible to first prepare each analysis, in this case in the form of Monte Carlo simulations. This allows fine tuning physics analyses to be able to filter the physics process being searched for, the signal, from everything else, the background. Since a large portion of the measurements will be focused on so-called rare processes, processes with a branching fraction of 10^{-6} and lower, a strong statistical knowledge of the backgrounds is required to accurately distinguish the signal from the remaining background. The approach taken during the Belle experiment was to simulate ten times the volume of measured data, i.e an integrated luminosity of $10 \times 0.711\,\mathrm{ab}^{-1}$. To use the same approach at Belle II would then require simulations of an integrated luminosity of $500\,\mathrm{ab}^{-1}$. Chapter 10 describes in detail the resource requirements of this level of data, however here I will simply state that the time requirements for simulating ten times the expected Belle II data make it infeasible using current or near-future resources available. Not only that, but for any given analysis, or even class of analyses, a large portion of the data is trivially thrown away as irrelevant to the particular study. Therefore more intelligent methods of simulation are required to produce

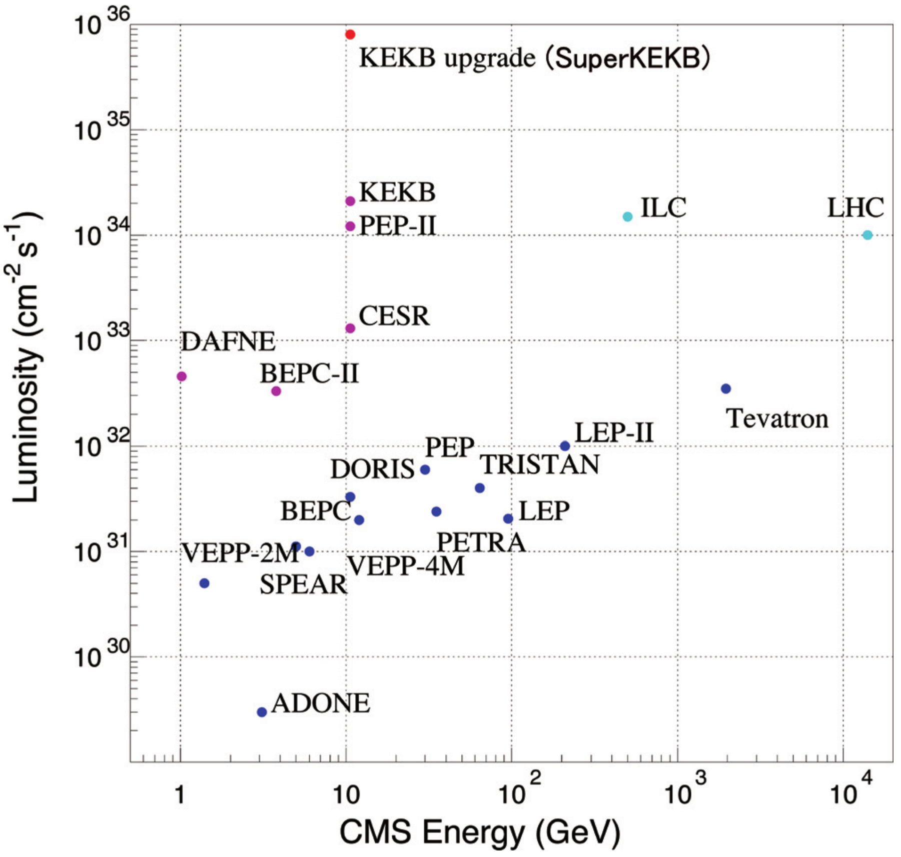


Figure 9.1: Comparison of past, present, and proposed high energy physics experiments' centre of mass (CMS) energies and luminosities [51]. Currently the experiments at SuperKEKB and LHC are at the forefront of the precision and energy frontiers.

large volumes of relevant data.

There are two immediately obvious ways to solve this problem: only simulate en masse events relevant to the given study, or make the simulation of events fast enough that it can be performed on-the-fly, thus removing entirely the existing bottleneck. While the second solution is clearly the ideal for a multitude of reasons, it is also the most difficult given the sheer complexity of simulating every individual component of the Belle II detector. In this study I attempt instead to tackle the first solution with the use of neural networks. The end goal is to develop a procedure for identifying useful events for any given study in a manner that integrates into the existing Belle II software and Monte Carlo production paradigm.

Chapter 10

Monte Carlo simulation

As was described in part I of this thesis, to prepare an analysis for experimental data, signal extraction and fitting is prepared on Monte Carlo simulations first. These simulations are intended to represent the expected behaviour of the Belle II detector as best as possible. In this chapter I will describe the Monte Carlo simulation procedure, as well as outline the operation of the Belle II Analysis Software Framework (basf2) [1, 40]. I will finish with a brief discussion of the grid-based mass production performed within the Belle II collaboration as it relates to this study.

10.1 Monte Carlo production

Figure 10.1 shows the outline of the full analysis procedure (repeated here for convenience). Within Belle II, *Monte Carlo simulation* refers to the output of the first three stages of the full procedure shown (excluding the real data), which are the initial Monte Carlo generation (MC), the detector simulation (Det. Sim), and the detected particle reconstructions (Reco.). The skim (section 6.2.1, analyse (sections 6.2.2 to 6.2.5), and fit (Chapter 7) were described for the particular study in part I of this thesis. The data represents the output of the actual Belle II experiment, expected to become available within the first half of this year (2019).

The steps in figure 12.2, from left to right, are as follows:

MC This is the first step in the simulation process and involves the generation of the initial collision event (labelled Event Generation in table 10.1) using the EvtGen package [52], to simulate hadron decays, in conjunction with Pythia [53], used to simulate quark hadronisation. This simply simulates the electron positron collision and subsequent decays to stable final state particles (FSPs) in empty space. The Monte Carlo particles generated here are referred to as *primary particles* and are flagged by a status bit(see Glossary on page 146 for details). This allows particles simulated in this stage to be distinguished from additional particles simulated later. Every collision event simulated is assigned a unique event number.

Det. Sim. This step takes the simulated decay from the MC stage and simulates the particle interactions with the Belle II detector. To do this the Geant4 toolkit [54] is used, calibrated to represent the Belle II detector. The output format of this stage is identical to that of the real experiment output (detector hits, track candidates, etc.), except each event also contains information about the decay simulated in the MC stage. At this point in the simulation the beam backgrounds are added to the initially simulated decay, as described in section 4.1.1. The strategy used to add backgrounds to the simulations used in this study is known as *background overlay* [35]. Using random events the background overlay method creates background event samples which are then mixed in with the simulated physics event. The mixing is performed during the simulation of the digitisation of detector hits, before reconstruction of particle candidates is performed. The other available method is *background mixing* of simulated backgrounds and is described in detail in [35] but is not included in this study. The accurate understanding of background sources and levels is expected to be the biggest cause of simulation - data differences at Belle II. The large increase in instantaneous luminosity is expected to produce significantly higher background levels than those at Belle, and indeed initial measurements [33] indicate levels potentially orders of magnitude higher than those predicted in simulation. This is one of the crucial systematic contributions to understand and counteract in order to prepare analyses for early data taking. Throughout this study the term *nominal background* is used to refer to the total background levels expected at Belle II at the time of simulation.

Data This is the real life running of the experiment and is what the MC and Det. Sim. stages attempt to simulate. No data is used or quoted in this study as none is available at the time of writing.

Reco This step performs the reconstruction of detector information into particle candidates. The output is a set of readily accessible potential final state particles with properties that can then be used to identify each particle's type, e.g. distinguish between kaons and pions. This output is typically what is saved in MDST files, with all previous stages being run consecutively. MDST is the name of the storage format used within Belle II to efficiently store collision event information of varying sizes with fast access. For the purposes of this study it is simply used to identify files containing reconstructed simulation event data. The reconstruction procedure is provided as-is within basf2 and developed within the collaboration.

Skim The skimming of data involves the selection of events which are of easily identifiable interest to particular sub-groups within the Belle II collaboration. For example those studying charm quark physics are all interested in events containing D mesons, therefore

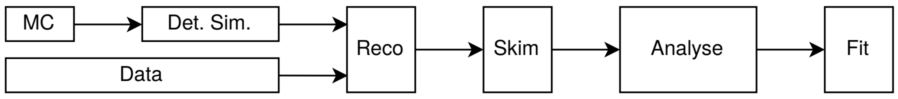


Figure 10.1: Data flow within a Monte Carlo based experiment. MC represents the event simulation performed by the EvtGen software. Det. Sim. is the Geant4 detector simulation. Reco is the reconstruction of detector hits into particle candidates. Skim is the collaboration wide data skimming and basic decay reconstruction. Analyse and Fit are the study specific analyses performed to measure observables within the data.

a charm skim will perform a simple reconstruction of many D meson decay channels with very loose selection requirements and keep only events involving potential D meson candidates. The study in this thesis is performed on the Full Event Interpretation [42] skims which are described in section 6.2.1.

Analyse/Fit The analysis and fitting stages are specific to the particular process being studied, and are not relevant for this study. I only note here that any modifications to the preceding simulation steps must be cross-checked for potential biases introduced into the final fitted observables. The risk is inducing a deviation from the correct modelling of real experimental data. Section 12.5 of this study investigates mitigating such biases.

The corresponding execution times of each step in the simulation chain shown in figure 10.1 are shown in table 10.1 [55]. The times shown are a standardised measure of seconds of simulation required per collision. Comparing these times with volume of Monte Carlo simulated events shown in table 6.1 ($\mathcal{O}(10^9)$ events), and recalling that this only represents 2% of the number of events expected at Belle II, it's apparent that this is an unfeasible task. Naively assuming 1 HEPSPEC06min/event, this corresponds to almost 100.000 years worth of processing time just to reproduce the volume of events expected at Belle II[1]. This point gets at the key motivation of this study; until the simulation procedure is able to be performed at a reasonable rate, the only option is to only simulate the events relevant to a particular study.

10.2 Belle II Analysis Software Framework

The Belle II Analysis Software Framework (basf2) [1] is the core framework used to manage each stage in the data flow shown in figure 10.1. The framework has two main functions: to provide a simple, consistent interface for users to create data via the chaining of modules into a single processing path, and to manage the exchange of data between modules via a common data store. Figure 10.2 shows an example of the basf2 processing flow can be conceptualised.

[1] Even with a high level of parallelisation, as is utilised in current collaboration commissioned Monte Carlo production, the rate of simulation is roughly six months per inverse attobarn.

Table 10.1: Belle II Analysis Software simulation requirements estimates given as benchmarked HEP-SPEC06 seconds per simulated event [55].

Stage	HEP-SPEC06 s/event
Framework overhead	0.79
Event generation	0.11
Detector simulation	47.23
Event reconstruction	26.77

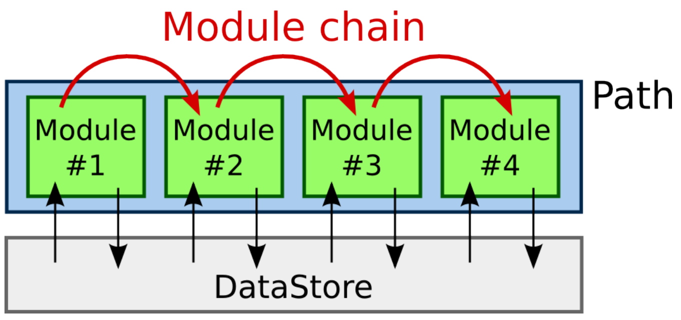


Figure 10.2: Outline of a simple execution path constructed of four sequential modules within basf2, with the the underlying DataStore providing a consistent interface to all data [40].

An example of the modules used here could be the initial Monte Carlo simulation steps: event generation, detector simulation, reconstruction, and writing to file. Any solutions that aim to improve the simulation efficiency must also be able to integrate natively into the current paradigm.

The basf2 software as a whole is logically separated into three parts: the framework itself, which contains Belle II specific code, the externals, which contains additional third-party dependencies, and the tools, containing the software installation and initialisation scripts. The entire software is made available via the CERN Virtual Machine File System [56]. In this study I have attempted to utilise libraries already available within the externals where possible (Pandas [57], NumPy [58], etc.), and where unavailable have had them added. Execution of so-called *steering files*, written in Python, are constructed by users and dictate the sequence of execution of basf2 modules. The modules themselves are in general written in C++ and compiled into shared libraries for fast, dynamic loading into basf2. The framework, however, also supports modules written in Python. While this does result in a small increase in execution time, as the modules are not pre-compiled for optimisation, it does allow the integration of current machine learning libraries, all of which are developed almost exclusively for Python[2].

[2]Or other common high-level languages such as R, Java, Matlab, etc., though Python is currently the most popular by a significant margin.

10.3 Grid-based production

To produce the large volume of Monte Carlo (MC) simulations required at Belle II, the collaboration currently commissions so-called *MC campaigns* [59]. These involve the large scale, parallel production of both generic and signal Monte Carlo events on Belle II computing resources [60] on grid-computing sites around the world. Therefore, any solutions developed to improve the simulations must be compatible with the Belle II computing requirements.

The basic workflow of jobs (simulations, analyses, etc.) performed on the grid resources is as follows: a user submits a job containing a steering file and any other supplementary local files, the user also specifies which datasets if any they would like to include as input (the datasets are collections of large files, e.g. simulation outputs, stored on the grid resources), the job is received by the central Belle II computing servers and distributed to so-called *worker nodes* for execution, once completed the output of the jobs can either be downloaded directly by the user from some temporary storage (for small files) or stored as a dataset on the grid. There are two important restrictions involved in the entire process, that is that the supplementary local files and the small output files that can be directly downloaded must both individually total less than ten megabytes. This restriction on the input file size is especially relevant in this study as it places an upper limit on the size of any neural network models developed.

Chapter 11

Machine learning

Many resources exist on the topic of machine learning, for example [61] provides an excellent free introduction to the basic mathematical concepts required and modern neural network architectures used in practical applications today. In this chapter I introduce the concepts in machine learning relevant to this study, along with the key architectures implemented in the solutions.

Most modern machine learning fits roughly into one of three categories: supervised, unsupervised, or reinforcement learning. Supervised machine learning involves the teaching of a system by presenting it with examples and telling it the answer (also called the *label*). The system attempts to guess each answer, is told the real answer, and then makes adjustments that help it get a more accurate answer next time. Unsupervised learning is the same but without the system being told the answer. The system must draw inferences from the data and learn to recognise patterns on its own. Reinforcement learning is the teaching of a system to interact with an external environment. The system learns the best action to take given the current state of the environment in order to optimise some predefined reward. A good example is teaching a neural network to beat a video game, where the reward is achieving the highest score. The study in this thesis deals exclusively with supervised learning. I present examples of Monte Carlo simulated physics events to the networks, labelled as whether that event was useful for a given physics analysis. I then train the networks to accurately predict whether Monte Carlo events will be useful in analysis.

Neural networks are based on a model of the neurons within the brain. They attempt to create a general statistical estimator that can be used to give high accuracy results to problems too difficult to be solved analytically. Using large combinations of simple functions neural networks are able to learn an approximation of any given set of data [62]. A well trained network will be able to generalise this approximation to new data and accurately perform the specific task it was trained for.

For this study I use the Tensorflow [63] library as the backend to construct the neural

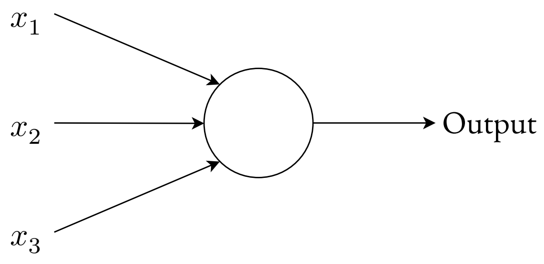


Figure 11.1: Single neuron in a neural network, x_i are the inputs.

networks and perform their training. I use the Keras [64] library as a high-level API running on top of Tensorflow to allow for fast, simple construction of network architectures. Therefore for the remainder of this document I will refer to the components of any networks by their names as they are implemented in Keras where necessary, with the relevant components being defined in the following sub-sections of this chapter.

A neuron is the simplest building block of a neural network. Figure 11.1 shows an example of a single neuron with three inputs and a single output. In its simplest form the neuron takes the multiple inputs x and returns an output

$$z = \sum_i w_i x_i + b, \tag{11.1}$$

where w is the *weight* applied to each input and b is the *bias* added to the total. The ideal combination of weight and bias values for each neuron is ultimately what the training process attempts to learn. Each step in the training of the network adjusts the values of the weights and biases according to how well the network approximates the training data[1]. The learned parameters in the complete network are known in Keras as the *trainable parameters*. A more complex neural network, involving more neurons, allows more combinations of equation (11.1) and hence more complex functions to approximate the given data. Too many trainable parameters, however, risks allowing the network to become too specialised towards the training data provided, potentially learning statistical fluctuations contained in the data and losing the ability to generalise to new, unseen data. More trainable parameters also translates to an increased training time.

The output shown in equation (11.1) is only a linear output, however in general what the neural network is attempting to model is not linear. Therefore non-linearities can be introduced by applying an *activation function* to the output of each neuron. The choice of activation functions used in each layer of a network is a hyperparameter to tune. Table 11.1 shows the definition of some common activations as they would be applied to the output of equation (11.1). Figure 11.2 shows the corresponding plots of each. For the layer type

[1] For more advance neuron implementation the specific parameters to be trained changes, however the general process remains the same.

Table 11.1: Common activation function definitions applied to the output in equation (11.1) of each neuron in a layer. Figure 11.2 shows the corresponding shape of each.

Name	Definition
Sigmoid	$f(z) = \frac{1}{1+e^{-z}}$
Tanh	$f(z) = \tanh(z)$
ReLU	$f(z) = \max(0, z)$
LeakyReLU	$f(z) = \begin{cases} z, & \text{if } z > 0 \\ az, & \text{otherwise} \end{cases}$

implemented in this study the activation functions that provide a good general performance for each are known. When designing network architectures in section 12.4 these are what I begin with. For the final network layer, the network output, a classification prediction must be made for this study. In the case of binary classification (true/false) this should be a number between 0 and 1 representing a probability as judged by the network. For this a sigmoid activation is appropriate as it squashes the output to this range. If the classifier is extended to multiple classes the softmax function should be used instead, defined as

$$f(z_i) = \frac{e^{z_i}}{\sum_i^K e^{z_i}}, \tag{11.2}$$

where i indicates the class and K the total number of classes. The sum in the denominator ensures that the sum of outputs across all classes is 1. Again the output of every individual class lies between 0 and 1.

The simplified outline of supervised training of a neural network is as follows:

1. The network is given a sample of data.
2. The network then makes a prediction of the label for that sample using its current weights and biases.
3. The accuracy of the prediction is quantified using a *loss function*.
4. The weights and biases are all updated according to the loss.

This process is repeated until the loss function output reaches a satisfactory level or no longer improves.

The training procedure should minimise the average loss across data samples as much as possible to produce the best general accuracy. For the task of classifying data into discrete

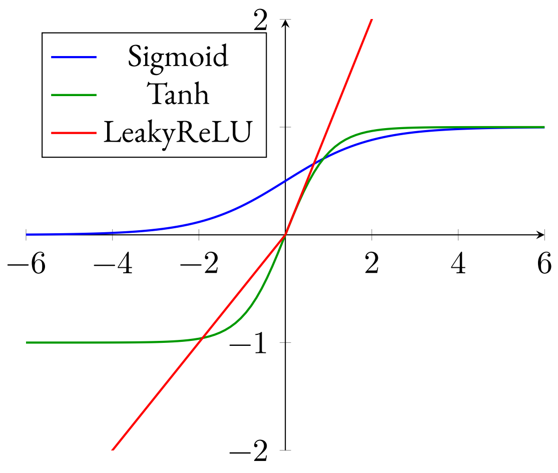


Figure 11.2: Example of activation functions used in this study, defined in table 11.1. ReLU (not shown) is identical to LeakyReLU except all negative values are zeroed (lie on the x-axis).

categories the cross-entropy loss function is appropriate. The generalised cross-entropy for multi-class classification (M classes) is

$$\mathrm{L} = -\sum_{c=1}^{M} y_{o,c} \ln p_{o,c}, \tag{11.3}$$

where $y_{o,c}$ is a binary indicator of whether the class label being checked, c, is the correct classification for the observation o, and $p_{o,c}$ is the network predicted probability that observation o is of class c. When performing binary classification, as is the case in this study, the binary cross-entropy can be written simply as

$$\mathrm{L} = -(y \ln p + (1-y) \ln (1-p)), \tag{11.4}$$

where y is the true label and p is the predicted output of the network. For the remainder of this study I use binary cross-entropy exclusively as the loss function. I use an additional metric, *accuracy*, throughout this study to provide a more intuitive measure of network performance. The accuracy of binary outputs in Keras is defined as

$$\text{Accuracy} = \frac{1}{N} \sum^{N} \left(y - \lfloor p \rceil \right) , \tag{11.5}$$

which is the mean of the difference between the true label, y, and the rounded predicted output, p, across the input batch of N events.

The details of how individual weights and biases are updated each training step is described in detail in [61]. Here it suffices to know that the gradient of the loss function is calculated with respect to the individual weights and biases, and used to adjust them in the direction of

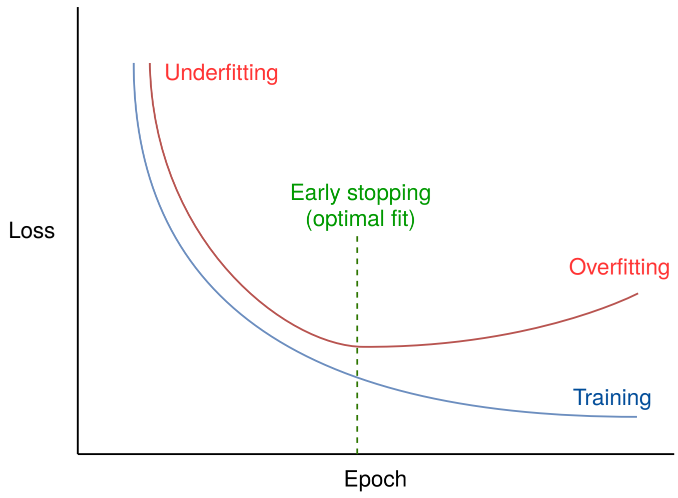


Figure 11.3: Example of over, under, and optimal fit stages during training, identified as the similarity between network performance on training (blue) and validation (red) data, as well as absolute loss. The green dashed line indicates the ideal stopping time when utilising early stopping.

the loss gradient that minimises the loss. The calculation of the gradient is performed by a process known as backpropagation. Using the gradient, changes to the weights and biases are made backwards through the network layer by layer.

The specific optimisation process chosen, or *optimiser*, is a hyperparameter of the network. Throughout this study the Adam optimiser [65], with the AMSGrad [66] variant enabled, performs sufficiently well and is used in the trainings described in section 12.4. Other optimisers such as Adadelta [67] and Adagrad [68] are also investigated but ultimately found to show no additional benefit for the task in this study. How aggressively the optimiser updates the network's trainable parameters at each training step is an additional hyperparameters called the *learning rate*. If the learning rate is too high the network may never find the optimal parameters, forever overstepping the loss minima in its adjustments. If the learning rate is too low the network may take an impractically long time to optimise.

To verify that the trained network is able to generalise to new data and not overfit the training sample, I set aside a subset of the labelled data for validation. After a certain number of training iterations have been performed I pass the network a sample of the validation data. The network makes a prediction and calculates the loss, however the weights and biases are not updated. I repeat this process is fixed intervals during training to prevent overfitting to the training data. Figure 11.3 shows and example of how a typical training may proceed, ending in overfitting, with the ideal time to stop training shown by the green dashed line.

When designing a neural network there are many initial parameters that must be fine tuned to produce optimal results; for example the number of neurons to use, or the optimisation

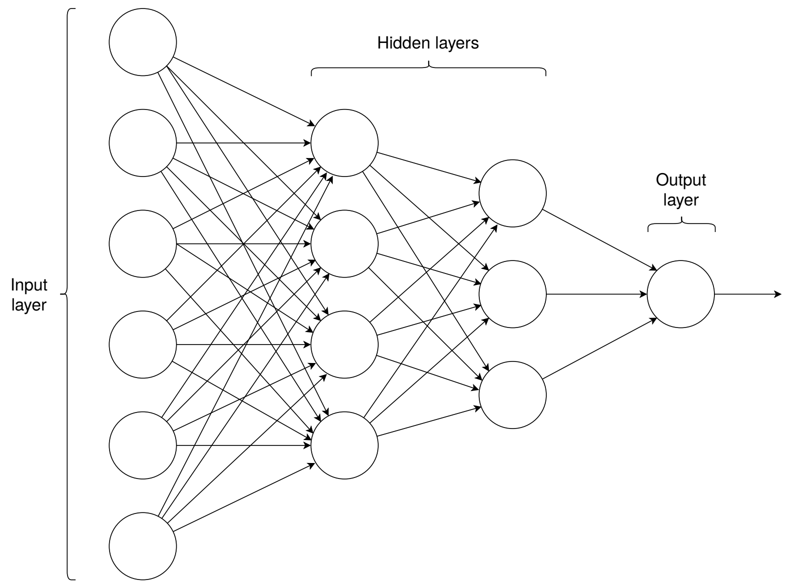


Figure 11.4: Basic structure of multi-layer perceptron, information flows directly from input to output. All nodes of previous layers are connected to all nodes of the following layer [69].

process the training should use. These parameters are collectively referred to as *hyperparameters*, not to be confused with the aforementioned trainable parameters, and in general are fixed before the start of training. The search for the optimal network and training configuration is called hyperparameter optimisation.

The simplest implementation of a neural network is a multi-layer perceptron (MLP), sometimes referred to as feed-forward or fully connected networks. Figure 11.4 shows a simplified outline of the structure of an MLP, where each circle is a neuron (also referred to as a node), and each column of neurons is a layer. Information flows only forwards from inputs on the left to the output on the right. All nodes of each layer has a connection to all nodes of the following layer. In Keras each individual fully connected layer is referred to as a *Dense* layer. The choice of how many nodes are in each Dense layer is a hyperparameter.

11.1 Convolutional neural networks

Convolutional neural networks (CNNs) [70] expand on the multi-layer perceptron to introduce spatial independence into the network's capability. This makes them ideal for the processing of images, in which the location of an object within an image is typically unrelated to what the object is itself.

Convolutional layers are defined as those implementing a convolution as opposed to the general matrix multiplication in multi-layer perceptrons. From [61], the discretised (discrete as each input is an individual pixel or time step) general form of a convolution is

$$s(t) = (x * w)(t) = \sum_{a=-\inf}^{\inf} x(a)w(t-a), \tag{11.6}$$

where $s(t)$ is the output at time step (or pixel location) t, known as the *feature map*, x is the input, e.g. pixels of an image, and w is the probability density function being convolved with the input, referred to as the *kernel*. The kernel in a convolutional network contains the weights to be learned during training. For two and higher dimensional inputs the kernel and feature map also increase dimension accordingly with each new dimension also being summed across. In practice the kernel is only non-zero for a small number of coordinates, with the number of non-zero coordinates known as the *kernel size*. Figure 11.5 (from [61]) shows an example of a single 2×2 kernel operating on a 4×3 input image. The four trainable parameters (w, x, y, z) are learnt during training. The output, a 3×2 image, is the feature map. Each convolutional layer typically contains multiple output feature maps, with the number of feature maps known as the number of *features* or *filters*). Each output feature map trains its own individual set of kernels, with a separate kernel for each input feature map within the set such that all input feature maps are inputs to each individual output feature map. In the example shown the *stride* is $(1, 1)$, meaning the kernel moves one pixel in either direction at each evaluation. Increasing the stride reduces the output filter size. The kernel size, number of features, and stride are all hyperparameters of convolutional network layers.

Unlike the fully connected layers in a multi-layer perceptron, a convolutional layer does not need to learn the weights and biases of every individual input pixel (which in the MLP correspond to neurons). Instead only the weights of each kernel must be learnt, meaning that the number of trainable parameters does not grow with input image size. This is a feature of convolutional networks known as *parameter sharing*. Coupling this with the parallelisation of the kernel matrix multiplication, something modern GPUs are highly optimised for, convolutional networks are able to train significantly faster than other network types. This is an important technical consideration in tasks involving very large data sets, as physics problems typically do.

Convolutional networks have the ability to handle inputs of arbitrary size, as the kernel moving across the inputs removes the spatial dependence of input pixels and focuses instead on local spatial dependence only. This is, however, only possible for fully convolutional networks – networks containing only convolutional layers. In practice the inputs are padded or truncated (covered in section 11.3) to a predetermined size to allow the building of Dense layers that typically comprise the final layers of a convolutional network. Fully convolutional networks are not covered in this study but a suggestion of why they may be useful in future iterations of this work is discussed in section 13.1.

Within Keras, the implementation of a one-dimensional convolutional layer is known as *conv1D*, and a two-dimensional layer as *conv2D*. In descriptions of network implementations

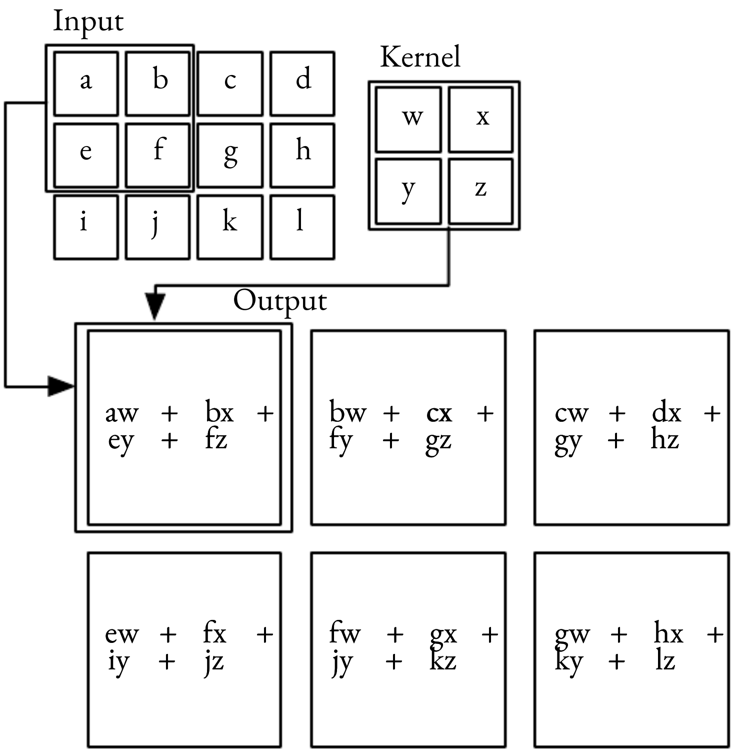


Figure 11.5: From [61], an example of a convolutional layer operating on a 4×3 input image (a – l), with stride $(1, 1)$. The kernel, size 2×2, is convolved with each 2×2 square of input pixels to produce a 3×2 output filter. The kernel values (w – z) are learned during training.

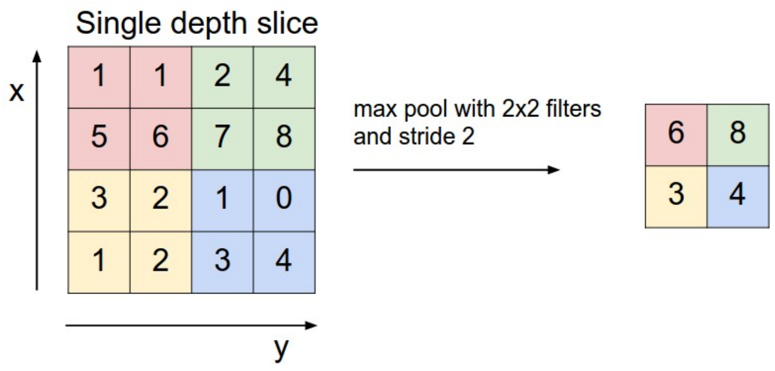


Figure 11.6: Max pooling applied to a single feature map [71]. The max pool filter has a size of 2×2 and a stride of 2, hence it is only applied to each corner of the input.

throughout section 12.4 I use the notation *Conv1D(k,f)* to refer to a single one-dimensional convolutional layer with kernel size k and f feature maps.

11.1.1 Pooling

Pooling is a function unique to convolutional networks which reduces the size of the outputs. It applies a common function independently to every feature map to reduce the spatial size. Figure 11.6 shows an example of *max pooling* applied to a single feature map. Looking again at figure 11.5, max pooling replaces the kernel values with a single max() operation, again with size and stride as hyperparameters. Unlike the typical convolutional operation, however, pooling does not apply simultaneously to each input feature map, only to a single feature map at a time. Therefore pooling does not modify the number of feature maps, only their size. In addition to max pooling, there is also *average pooling* which simply averages across the kernel window. Throughout this part of the thesis I refer to these operations as *MaxPool(n)* and *AvgPool(n)*, where n indicates the pooling kernel size. Stride can be assumed to equal the pooling kernel size unless explicitly stated otherwise. When not given, $n = 2$.

In the final layers of a convolutional network a fully connected network is typically used to consolidate the outputs into the output nodes necessary for classification. To do so requires a dimensionality reduction down from the final feature maps to a one-dimensional array of neurons. Two common methods exist to achieve this: flattening and global pooling. A simple example of flatten is shown in figure 11.7, where all inputs from the previous layer are stacked into a single dimension. Global pooling is shown in the same figure and is the extreme case of the pooling operations described above, where the pooling kernel size is set to the feature size. For example global max pooling will keep only the maximum value of each input feature map. In describing network architectures I refer to the global pooling operations as *GAvgPool* and *GMaxPool* for global average and max pooling.

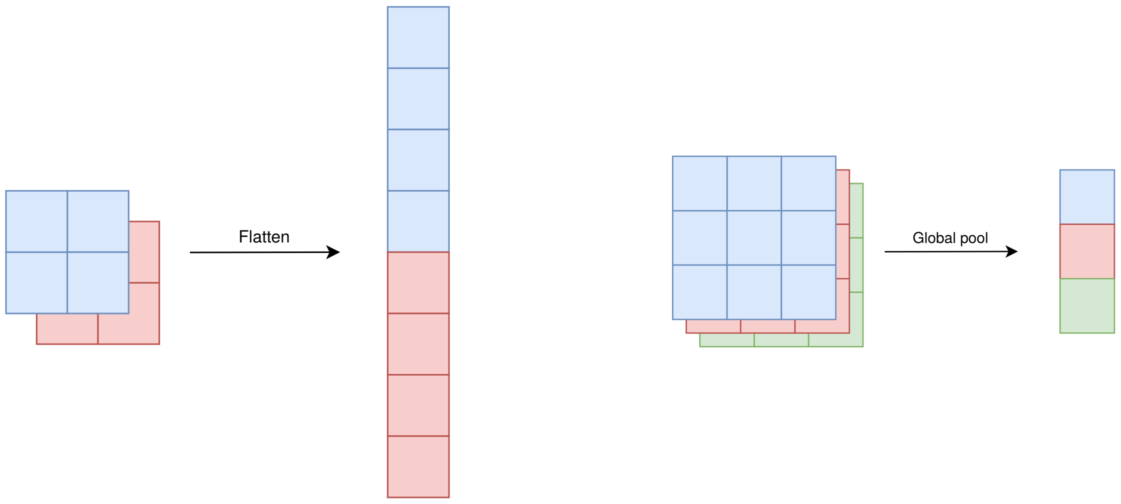


Figure 11.7: Flatten (left) and global pooling (right) are used as dimensionality reductions to transition between convolutional network layers and fully connected MLP layers.

A common convolutional layer grouping scheme is multiple Conv1D layers followed by a single pooling layer, described below. For this configuration I use the shorthand *Node(n, k, f, p)*, where n indicates the number of successive convolutional layers with kernel sizes k and feature maps f, and p indicates the pooling function applied, either *max* or *avg*.

11.1.2 ResNet and ResNeXt

In deeper implementations of neural networks, models with many sequential layers, a degradation problem begins to occur. It becomes difficult for the training to calculate the effect of updates to earlier network layers, negatively impacting the overall performance of the network. To combat this, residual networks (ResNet) [72] were proposed which introduce shortcuts throughout the network. The left image of figure 11.8 shows an example of the shortcut implementation, where the input to the first convolutional layer of the block is added to the output of the final layer, and the intermediate layers preserve the initial input dimensions (256). The notation used in this image for convolutional layers is (input dims, kernel size, output filters). This increases the impact adjustments to earlier network layers has on the final output loss.

A variant of ResNet, known as ResNeXt [73], builds on ResNet to aggregate many connected layers in a single block. The right diagram in figure 11.8 shows a set of 32 aggregated layer sequences, again with a shortcut connection and the total input dimensions preserved. The number of aggregated sequences, 32 in this case, is known as the *cardinality* of the network.

In describing architectures in section 12.4, I use the shorthand for ResNet blocks of *ResNetNode(n,k,f,p)*, where n refers to the number of intermediate convolutional layers with

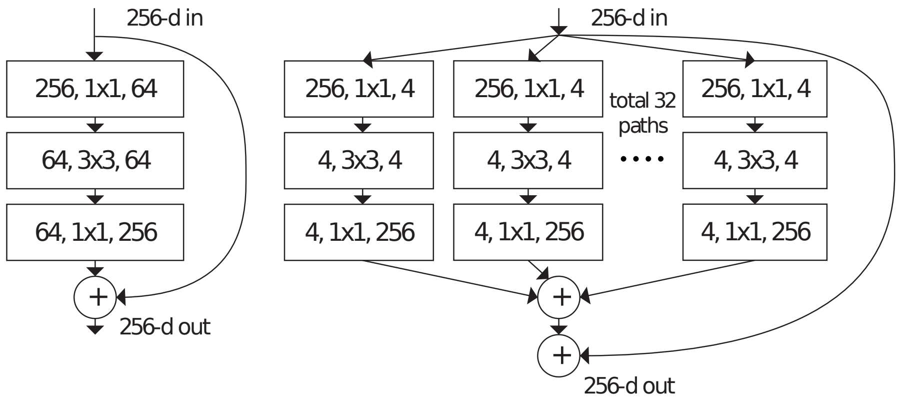


Figure 11.8: Example of ResNet (left) and ResNeXt (right) blocks from [73]. The input to each block is added to the output. Intermediate convolutional layers are shown as (input dimensions, kernel size, output filters).

kernel size k and f filters, and p indicates the pooling procedure applied if any. All intermediate layers have the same kernel size and number of filters unless explicitly stated otherwise. For ResNeXt blocks I use the notation *ResNeXtNode([a, b, c], f)*, where *[a, b, c]* indicates the intermediate kernel sizes used in each aggregated layer sequence, and f is the number of filters of each intermediate layer (except the last which is fixed to the number of input filters to the block). In this study I find a cardinality of 16 to be optimal in all implementations, therefore this can be assumed if not stated.

11.1.3 1×1 convolutions

One of the more unintuitive applications of convolutional networks is the use of 1×1 convolution kernels, first introduced in the Network In Network (NIN) architecture [74]. Unlike traditional convolutional networks, the NIN structure first considers each pixel individually, focusing the filters instead on the depth (colour) dimension of the pixels. In doing so, an initial learned preprocessing of pixel features is essentially performed before a later layer in the network is processed by larger convolutional kernels. Figure 11.9 shows a simplified one-dimensional example of a single kernel convolution. All filter values of each previous layer's individual pixels are inputs to the following layer's corresponding pixel, effectively applying a fully connected MLP pixel-wise. To give a concrete example, consider an input consisting of some number of particles each containing only their position in Cartesian coordinates (where the coordinates are analogous to colours in an image). Now consider a problem that becomes trivially solvable in cylindrical coordinates but is significantly more difficult in Cartesian space. Performing a series of initial single kernel convolutions would in theory allow the network to first approximate the coordinate transformation before further processing[2]. These 1×1 convolutions

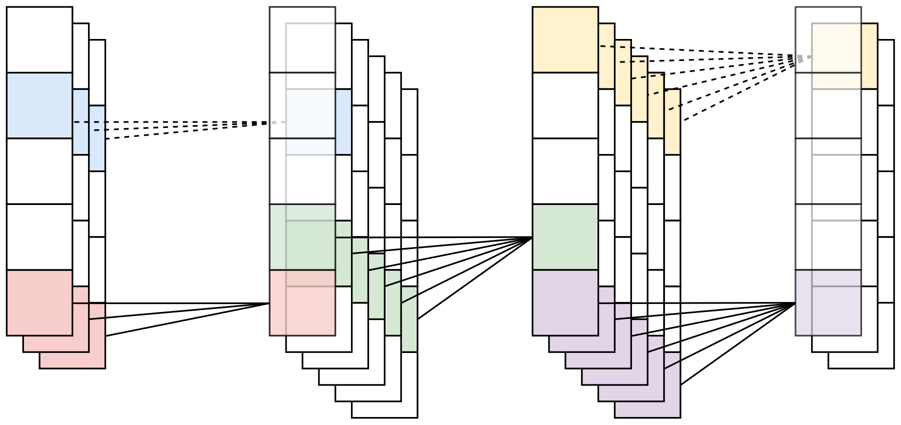


Figure 11.9: One dimensional example of 1×1 convolutions. The network structure is of four pixels input with three colour channels, followed by two layers containing six filters each and a final three filter layer. Note that the total number of pixels (five) never changes as the kernel size is one, i.e. there is a one-to-one mapping in the pixel locations. Instead only the colour channels are mixed as inputs to the convolutions. The different colours indicated the various connected pixels/filters. The dashed lines simply show connections to filters below the top visual layer.

have also been used effectively to perform dimensionality reduction in GoogLeNet [75].

11.2 Recurrent neural networks

Recurrent neural networks (RNNs) are a class of specialised networks for processing sequential data. The simplest example being the processing of individual sentences of text, in which not only the individual words but also their location in the sentence relative to other words is crucial for extracting the true meaning. Just like CNNs, recurrent networks can process sequences of varying lengths. Though again in practice the sequences are padded to a fixed length to simplify the building of the network.

Figure 11.10 shows a simplified example of a node in a recurrent network and the unrolled projection of its processing of a sequence of inputs. The loop in the node allows information from previous time steps to persist. The trade-off for this is the lack of parallelisation possible in convolutional networks. Each neuron on a layer relies on the output of its neighbour resulting in significantly longer training times. For this reason, a small number convolutional and pooling layers are typically used before recurrent layers to downsample the inputs, reducing the training time required with minimal impact on overall performance.

[2]Note the approximation of the transformation. There have been several interesting failings of convolutional networks recently [76, 77] which motivate further improvements when dealing with inputs other than images.

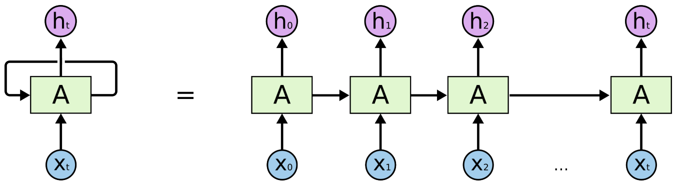


Figure 11.10: Recurrent network nodes contain loops allowing information from previous time steps to be included in the output calculation [79]. The unrolled representation on the right shows how information from previous time steps is used as input in addition to the previous network layer's input x_t at time step t.

Long short-term memory networks (LSTMs) [78] are a special implementation of recurrent networks that were designed explicitly to solve the problem of RNNs struggling to remember long term dependencies. Nowadays LSTMs are the standard method of implementing RNNs, and in their Keras implementation are referred to as *LSTM*. In general LSTM layers require little tuning, other than deciding an appropriate number of nodes to add to each layer to train independently. When describing LSTM implementations in this study I use the notation *LSTM(n)*, where n is the number of LSTM nodes in the layer. The *tanh* activation is always used for LSTM layers.

11.3 Preprocessing

Neural networks are only able to handle specific types of inputs. In general there are two classes of input that are accepted: continuous and discrete.

Continuous inputs can in theory be of any range, with the learned weights handling the appropriate rescaling of each. For example, inspecting equations (11.1) and (11.6) it can be seen that the neuron outputs will scale directly with the input. Therefore if one input, for example a particle's energy, has a significantly larger range than another, for example the particle's position, then the energy will dominate the influence on the output until it is scaled down. As the ranges of most input variables in this study span many orders of magnitude (e.g. energy ranges from Gev to MeV) I normalise all continuous input data. Two common methods of normalisation are:

min-max: $\frac{x - x_{min}}{x_{max} - x_{min}}$

z-score: $\frac{x - x_{mean}}{\sigma_x}$

In this study I have chosen min-max normalisation, though possible alternatives that may be tested in future work are discussed in section 13.1.

Discrete inputs represent different classes of a particular input type. For example in this study each input particle has an associated charge ranging from -2 to 2 in integer steps; therefore there are five classes that all particles fit in to. To reflect this the charge value of each particle can be encoded into a five bit vector, with each bit representing a flag for each charge value. This process is called *one-hot encoding*. Each of the resulting five bits in the one-hot encoded output are then used as individual inputs to the network, i.e. charge is represented by five input nodes. Recalling the node output in equation (11.1), a one-hot encoded input means that for a subset of the weights (w_i) only one of them is active:

$$z = \sum_i w_i x_i + \sum_j w_j \delta(x_j - c) + b \,, \tag{11.7}$$

where inputs j are the one-hot encoded inputs for input category c and i are all other inputs. This can be simplified to a conditional bias term

$$z = \sum_i w_i x_i + b_c \,, \tag{11.8}$$

where the normal bias now includes the learned weight for the input category (c): $b_c = w_c + b$.

Sometimes the number of classes of a discrete input is too high to practically one-hot encode. In those cases a common solution is to perform what is called *embedding*. Other solutions exist, such as the *hashing trick* [80], but are not covered in this study. Embedding is outlined in section 11.5 as it is a learned representation of discrete inputs, however inputs to embedding layers must undergo an initial preprocessing step called *tokenization*. This involves the one-to-one mapping of each input to an integer, generally in the range $[1, \#\text{categories}]$, with 0 reserved for padding as described below. For example the PDG codes assigned to each MCParticle in section 12.3 ranges sparsely from 1 to ± 1000020040. In total there are only about 540 PDG codes, with the remainder being unused. Therefore it's possible to map each code directly to an integer in the range $[1, 540]$ with this mapped integer then input to the embedding layers.

As discussed in section 11.1, the inputs to any networks containing fully connected layers must be of a fixed size. This is necessary for calculating the number of connections required in the fully connected layers when building the network. As the number of inputs in this study, particles in a simulated decay, can be of varying size I utilise padding and truncating to set all to a predetermined size. Truncating simply drops particles from inputs exceeding this size as necessary. Padding is the inverse, adding blank particles (particles with all properties set to zero) to the end of the inputs as needed.

11.4 Hyperparameters

The architectures discussed in previous sections of this chapter have already introduced various hyperparameters relating to the specific layers that each introduces. In addition to those there are hyperparameters that are variables of the network as a whole. The first being the overall architecture of the network, that is the specific sequence of layers in the network and how many times they are repeated. The various combination patterns used in this study are explained in section 12.4. There are also hyperparameters relating to the way in which data is fed to the network, the two most important being *batch size* and *epochs*.

The number of epochs is simply the number of times the training will read the entire input data set. The choice of number of epochs is made from a combination of technical constraints, how long it takes to process the entire training data set, and performance results, how many epochs before the network begins overfitting the data and validation loss increases. With the use of early stopping controls within this study, the latter is of less relevance to the choice of epochs. Figure 11.3 showed an example of when the early stopping module would execute.

As mentioned in section 11, the training/updating of the network is performed on the results of processing a small sample of the input data. The size of this sample is called the *batch size* and set at the beginning of training. The batch size can range anywhere from one, known as stochastic mode where the gradient and network parameters are updated after each input sample, to the size of the entire training sample, known as batch mode where the network is updated once per processing of all training data. The choice of batch size requires a trade-off between accuracy gained per epoch, available memory, and size of each data sample. A good general approach to finding the optimal batch size is to begin with something small, e.g. 16, and increase through powers of 2 up to something large, e.g. 512.

The learning rate hyperparameter introduced in section 11 is explored during hyperparameter optimisation. In this study I implement an adaptive learning rate. When the loss during training is no longer improving the learning rate is lowered by a factor of 0.1. This allows the training to effectively focus in on the optimal parameters as the loss approaches a minima. Additionally, I implement early stopping to prevent overfitting. The network stops training when its performance on validation data is no longer improving. This ensures that what the is network learning is not only improving performance on the training data.

11.5 Additional operations

In addition to the components of neural networks discussed in the previous sections of this chapter, there are further operations that are involved in the processing flow. In cases dealing with different input types it is useful to be able to split the network into separate sub-networks for individual processing. These sub-networks then need to be merged in order for a single output to be given as a whole. To achieve this merging two methods were used in this study: concatenate and add. Concatenation simply involves the linking of two or more layer's outputs as a single input to the following layer. For example if the outputs of two networks with individual output layers of length 10 and 20 are concatenated, the following layer with have an input of length 30. Addition involves the adding of each output value from all input layers and therefore requires that the preceding outputs are all of the same length. For example if the outputs of two networks both with output layers of length 10 are added, then the input to the following layer will also be of length 10, with the first input being the sum of the first output node from the two added networks and so forth. In Keras the layers for concatenation and addition are referred to as *Concatenate* and *Add*.

Embedding, used for discrete network inputs, involves the conversion of a categorical input into an N-dimensional vector whose direction relative to other categories represents the relation in meaning of that category. For example in the case of word embeddings the coordinates represent lexical semantics, and so the vectors between words will be similar for those with similar differences in meaning. In the learned relations shown in figure 11.11, the operation of vector("King") − vector("Man") + vector("Woman") would return a vector close to that of the word *Queen* [81]. As the embeddings must be learned they are included as an initial layer in the neural network. In Keras the embedding layer is referred to as *Embedding*, with the number of dimensions of the output vector being a hyperparameter to be tuned.

Dropout is a regularisation[3] technique used to help prevent overfitting of the network [82]. During training, each neuron within a layer using dropout is only kept active with some probability p. Otherwise the neuron and all connections in and out of it are deactivated (weight set to zero). The remaining neurons are scaled by $\frac{1}{1-p}$ so that their sum is unchanged at training and inference time. Figure 11.12 shows an example of dropout being applied to a hidden layer. The choice of p is a hyperparameter which as a general rule of thumb should increase with the number of nodes in the layer it is being applied to. Importantly, dropout is turned off during application of the network so that all connections are active during prediction. Therefore it's not uncommon to see higher performance during validation stages of training when high dropout rates are used. Within Keras dropout is applied as a separate layer of the network, named *Dropout*, between other core layers. This study uses dropout heavily in deep network models to prevent overfitting as the number of training parameters grows quickly into the millions. However, recently the usage of dropout in convolutional layers has fallen out of favour amongst the machine learning community, with overfitting instead being handled with batch normalisation.

Batch normalisation [83] is a regularisation technique created to combat a phenomenon known as internal covariate shift. Internal covariate shift describes the change in each of the network's hidden layer inputs caused by updates to the previous layers. This is thought to slow down training times as it requires each hidden layer to effectively aim for a moving goalpost. Batch normalisation attempts to stabilise this effect by normalising the inputs of each hidden layer so their distribution remains relatively constant throughout the training process. The normalisation is performed across each training batch. There are additional regularisation techniques, the most common of which is *L1 & L2 regularisation* [84] which involves updating

[3] Regularisation is a generic term used to describe any technique which reduces the generalisation (validation) error of the network but not the training error, i.e. prevents overfitting.

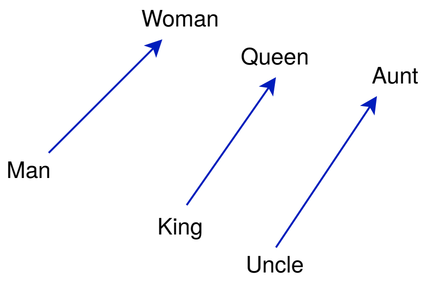


Figure 11.11: Example of learned vector representations between words with similar relative meanings [81].

the cost function to include an additional regularisation term, however these were not used in this study and are left for future work.

After training of a network is complete the state needs to be saved for later reapplication. There are two ways to do this in Keras: save the entire model, or save only the trained weights. Saving the model saves the architecture, weights, and optimiser state. Saving only the weights requires the same architecture used in training to be built before loading the weights. In the context of grid-based Monte Carlo simulations, saving the entire model is required for flexible distribution. If only the weights are saved then the basf2 module which performs the inference must be tailored to the specific architecture trained. Therefore in this study I save the entire model of each trained network. This presents a limit on the complexity of the network as the total saved model can not be larger than the maximum total supplementary input file size of 10 MB. Within the Belle II grid-computing infrastructure there is a so-called conditions database [85] which interfaces with basf2 to provide downloadable payloads at runtime. Currently, the uploading and downloading of Fast boosted decision tree and Full event Interpretation trainings is supported. Support for saved neural network models is also possible in theory, though this requires extensions to the existing interface to work for the networks in this study. Use of the conditions database is not covered in this study but something worth investigating in future.

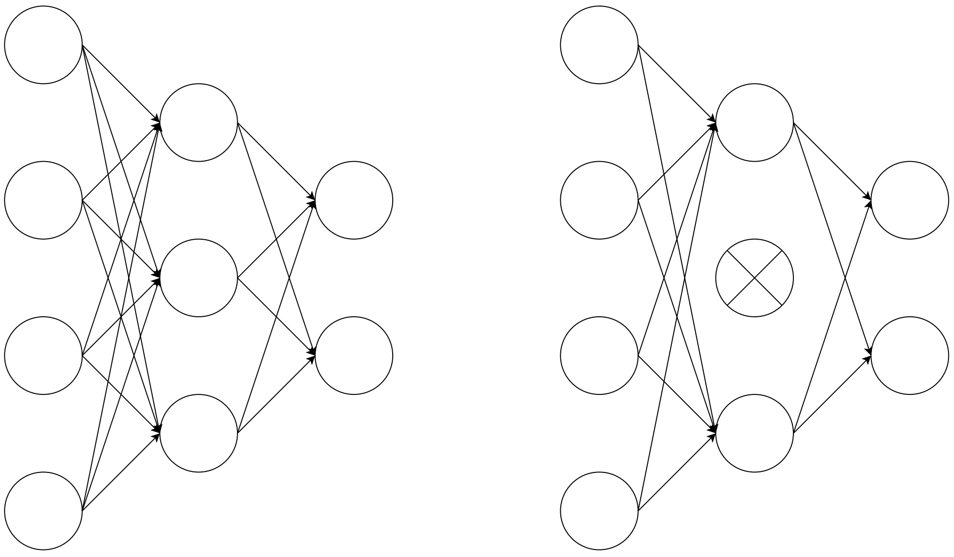

(a) Without dropout applied (b) With dropout applied

Figure 11.12: Dropout randomly removes some fraction of connections during training to discourage overfitting.

Chapter 12

Experimental procedure

The goal of this study is to be able to predict early on in the Monte Carlo simulation process how likely a simulated event is to survive the reconstruction and selections applied later, i.e. a pass event. Returning to the data flow shown in figure 10.1, the splitting of pass and fail events after, for example, the skim process can be inserted. Figure 12.1 shows the updated data flow, with fail events being discarded and only pass events continuing being processed. Here the real events coming from the experiment, data, have been omitted.

Recalling the processing times of each stage of the simulation from table 10.1, the ability to avoid performing the detector simulation and reconstruction for fail events would provide a significant reduction in the simulation time required with no loss to the data volume of pass events. Figure 12.2 shows the stages of the data flow that the neural network in the following sections attempts to learn. That network then needs to be integrated into the simulation data flow. Figure 12.3 shows where the network would integrate, as a basf2 processing module, and preemptively discard or keep events based on their predicted pass/fail probability, denoted *pass** and *fail**. The true values of which events pass or fail the skim stage are what will be used as the training labels for the network.

As all Monte Carlo simulation production is now performed on grid computing resources (section 10.3) the resulting module should ideally be compatible with the current grid job

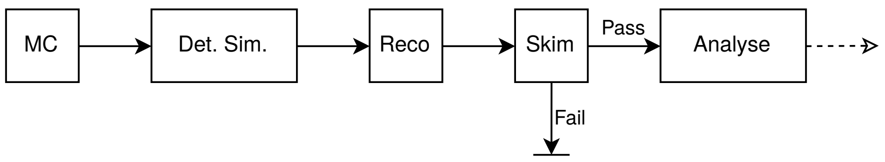


Figure 12.1: Monte Carlo simulation data flow with the retaining/rejection of events by the skim procedure shown. Pass events, those with satisfy the selection requirements of the skim, continue on to the physics analysis. Fail event are discarded and no longer processed.

requirements, namely that the supplemental files required by the module (i.e. the saved trained neural network model) total less than 10 MB. Anything over this requires specialised solutions that may involve changes to the Belle II grid computing setup itself, therefore remaining within these restrictions is desired.

The solution I have opted for is to utilise the information contained in the output of the MC stage of simulation, leveraging the existing large volume of simulation data available from the FEI skims, to explore a range of suitable neural network architectures. This is a highly data-driven solution and relies on the assumption that the kinematics of the primary decay (the decay originating from the e^+e^- collision) is the dominating factor in deciding whether an event passes the skim. Recall that the selections applied in the commissioned FEI skims from section 6.2.1 are:

nTracks ≤ 12

$M_{\textbf{bc}} > 5.24\,\mathrm{GeV/c^2}$

$|\Delta(E)| < 0.2\,\mathrm{GeV}$

sigProb > 0.001

Table 12.1 shows the skim retention rates from the same section, repeated here for convenience.

An alternative approach is to design a solution based on the known selections applied by FEI, for example to the kinematic requirements placed on the reconstructed B meson candidates. While this solution at first glance appears to be the simplest, it is in fact non-trivial to implement and in certain cases not possible at all. The candidates reconstructed by FEI are built from the final state particles reconstructed from detector hits, whereas the Monte Carlo particles (MCParticles) output by the MC stage of simulation were those used to create the detector hits. Therefore there is often not a direct mapping between reconstructed and simulated particles. There can be misidentified particles (e.g. kaon identified as pion), ghost particles (a particle reconstructed from a series of hits that was never there), or decaying particles reconstructed with the wrong combination of daughters. In all of the aforementioned cases, the kinematic selections placed on the FEI skims can not be applied directly to any of the MCParticles. Indeed by definition at least one of these cases must have occurred for every pass event in the continuum background samples. Therefore this approach is not investigated in this study.

As this is a machine learning problem at heart, I follow the typical work-flow for tackling supervised machine learning problems:

1. Data set generation

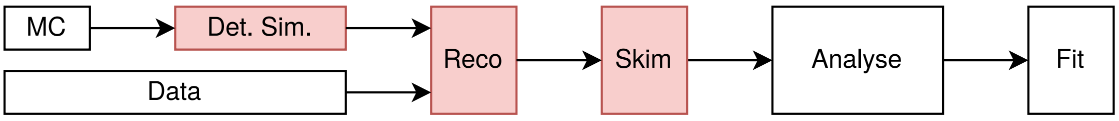


Figure 12.2: The data flow within a Monte Carlo based experiment as shown in figure 10.1. The highlighted steps are what the neural network in this study will attempt to learn. The network aims to be able to accurately predict the likelihood of an event output by the MC stage surviving all the way to the Analyse stage.

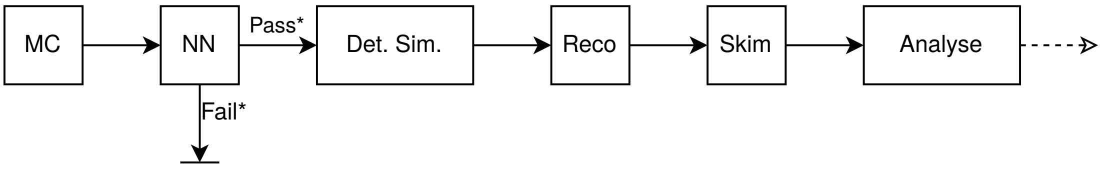


Figure 12.3: Monte Carlo simulation flow with the selective background module inserted. The predicted pass* events are kept and continue along the simulation process, events predicted to fail the skim (fail*) are discarded and the simulation moves on to the next event.

Table 12.1: FEI skim event retention rates of each channel reconstructed (repeat of table 6.2 for convenience).

Channel	Had B^+	Had B^0
K^+	3.10%	–
K^0	–	1.30%
K^{*+}	3.26%	–
K^{*0}	–	1.27%
Mixed	5.62%	4.25%
Charged	8.35%	3.82%
$u\bar{u}$	6.86%	3.78%
$d\bar{d}$	7.20%	3.39%
$c\bar{c}$	12.0%	5.73%
$s\bar{s}$	6.13%	2.95%

A sample representative of the data the network will be applied to is required for learning accurate classification. In general this should be as large as possible to allow the greatest possible generalisation by the trained network. The data set is split into training and testing/validation subsets (usually at a ratio of $0.9 : 0.1$), though in practice this is usually performed directly prior to the training stage to allow random subsets to be used for validation.

2. Feature generation

   Information is extracted from the data in the form of individual features (e.g. individual particle momentum and energy). These should attempt to encapsulate enough information to wholly describe each event.

3. Feature preprocessing

   Here the features expected to be most useful for training are selected. The data is then reshaped into a format appropriate for a neural network that encourages generalised learning.

4. Training

   Various neural networks are constructed and trained on the preprocessed data. A range of networks and techniques are explored along with different hyperparameters for optimisation. The trainings are performed event-wise, with the network learning to make a single pass prediction for each event based on the simulated particles within.

5. Evaluation

   A select set of trained networks are applied to a test case to get an independent evaluation of performance. Specifically, how much speedup do the networks provide and do the network outputs introduce bias?

12.1 Data set generation

Given that obtaining the final relevant background data for this study is by design difficult, coupled with the fact that the multiplicity of possible decay topologies is intractable, I opt to learn classifications of the output of the FEI skims (section 6.2.1). This has several benefits:

1. The background retention rate is high enough to provide the large volume of data required for such a classification task (table 6.2).

2. The data is already generated and readily available on the Belle II grid (section 10.3). This means the techniques developed in this study can be applied directly by others.

3. The data skim selected is study independent. Anyone requiring the use of the full event reconstruction tool Full Event Interpretation (FEI) can make use of the same resulting concentrated data set.

4. It provides a good test bed for identifying introduced biases as the skim outputs a reconstructed B meson allowing a statistical comparison of true positives and false negatives (section 12.5).

The FEI skims I select are the charged FEI B^+, with roughly 2×10^6 events total in the data set, and the mixed FEI B^0, containing roughly 4×10^6 events. I use these as the *pass* event samples. The skim categories are those shown in table 6.2, where mixed/charged refers to the simulated decay, and $B^{+/0}$ refers to the charge of the B meson reconstructed by FEI. The *fail* event samples I obtain from events discarded by the FEI skims. I require the fail datasets to contain roughly equal numbers of events as the pass sample, bringing the total training data set sizes to 4×10^6 events for charged FEI B^+ and 8×10^6 events for mixed FEI B^0. I mark all events saved during this stage with a boolean True/False flag for use as the training labels. True indicates an event that survived the FEI reconstruction and selection cuts (pass), and False the converse, events discarded during the reconstruction and selection stages (fail).

12.2 Feature generation

The goal in this step is to extract enough information from the available input data to be able to accurately represent the decay event. The complete set of available Monte Carlo generated information can best be thought of as a rooted tree graph with each node representing a single particle and each edge representing the connection between parent and decay daughter and each node carrying all of the information associated with the corresponding particle. Figure 12.4 shows an example of the tree structure of a single event. The data is stored in what is known as an MCParticle array, with an individual array index assigned to each particle ordered by the depth (the number of parent generations to the initial particle at array index 1). Note the varying number of daughters of each decaying particle, the range of possible numbers of daughters is an important factor that must be considered when designing network architectures for training.

The features available for each particle in an MCParticle array are homogeneous and defined by the Belle II software framework [1]. Table 12.2 lists all of the available, relevant variables for a particle and a brief explanation; for the full definitions see the glossary on page 146. For final state particles (FSPs), particles which are stable in the event generator, the lifetime, and hence the decay time and decay vertex, is infinite. Therefore I do not include these variables in the features saved during this stage of processing. I expect that this will not cause any significant information loss as it is already contained in the production vertex and the connections via mother PDG code and PDG code of each particle and their mother.

Because the MCParticles present in the MDSTs used for data set generation are those output by the reconstruction stage of simulation, there are additional MCParticles present not generated by EvtGen in the MC stage, for example secondary decays of particle interactions with the detector. These must be removed as they will not be available at the time of application of the neural network (figure 12.3). To do so I place a requirement on the status bit such that only features of primary particles are saved, with the remaining particles discarded. I also discard the array index and No. daughters as they are only valid for the reconstruction stage MCParticles.

The final choice of variables to be used for MCParticle processing are:

PDG code This alone contains information about the particle's type and charge. Since each MCParticles is simulated with exactly the nominal mass, the PDG code magnitude implicitly contains the mass information too.

Mother PDG code This is included to help the network learn to associate the properties of particles with the same mother PDG code. For example, if a low-momentum pion emitted by decaying D^* mesons provides useful identification information, inclusion of the D^* PDG code will help the network recognise this.

Charge Charge is included to be a one-hot encoded input. This is included to complement the PDG code information and help the network identify valid particle combinations based on charge conservation.

Kinematics Energy, Production time, production vertex, momentum.
The kinematic properties are selected such they completely describe the properties of each particle relevant to a decay. Namely, where and when each particle was created, as well as in which direction it travelled. Coupled with the same information from the parent particle (indicated by the mother PDG code) this provides information about the entire decay chain. Certain combinations of particles will be reconstruct-able by FEI and others will not. To give an explicit example, consider a decay in which a charged final state particle travels in the direction of the beam-pipe. This event may not be kept by FEI as that particle is never detected, regardless of whether the decay is included in those searched for by FEI. If the particle is a neutrino, however, then this decay can be kept as no reconstructed decay information was lost.

For the purposes of enabling fast reprocessing, I save all particles originating from the EvtGen generator with their corresponding variables in a Pandas [57] dataframe, with training label, event number, and array index as the multi-index used to access each particle individually.

Figure 12.5 shows an example snippet of a dataframe containing MCParticles, with the features truncated for demonstration.

While the array of particles discussed above stored in the Pandas dataframe does technically contain all of the possible information available from EvtGen, it is not the only way to represent each decay. When considering an experimental physics analysis, the first means of describing the study target is to specify a decay, set of decays, or required sub-decay channel to be present. For example in the study performed in part I of this thesis, one of the signal processes searched for was introduced as $B^0 \to K^{*0}(\to K^+\pi^-)\nu\bar{\nu}$. While this is instantly understandable to a particle physicist, who may trivially recognise this from the decay structure presented in figure 12.4, this is not the case for a neural network. It requires both an understanding that the PDG code or array index is an identifier for a given particle, and also that the same feature for the mother particle is what is needed to link the two. In the case of FEI, the processes being searched for is a large set of mostly B and D meson decays. Therefore, I store an additional dataframe containing the raw decay string, formatted with `(-->` and `<--)` separators enclosing all of the preceding particle's decay products. For example the decay $K^0_S \to \pi^0(\to \gamma\gamma)\pi^0(\to \gamma\gamma)$ is represented as:

```
310 (--> 111 (--> 22 22 <--) 111 (--> 22 22 <--) <--)
```

Figure 12.6 shows an example snippet of the resulting stored Pandas dataframe, containing the corresponding event numbers and training labels. This is now a familiar format used in natural language processing tasks, of which current neural networks excel at classifying, where the task is to understand the meaning (pass or fail) of a sentence (decay string). The hope is that this can add additional discriminating power through the recognition of decay and sub-decay patterns when combined with the information present in the MCParticles arrays.

12.3 Feature preprocessing

Arguably the most important stage in a functioning machine learning solution is the preprocessing of the input data to a format learnable by a neural network. As discussed in section 11.3, data fed to a neural network is either continuous, such as momentum and production vertex, or discrete, as PDG code and charge are. The discrete variables relevant are PDG code and the corresponding mother PDG code, and the charge. The range of possible charge values is small enough, -2 to $+2$, that a one-hot encoding is appropriate. An example of the encoding is shown in figure 12.8, where the single charge variable is expanded to five input features. The No. daughters variable is excluded as its value is modified significantly during the reconstruction phase of simulation, however if included in future it should be one-hot encoded as the charges are.

The range of possible values for PDG code is too large for one-hot encoding, roughly 540 distinct codes, and instead I encode these using an initial embedding layer within the network.

```
 1   300553 (Upsilon(4S))    E: 1.100e+01 m: 1.058e+01 p:(...) v:(...)
 2       521 (B+)            E: 5.511e+00 m: 5.279e+00 p:(...) v:(...)
 4        -421 (anti-D0)     E: 1.939e+00 m: 1.865e+00 p:(...) v:(...)
10           310 (K_S0)      E: 5.913e-01 m: 4.976e-01 p:(...) v:(...)
20             211 (pi+)     E: 2.379e-01 m: 1.396e-01 p:(...) v:(...)
21            -211 (pi-)     E: 3.534e-01 m: 1.396e-01 p:(...) v:(...)
11           211 (pi+)       E: 3.484e-01 m: 1.396e-01 p:(...) v:(...)
12          -211 (pi-)       E: 7.756e-01 m: 1.396e-01 p:(...) v:(...)
13           111 (pi0)       E: 2.234e-01 m: 1.350e-01 p:(...) v:(...)
26              22 (gamma)   E: 1.660e-01 m: 0.000e+00 p:(...) v:(...)
27              22 (gamma)   E: 5.742e-02 m: 0.000e+00 p:(...) v:(...)
 5         413 (D*+)         E: 2.453e+00 m: 2.010e+00 p:(...) v:(...)
14           411 (D+)        E: 2.239e+00 m: 1.870e+00 p:(...) v:(...)
28            -321 (K-)      E: 8.801e-01 m: 4.937e-01 p:(...) v:(...)
29             211 (pi+)     E: 5.853e-01 m: 1.396e-01 p:(...) v:(...)
30             211 (pi+)     E: 7.737e-01 m: 1.396e-01 p:(...) v:(...)
15            22 (gamma)     E: 2.138e-01 m: 0.000e+00 p:(...) v:(...)
 6         313 (K*0)         E: 1.111e+00 m: 8.617e-01 p:(...) v:(...)
16           321 (K+)        E: 8.508e-01 m: 4.937e-01 p:(...) v:(...)
17          -211 (pi-)       E: 2.683e-01 m: 1.396e-01 p:(...) v:(...)
 3      -521 (B-)            E: 5.493e+00 m: 5.279e+00 p:(...) v:(...)
 7          43 (Xu0)         E: 2.052e+00 m: 8.646e-01 p:(...) v:(...)
18           211 (pi+)       E: 4.797e-01 m: 1.396e-01 p:(...) v:(...)
19          -211 (pi-)       E: 1.572e+00 m: 1.396e-01 p:(...) v:(...)
 8          11 (e-)          E: 1.621e+00 m: 5.110e-04 p:(...) v:(...)
 9         -12 (anti-nu_e)   E: 1.820e+00 m: 0.000e+00 p:(...) v:(...)
```

Figure 12.4: Example of $\Upsilon(4S)$ decay structure within an event as represented by the Belle II software. The first two columns represent the (array index $+ 1$) and particle (PDG code and human readable name) respectively. The indentation level of the second column corresponds to the depth of the particle within the decay, with all particles originating from the initial $\Upsilon(4S)$. The remaining columns show the (non-exhaustive) array of data for each particle, shown simply for a visual demonstration of the data structure.

Table 12.2: Information available for each MC generated particle to input to network training. A detailed description of each is available in the glossary on page 146.

Variable	Definition
PDG code	Identifier of particle type and charge [9].
Mother PDG code	The particle's parent PDG code.
Mass	Particle mass in GeV/c^2.
Charge	Electric charge of the particle.
Energy	Particle energy in GeV.
Production time	Production time in ns relative to $\Upsilon(4S)$ production.
Decay time	Time stamp of the particle's decay in ns relative to $\Upsilon(4S)$ production time.
Lifetime	Difference between decay and production time of the particle.
Production vertex	Coordinates of the particle's production vertex.
Momentum	Three momenta of the particle in Gev/c.
Decay vertex	Coordinates of the particle's decay vertex.
Array index	Unique 1-based per event index of the particle.
No. daughters	Number of daughter MC particles.
Status bit	Bitmask representing the particle's MC production conditions.

```
                              PDG        mass   charge     energy
label evtNum    arrayIndex
False 681588002 0          300553   10.572033      0.0  10.996360
                1            -511    5.279530      0.0   5.482315
                2             511    5.279530      0.0   5.514045
                3             413    2.010231      1.0   2.200882
                4              13    0.105658     -1.0   1.210511
                5             -14    0.000000      0.0   2.070864
                6              22    0.000000      0.0   0.000058
                7             333    1.016531      0.0   1.209051
                8            -413    2.010280     -1.0   2.075037
                9             211    0.139570      1.0   0.154342
```

Figure 12.5: Example snippet of the Pandas dataframe output of the feature generation stage. Here the first ten particles of a single fail event are shown, sorted by array index and with the MCParticles features truncated for demonstration.

```
                                                  decay_str  label
evtNum
681588002   300553 (--> -511 (--> 413 (--> 421 (--> 310 (...  False
681588003   300553 (--> 511 (--> -421 (--> 323 (--> 311 (...  False
681588004   300553 (--> 511 (--> -411 (--> 313 (--> 311 (...  False
681588005   300553 (--> 511 (--> -413 (--> -411 (--> 313 ...  False
681588006   300553 (--> 511 (--> -413 (--> -421 (--> 310 ...  False
```

Figure 12.6: Example snippet of Pandas dataframe showing single event decay strings (decay_str). The decay strings are truncated for demonstration.

As the PDG codes are sparse integers with values ranging from 1 to ±1000020040 (α particle) I first tokenize them: each code is mapped directly to a positive integer ranging from 1 to $num(PDGcodes)$. 0 is a reserved index not assigned to any word, this is necessary for the padding described below. The resulting tokens are then given as-is to the network during training, with the training process learning an appropriate vector representation of each PDG code on the fly in Embedding layers. I do the same for decay strings, with each particle in the string being tokenized and embedded.

I normalise the continuous variables using min-max scaling to transform all variables to the range $[0, 1]$ (described in section 11.3). The minimum and maximum values for each variable I obtain from the physical limits of the experiment itself, for example the min-max range for the energies of particles is $[0.0, 11.0]$GeV with the upper limit given by the total initial energy of the electron positron collision at SuperKEKB. The limit ranges for particle vertex positions and momenta I obtain similarly, for example the vertex positions are determined by the physical size of the Belle II detector (7×7.5 m). For vertex position, the majority of particles are created within a very small volume around the interaction point (i.e. most particles involved decay rapidly). For this an alternative normalisation scheme which more evenly distributes the particle positions may provide some benefit and is discussed in section 13.1.

Before the data is able to be fed into the networks for training a final preprocessing must be performed. As discussed in section 11.1, convolutional networks are by themselves able to handle arbitrary sized inputs (in the width and height dimensions, depth must always be fixed). When the network contains fully connected (MLP) layers, however, the inputs must then be padded or truncated to some fixed size. This is necessary for the calculation of node connections and parameters when first building the network. Figure 12.7 shows the distribution of number of particles in the MCParticles data and the lengths of decay strings. I conservatively fix the sizes to 100 for the MCParticles (i.e. 100 particles at most per event) and 150 for the decay string (i.e. 150 PDG codes and `(-->/<--)` separators). Any events below these sizes I pad with zeroes to populate the remainder, and any exceeding this I truncate. The sizes are intended to include all possible input particle information.

Finally, any discrepancy between the number of pass and fail events must be accounted

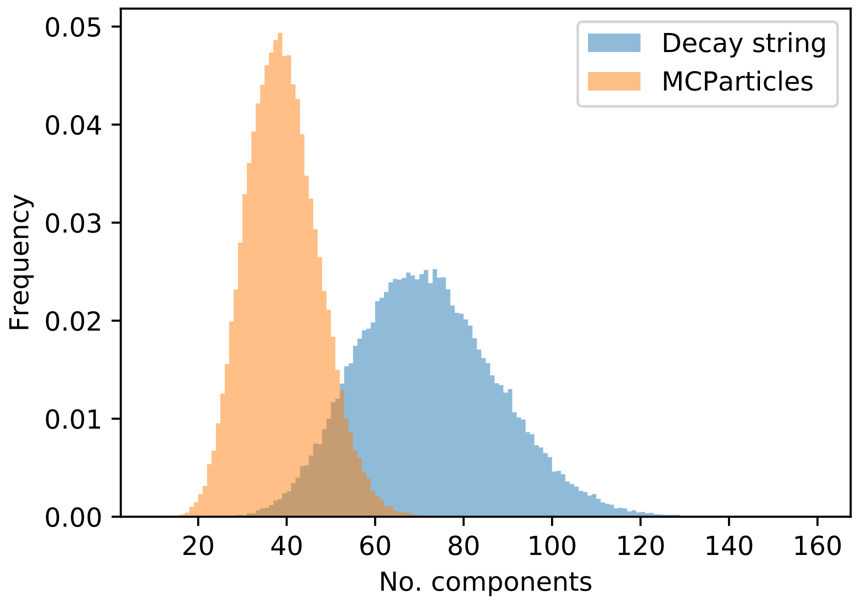


Figure 12.7: Distribution of number of MCParticles (orange) and length of decay strings (blue) per event.

for, as an over-representation of either risks encouraging cheating by the network. In cases where the network is unable to clearly classify an event it may learn that always guessing the over-represented class results in a better performance. To combat this I use class weights. Class weights tally the number of samples (labels) in the training data set of each class and add the imbalance as a weighting factor to the loss function. Recalling the binary cross-entropy loss from equation (11.4), the class weights are added to their corresponding label losses as

$$-\left(w_0 y \ln p + w_1(1-y) \ln 1 - p\right), \tag{12.1}$$

where w_0 and w_1 correspond to the weights calculated for the fail and pass classes respectively.

12.4 Training

This section presents the results of network trainings and the various architectures tested. The goal of this is to find the best network setup appropriate for the end use case and guide future improvements to the simulation procedure when considering current machine learning techniques. In the following sections I perform trainings of a range of common and state of the art neural network architectures on the individual datasets corresponding to the event decay strings (section 12.4.1), the MCParticles information (section 12.4.2), and the two combined (section 12.4.3). Finally, I give my remarks on the training results overall (section 12.4.4).

```
                          c_-2  c_-1  c_0  c_1  c_2
label evtNum    arrayIndex
False 681588002 0            0     0    1    0    0
                1            0     0    1    0    0
                2            0     0    1    0    0
                3            0     0    0    1    0
                4            0     1    0    0    0
                5            0     0    1    0    0
                6            0     0    1    0    0
                7            0     0    1    0    0
                8            0     1    0    0    0
                9            0     0    0    1    0
```

Figure 12.8: Example snippet of Pandas dataframe showing one-hot encoded charges of the first ten particles in an event.

The rough procedure in each of the training sections is similar. I first create a benchmark training using a simple fully connected feed-forward (MLP) network. This sets the baseline performance from which to compare other trainings. I then test various architectures from the literature and compare their training results to each other and the benchmark. I compare the final score (validation loss and accuracy), in addition to the training time and model size. Validation loss is a direct measure of how well the trained network will be able to reject fail events. The training time on the other hand is an important consideration for scalability; if the training is to be repeated for every channel of every skim within the collaboration, or by individuals looking for analysis-specific background simulation, then the total training time required should be minimal. This is an especially important point when considering that many researchers and institutes may not have access to GPU resources for training, and hence suffer from significantly slower training times. I also monitor model size for suitability of usage in grid-based Monte Carlo production (section 10.3).

For all of the trainings I vary the relevant hyperparameters described in Chapter 11 within reasonable values until optimal training performances are found. In general this is done according to the recommended ranges of the associated literature for the given network type. In almost all hyperparameter optimisations tested I found that the Adam optimiser with an initial learning rate of 0.001 provided as-good or better training performance as other common optimisers. The reduction of the learning rate on training plateaus allows smaller learning rates to also be accessed. AMSGrad [66] was left enabled in all cases. Similarly, I found the LeakyReLU activation to be robust in the various architectures tested and in all hyperparameter optimisations tested achieved the maximal performance. I applied batch normalisation to intermediate network layers where appropriate to mitigate overtraining and improve training times. Therefore throughout the following sections these hyperparameters may be assumed unless explicitly stated otherwise. In the training closing remarks in section 12.4.4 I comment on how this simplification may be improved upon in future.

Throughout this section I present the results of network trainings as a best-of by showing the progress of the following four metrics across epochs: binary cross-entropy loss (*loss*), prediction accuracy (*acc*), and both of these when validated on independent data (*val_loss* and *val_acc*). The significant size of the training data warrants the splitting of the total dataset into sub-epochs to allow validation after reasonable time intervals required for monitoring for overtraining. These sub-epochs are what is shown in the metric plots and I fix at four sub-epochs during training[1]. Due to the use of early-stopping the number of epochs shown in each metric plot may vary. I set the early-stopping requirement at three epochs (not sub-epochs) such that if no improvement in the validation loss is seen after three consecutive passes over the training data then training is stopped. The choice of three epochs is to allow for several reductions in the optimiser's learning rate. To prevent overfitting throughout the trainings I relied heavily on regularisation (section 11.5), which is switched off during validation. Therefore, many of the training metrics show a validation performance slightly above that of training.

In reading the metrics plots the reader should look for the following:

- The final (rightmost) values of each metric. Due to the splitting of training and validation data into subsets the metrics are susceptible to small random statistical fluctuations, therefore the metric values after several epochs are the better indication of each metric's final average.

- Differences between training and validation values. Because of the use of dropout regularisation to prevent overfitting, a slight over-performance of the validation metrics can be expected when the network is able to generalise well. A significant under-performance of the validation metrics indicates a training too specialised towards the training dataset.

- A consistent trend of diminishing improvement of each metric. A well-tuned network should show a consistent but decreasing improvement, especially in the training data metrics, throughout the training lifetime. Note that in the case of very large datasets with long training times between measurements of metrics (as is the case for the mixed FEI B^0 here) this can be less obvious, with the networks achieving near optimal performance early on.

- Accuracy as an intuitive comparison and the loss for a quantitative comparison. As explained in Chapter 11, the loss is a direct measure of the network's prediction error, whereas accuracy measures the mean of the difference between the rounded network output and the true value (recall in binary classification the network outputs a value in the range $[0-1]$).

The architecture of each optimised network is given in tables at the end of each training section. Where possible I have reused mixed and charged training architectures for simplicity. In those cases I observed no noticeable penalty to training performance. Included in each table is the best validation loss achieved by each network. In certain cases, especially in early stages of mixed trainings, large fluctuations in the loss are seen. This can occur when the learning rate of the network is high and the training is making large adjustments to the network as it explores the network parameter space. While doing so the network may chance upon configurations that perform exceptionally on a particular subset (sub-epoch) of the validation data. This configuration, however, does not generalise to the entire data set and the fluctuations dissipate as training progresses. Sometimes this produces a validation loss that is not representative of the final performance of the network. In those cases I instead present the final validation loss.

All of the trainings in this study were performed on a host with Ubuntu 16.04 LTS, 64 Gb of local memory, 40 Intel Xeon E5-2660 v3 CPUs, and an NVIDIA GeForce GTX 1080Ti GPU. Any times given with regards to training or inference are with respect to this host alone and will vary when performed on other machines.

12.4.1 Decay string training

The first trainings I perform are on the decay strings alone. By construction, Full Event Interpretation is targeted towards specific processes: it performs a bottom-up reconstruction of a predefined set of sub-decays. Therefore, this is a reasonable first step. I equate each sub-decay within the overall event decay string with the semantic components of a sentence. Figure 12.9 shows and example of the analogous syntax tree representations of language and particle decays, with the leaves (nodes) of the trees being the only components detected by the observer, i.e. only final state particles are detected in experiment. Therefore, known solutions to language processing problems provide a good starting point. There is a difference to be noted here though: in the processing of the decay strings, I have access to information from the entire syntax tree, whereas most language processing tasks must learn the tree from the sentence words (leaves) themselves. An alternative approach to parsing syntax trees is the use of recursive neural networks [86, 87] which are able to sequentially process the tree while maintaining structural information. However the use of recursive networks is still in the primitive stages at the time of this study and the majority of applications are on representations in the form of binary trees (or at least each node having a fixed number of daughters). Given that particle decays can results in a range of number of daughters this makes the use of a recursive network here significantly more complicated. Therefore I neglect its use in this study and leave it as an avenue for future investigation when the use of recursive networks in the machine learning

[1] Except in certain cases of the charged FEI B^+ trainings in which significantly larger epochs are seen. In those cases I utilised a larger number of sub-epochs for fine-tuning of the networks, however it has no effect on the overall training performance as it simply increases the number of times data validation is performed.

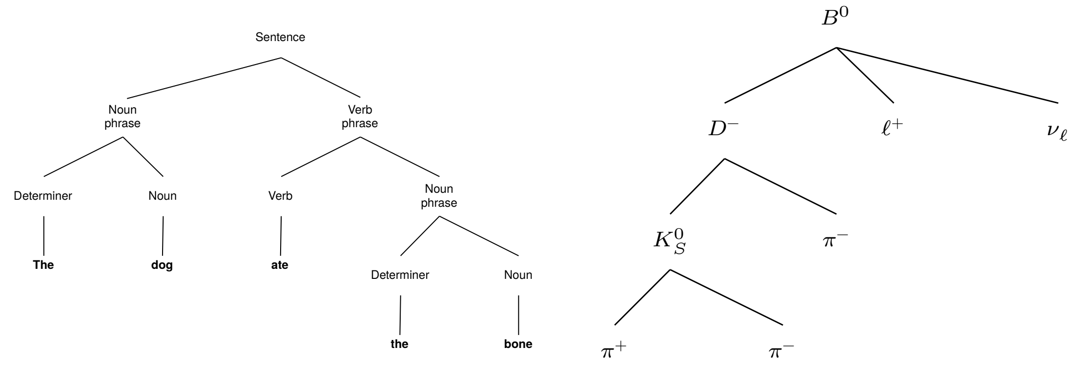


Figure 12.9: Syntax tree representation examples of language (left) and particle decays (right). Both exhibit similar semantic structures which allows established solutions for language processing to be adapted to physics problems.

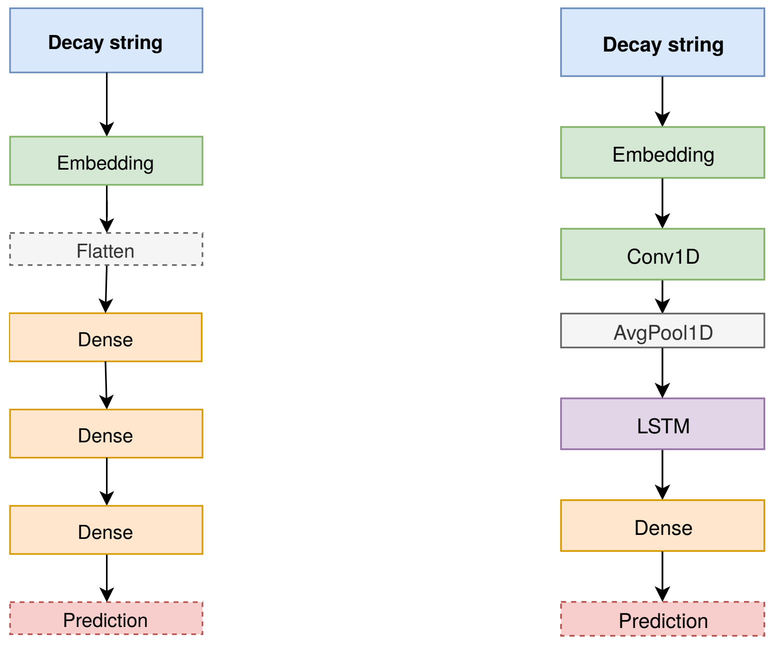
community is more well developed. This point I elaborate on in the Outlook (section 13.1) where I discuss the use of graph neural networks instead [88, 89].

To asses the relative performance of various architectures and hyperparameter optimisations explored in this chapter, I first create a benchmark training. The baseline I use is a simple, fully connected (MLP) network. The architecture I implement for the decay string training benchmark is shown in figure 12.10a, with the corresponding best performing training's metrics in figure 12.11. These results, after hyperparameter optimisations, set the decay string benchmarks at the validation losses shown in table 12.3. The final trained network model when saved is roughly 40 MB in size. This is well above the 10 MB limit outlined in section 10.3.

The next network architecture I train on decay strings is shown in figure 12.10b. This architecture has been shown to perform optimally on a wide range of language processing tasks [90] with the initial convolutional layer allowing the network to also extract any information from spatial features of the decay string. As every input decay follows the structure $\Upsilon(4S) \rightarrow B(\rightarrow ...)B(\rightarrow ...)$, where in the case of neutral B mesons the two B's are interchangeable, the relative position of sub-decays in the complete decay string will affect which sub-decays the FEI has attempted to reconstruct. Therefore a spatially aware recurrent network architecture is a reasonable starting point.

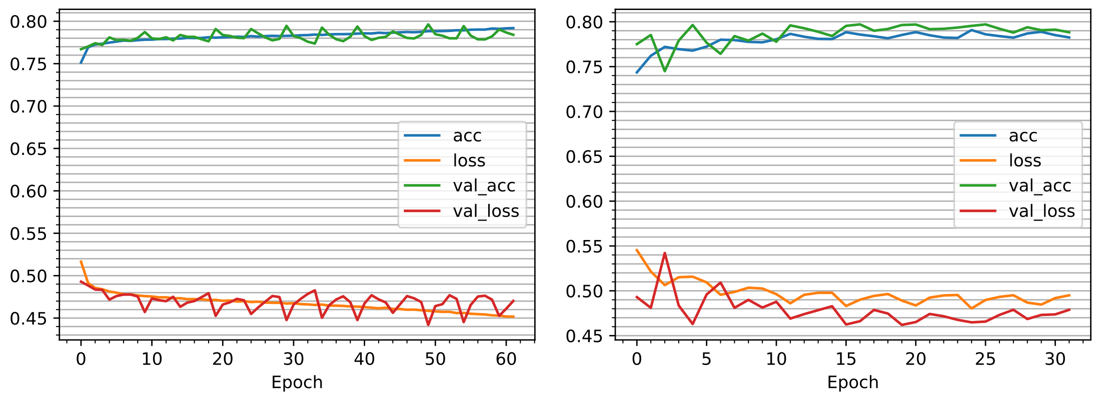
LSTM layers cannot be processed in parallel within a processor as convolutional layers can, where a single layer's filter can be applied to all inputs simultaneously. This results in significantly slower processing times for LSTM networks than convolutional networks. To combat this, I use intermediate pooling layers to reduce the input parameters to the LSTM layer, resulting in a significant reduction in training times.

Figure 12.12 shows the training results of the best LSTM architecture on both the charged



(a) MLP (b) LSTM

Figure 12.10: Base network architectures trained on decay strings. As part of the hyperparameter optimisation the number and sizes of layers used was varied, this diagram simply shows their relative orderings within the network.



(a) Charged FEI B^+ (b) Mixed FEI B^0

Figure 12.11: Examples of training metrics of best performing fully connected (MLP) architecture on decay strings. The blue and green lines show the training and validation accuracy respectively. The orange and red lines show the training and validation losses respectively.

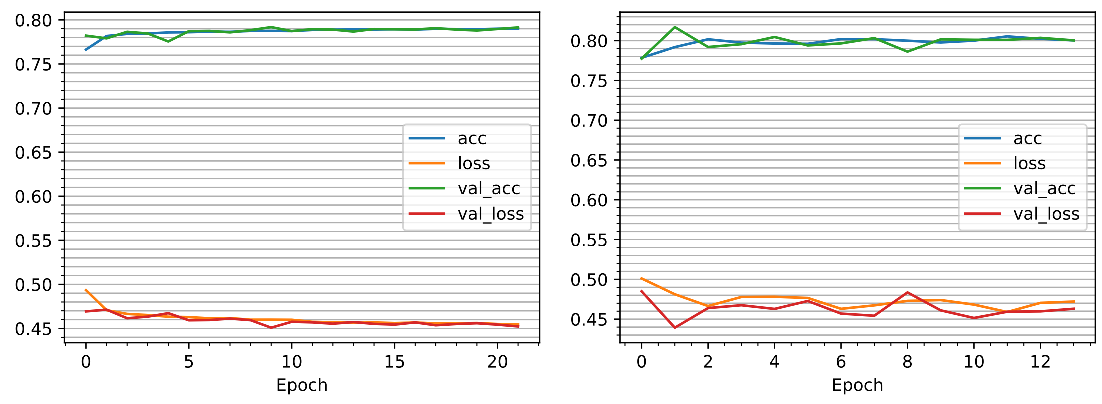


Figure 12.12: Examples of training metrics of best performing CNN-LSTM architecture on decay strings. The blue and green lines show the training and validation accuracy respectively. The orange and red lines show the training and validation losses respectively.

and neutral data samples. Both achieve a validation accuracy around the 80% range, with the validation loss values given in table 12.3.

Another approach which has shown great promise in text classification tasks is the use of convolutional layers with fully connected final layers [91, 92]. This architecture is introduced in [93], with the architecture used in the original study shown in figure 12.13. My motivation for implementing this multi-kernel structure in this study is the variation in number of daughters in any individual particle's decay. For example, two-body decays in the decay string require five characters (`['X', '(-->', 'd0', 'd1', '<--)']`), therefore a kernel size of five can capture each of these in their entirety. In contrast, high multiplicity decays can have up to five or six daughter particles, so larger kernel sizes are required to achieve the same encapsulation. A key advantage to this setup is the immense speedup in training and inference times with the absence of any recurrent layers. A single training epoch performed on the entire charged FEI B^+ input sample of decay strings with a batch size of 128 showed an average training time of 190 s and 970 s for the wide CNN and CNN-LSTM respectively. This architecture of multiple parallel kernel sizes I refer to as *wide CNN* for the remainder of this thesis.

I test variations on this wide convolutional architecture and find that the wide CNN structure shown in figure 12.14 gives the best performance while retaining a fast training time on GPU. The architecture takes the output of the embedding layer and inputs it into multiple convolutional layers in parallel, each with differing kernel sizes ranging from four to nine. I global average pool and concatenate each of the convolutional layer outputs before input to the final set of Dense layers.

Figure 12.15 shows the characteristic optimal performance of the wide convolutional network. The performance is slightly worse than that of the benchmark for the charged

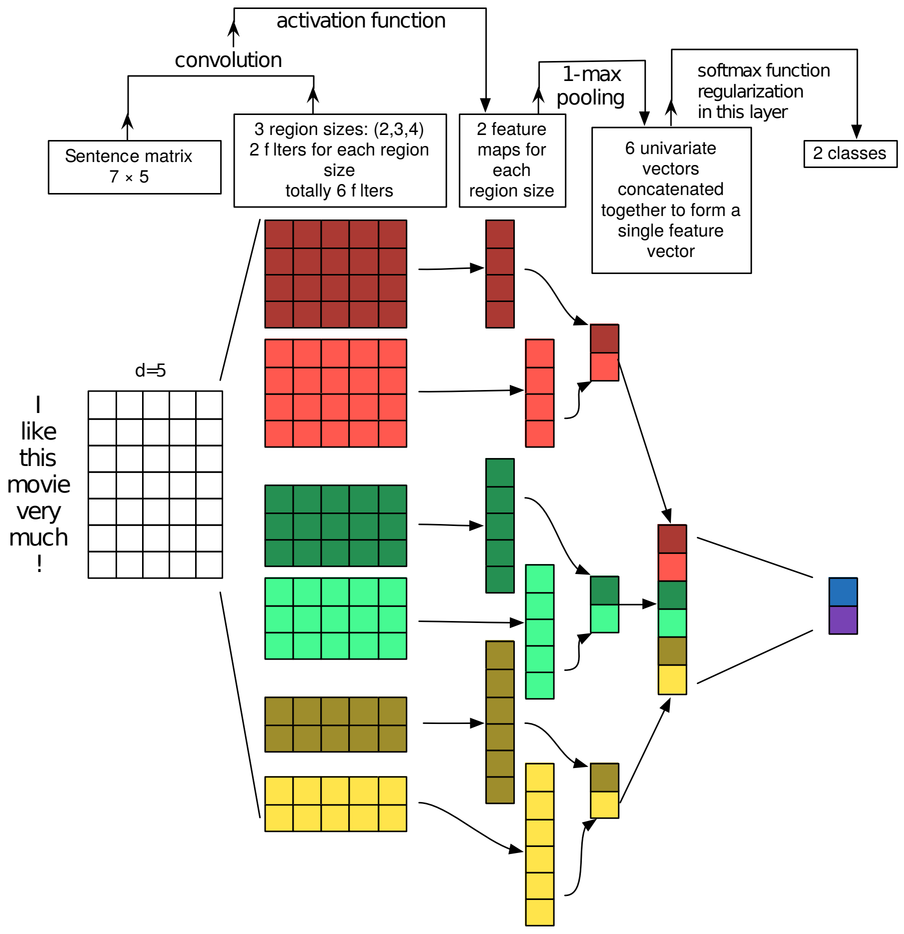

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 f lters for each region size totally 6 f lters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 12.13: Wide convolutional architecture used in [93]. The inputs (far left) in this study were word sequences (sentences) embedded into five dimensions ($d = 5$), with the embedding layer not shown here. The second layer shows three kernel sizes (2, 3, 4) with two filters each. The filters are then max pooled, concatenated, and finally output via a fully connected layer to two classification nodes (far right).

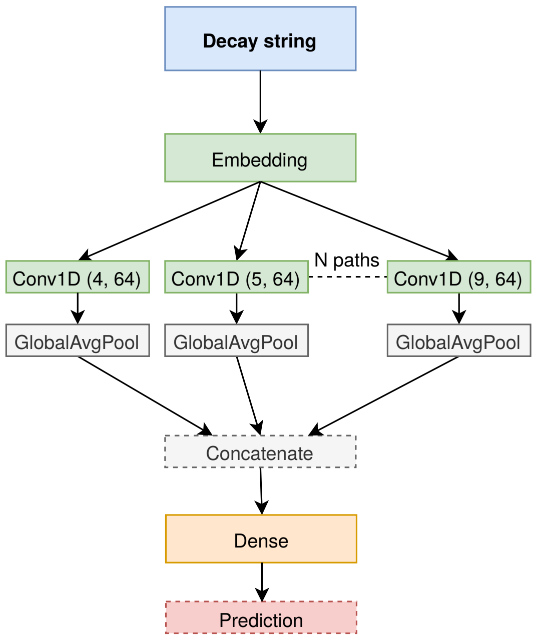


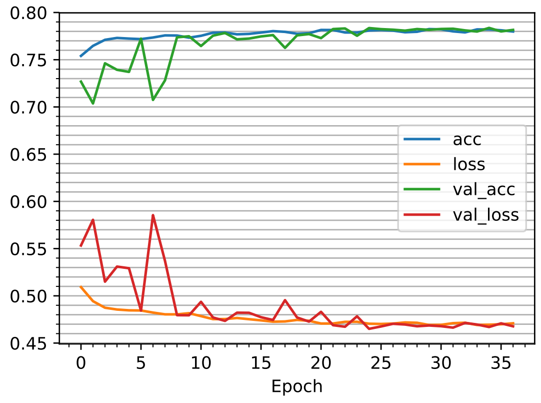
Figure 12.14: Wide convolutional architecture for processing of embedded decay string.

training. Nevertheless, the small training time per epoch and saved network size of 2.3 MB makes this an appealing architecture for use when scaling up the trainings for production use in grid-based Monte Carlo simulation.

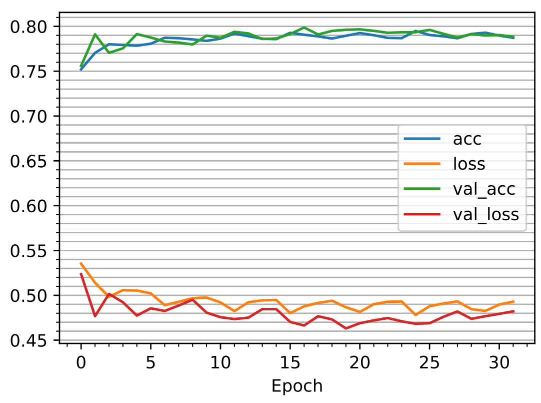
12.4.2 MCParticles training

The next set of trainings I perform on the MCParticles array alone. In this solution I attempt to classify each event from patterns in the kinematics of the decay. The initial trick here is to equate the MCParticles structure (figure 12.4) with a picture, such that each particle is treated as a pixel and each particle's properties are the pixel's colour. An image has three dimensions: height, width, and colour. The MCParticles array has instead only two: particles (width) and their properties (colour). Therefore a one-dimensional convolutional network is appropriate. As image processing via the use of convolutional networks is already a well-established field, I make use of some known robust network architectures, in particular: vanilla CNN, ResNet and ResNeXt (section 11.1.2), and 1×1 convolutions (section 11.1.3).

To provide a baseline for the convolutional networks I begin with a fully connected (MLP) network. The base architecture I use is shown in figure 12.16a. I first embed both the PDG code and mother PDG code inputs using a shared embedding layer to reduce training parameters. I also test separate embedding layers and find no improvement in training loss. I follow this with a flattening of the data to produce a one-dimensional array appropriate for input to Dense layers. Figure 12.17 shows the metrics of the baseline training after hyperparameter



(a) Charged FEI B^+



(b) Mixed FEI B^0

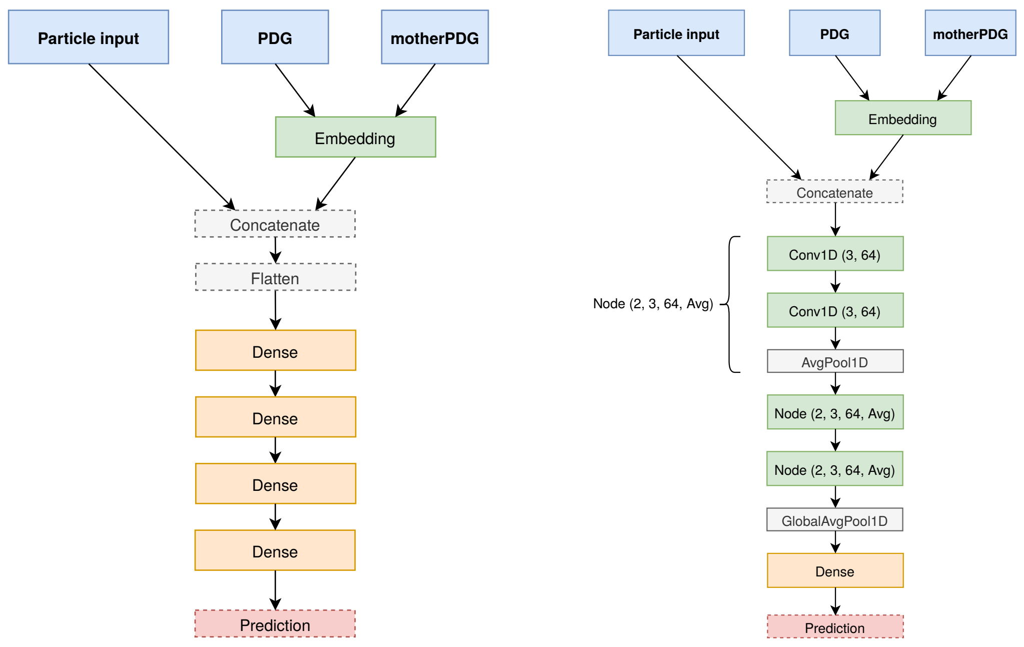
Figure 12.15: Examples of training metrics of best performing wide convolutional architecture on decay strings. The blue and green lines show the training and validation accuracy respectively. The orange and red lines show the training and validation losses respectively.

Table 12.3: Optimised architectures for decay string trainings.

Model	MLP (B^+)	MLP (B^0)	CNN-LSTM (B^+/B^0)	wideCNN (B^+/B^0)
			Embedding (8)	

Layers	Flatten
	Dense(1024)
	Dense(1024)
	Dense(1024)
	Dense(512)
		Flatten
		Dense(1024)
		Dense(1024)
		Dense(256)
			Conv1d(3, 64)
			MaxPool(2)
			LSTM(64)
			Dense(256)
			Dense(64)
				Conv1d(4–10, 64)
				GAvgPool
				Concatenate
				Dense(256)
				Dense(64)

Val. loss	0.442	0.462	0.451/0.463	0.465/0.463



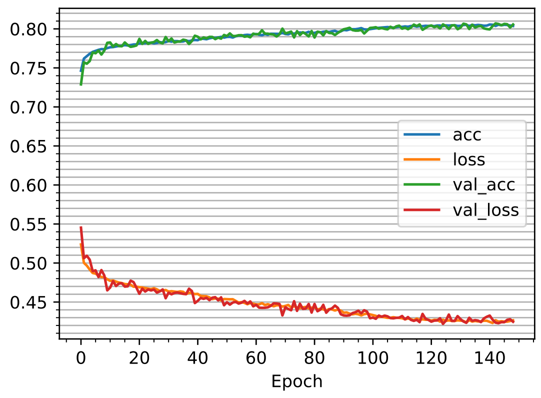
(a) MLP (b) CNN

Figure 12.16: Template of network architectures used to process MCParticles.

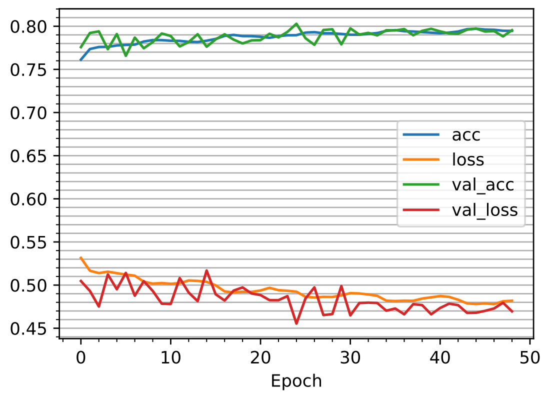
optimisation. The fully connected baseline is only able to achieve about an 80% validation accuracy after optimisation (losses shown in table 12.4). Crucially, the number of training parameters is $\mathcal{O}(10^7)$, resulting in a saved model file size range of 65 to 180MB (depending on hidden layer sizes). As in the decay string trainings, this is well above the grid-based Monte Carlo production limit of 10 MB.

The basic architecture of vanilla convolutional networks I implement is shown in figure 12.16b. I perform the embedding-concatenate sequence to join the inputs, I then concatenate the embedding outputs and pass them through a sequence of convolutional and pooling layers. The sequence shown of multiple convolutional layers followed by a pooling layer is one I use regularly in convolutional architectures in this study. Therefore, I refer to this specific sequence when discussing convolutional networks as a *Node(n,k,f,pool)*, where *n* is the number of convolutional layers applied in the Node, *k* is the kernel size used in each convolutional layer, *f* are the corresponding filter sizes, and *pool* describes the pooling operation performed. If the filter sizes are not given they can be assumed to be 64.

The optimised results of the vanilla CNN are shown in figure 12.18. Immediately I observe a slight increase in the final training loss, with a significant difference in training parameters required to achieve it: $\mathcal{O}(10^5)$ for the CNN compared to $\mathcal{O}(10^7)$ for the MLP. The large fluctuations seen in the early epochs of the mixed training are present in all hyperparameters configurations I test. This doesn't appear to affect the end result of the training, with the



(a) Charged FEI B^+



(b) Mixed FEI B^0

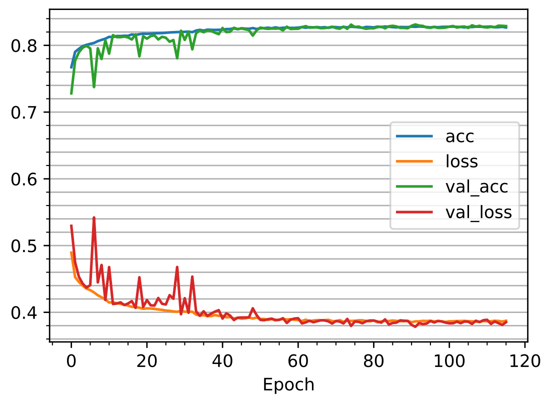
Figure 12.17: Baseline training metrics from fully connected (MLP) architecture trained on MCParticles array.

fluctuations dissipating over training time. In an attempt to reduce this I try lower initial learning rates and training without AMSGrad enabled. This produces no significant changes to the initial fluctuations. Overall, the vanilla CNN provides a competitive training performance with significantly lower memory overhead and training time. The saved model size is only roughly 3 MB, making it suitable for grid-based Monte Carlo production jobs.

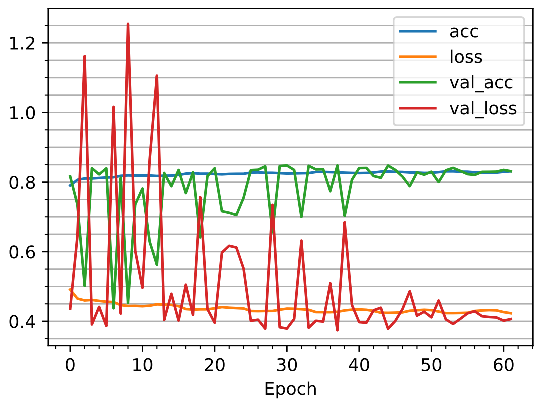
Next I train both the ResNet and ResNeXt variants. Figures 12.19a and 12.19b show the optimal training results of ResNet and ResNeXt respectively. I find no significant improvement over the vanilla CNN for ResNet. ResNext performs worse in both trainings, though this is likely due to the smaller network size used. The large array of connections in each grouped convolutional block significantly increases the training time required. Therefore to allow reasonable training times necessary for the many training runs needed for hyperparameter optimisation, I utilise pooling in the early layers to down-sample the data. Even with the use of pooling, the ResNeXt trainings still requires almost nine times the vanilla CNN training time per epoch.

Both ResNet and ResNeXt appear to significantly reduce the initial training fluctuations in the mixed training. For the ResNet the total training time required is also less than that of the vanilla CNN. This is not unexpected; the skip-connections in residual networks mean that changes to earlier layers results in larger changes to the network output, i.e. have a greater impact on the loss. Recall from Chapter 11 that the training procedure updates weights and biases according to the gradient of the loss function. Therefore, residual networks have a finer control over all network layers (see [94] for details of the gradient calculation for ResNets).

Finally, I train a 1×1 convolutional architecture. As discussed in section 11.1.3, when stacked in successive layers the 1×1 convolutions effectively apply a standard MLP network to each individual pixel. While this has uses in dimensionality reduction (e.g. as used in



(a) Charged FEI B^+

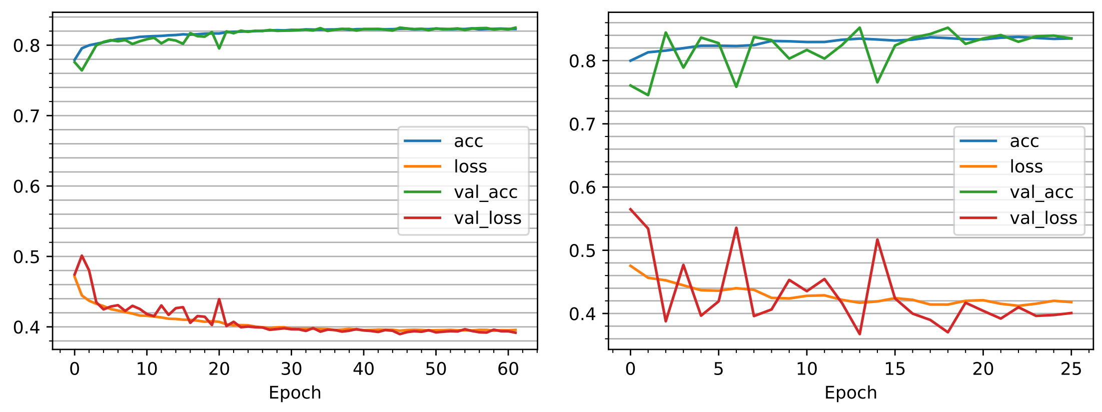


(b) Mixed FEI B^0

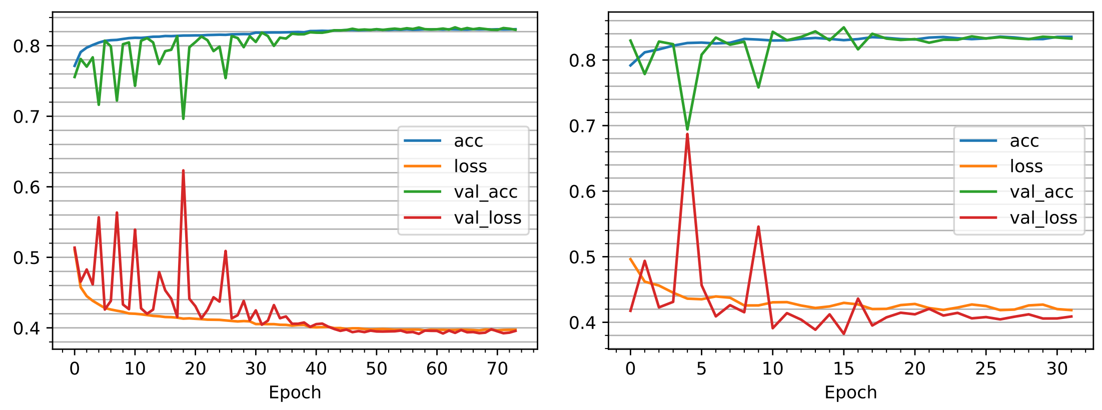
Figure 12.18: Training metrics from vanilla convolutional architecture trained on MCParticles array.

GoogLeNet [75]), here I am interested in the direct application to the MCParticle inputs. The individual particle properties are variables intuitive to humans performing physics calculations (energy, three-momentum, etc.), however this does not guarantee they are optimal for a neural network's classification. Once individual particle property manipulations have been performed by the 1×1 convolutions, I pass the resulting outputs to the vanilla convolutional layers. Note that both of these steps are trained together (not as separate networks), it is simply convenient for understanding the underlying operations to make the distinction. Figure 12.19c shows the results of the 1×1 trainings. I observe a slight improvement in the mixed FEI B^0 training, especially with respect to the stability of the validation metrics.

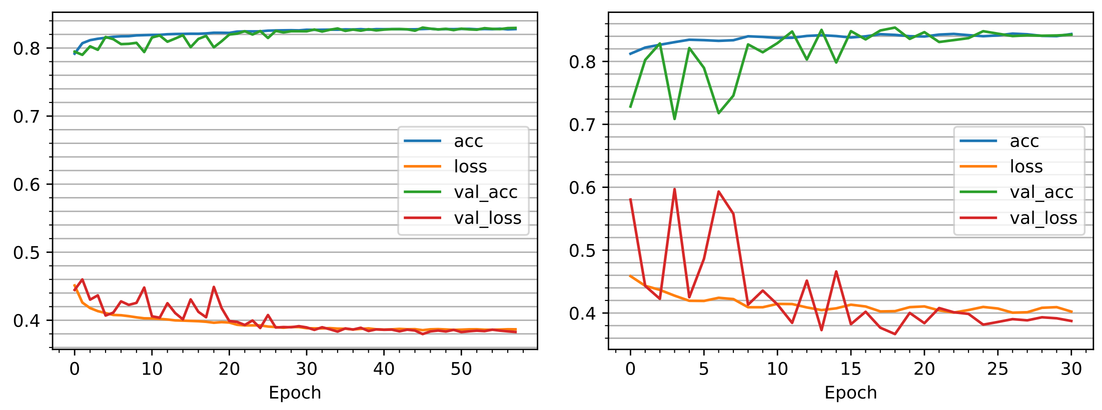
All variants of the convolutional architectures outperform the benchmark and all decay string trainings. I have summarised the best performing architectures and their losses in table 12.4. I find the 1×1 convolutions to provide the most robust trainings, achieving the best average loss when considering both charged and mixed. The key difference between the convolutional architectures appears to be the training stability, with the results suggesting a hypersensitivity to fluctuations in the particle property inputs. I infer this from the residual network skip-connections showing that the instabilities are stabilised by greater control of the early layers in the network. I also note that the 1×1 convolutional architecture has the same final convolutional layers as the vanilla CNN. The non-linearities introduced to each particle's set of properties by the 1×1 convolutions allows for a non-linear renormalisation of the particle's properties as well as weighted combinations. To isolate this further, I recommend exploration of different input normalisation schemes, though this is outside the scope of this study and a point I outline in the Outlook (section 13.1).



(a) ResNet



(b) ResNeXt



(c) 1×1

Figure 12.19: Training metrics for charged FEI B^+ (left column) and mixed FEI B^0 (right column) for the ResNet, ResNeXt, and 1×1 convolutional architectures trained on MCParticles array.

Table 12.4: Optimised architectures for MCParticle trainings.

Model	MLP (B^+)	MLP (B^0)	Vanilla CNN (B^+)	Vanilla CNN (B^0)	ResNet (B^+/B^0)
	PDG code/mother PDG code: Embedding (8)				

Layers	Flatten
	Dense(2056)
	Dense(2056)
	Dense(2056)
	Dense(512)
		Flatten
		Dense(1024)
		Dense(1024)
		Dense(1024)
		Dense(512)
			Node(2, 4, 64)
			Node(6, 3, 64)
			GAvgPool
			Dense(512)
			Dense(128)
				Node(1, 4, 64,Avg)
				Node(1, 3, 64,Avg)
				Node(4, 3, 64)
				GAvgPool
				Dense(512)
				Dense(128)
					ResNetNode(2, 3, 64, Avg)
					ResNetNode(2, 3, 64, Avg)
					ResNetNode(2, 3, 64)
					ResNetNode(2, 3, 64)
					GAvgPool
					Dense(256)
					Dense(128)

Val. loss	0.422	0.470	0.378	0.406	0.388/0.401

Model	ResNeXt (B^+/B^0)	1×1 CNN (B^+/B^0)
	PDG code/mother PDG code: Embedding (8)	

Layers	Conv1D(3, 64)
	MaxPool(2)
	ResNeXtNode([1, 3, 1], 4)
	ResNeXtNode([1, 3, 1], 4)
	GAvgPool
	Dense(512)
	Dense(128)
		Node(6, 1, 32)
		Node(6, 3, 64)
		GAvgPool
		Dense(512)
		Dense(128)

Val. loss	0.391/0.409	0.380/0.387

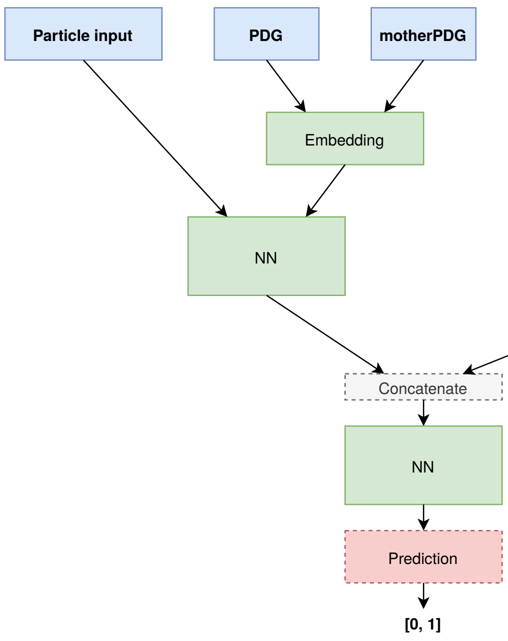


Figure 12.20: Outline of the structure of the combined networks used in this section.

12.4.3 Combined trainings

The final classification model I investigate in this study is the combination of the previous two training methods discussed. Here I combine the outputs of the best performing networks for processing decay strings and MCParticles in a final combined network to make a single prediction using all available information. Figure 12.20 shows the base structure I use, where the initial embedding of inputs is the same and simply the final concatenate-NN before the prediction is added. For these trainings I only copy the network architectures from the previous sections; each training I perform from scratch on a re-initialised network.

Similar to the MCParticles trainings, I begin with a fully connected network for the MCParticles and decay string inputs as a benchmark. I use the already-optimised fully connected network architectures from the previous two training sections. I follow these with three fully connected layers of 256 nodes, each with a dropout of 0.2 for all but the last layer (selected after hyperparameter tuning). I change the output of the decay string sub-network from a single output with sigmoid activation to a size 32 output with LeakyReLU activation. I remove the fully-connected layers following the global average pooling of the MCParticles sub-network, and directly concatenate the output with that of the decay string sub-network. Additionally, I lower the learning rate to 5×10^{-4} to encourage training stability in the mixed FEI B^0 training. Figure 12.21a shows the resulting benchmark training metrics. Immediately I observe an improvement in the training performance, with validation accuracy in both training datasets jumping to the 0.86–0.88 range (and loss dropping correspondingly). This benchmark training does still suffer from the deficiencies of the previous sections' benchmarks though: the saved model is over 100 MB and the trainable parameters are around 8.6×10^6.

Next I test the vanilla CNN (fig. 12.21b), ResNet (fig. 12.22a), ResNeXt (fig. 12.22b), and 1×1 convolution (fig. 12.22c) architectures for the MCParticles sub-networks. In all cases I use the wide CNN from section 12.4.1 for the decay string sub-network. I vary the dense layers in the final combined sub-network as part of the hyperparameter optimisation, in addition to exploring various combinations of the initial sub-networks (adding their outputs, changing sub-network output sizes, etc.) Finally, for completeness, I also train the LSTM architecture used in section 12.4.1. For I apply a vanilla CNN to the MCParticles input. The results are shown in figure 12.21c.

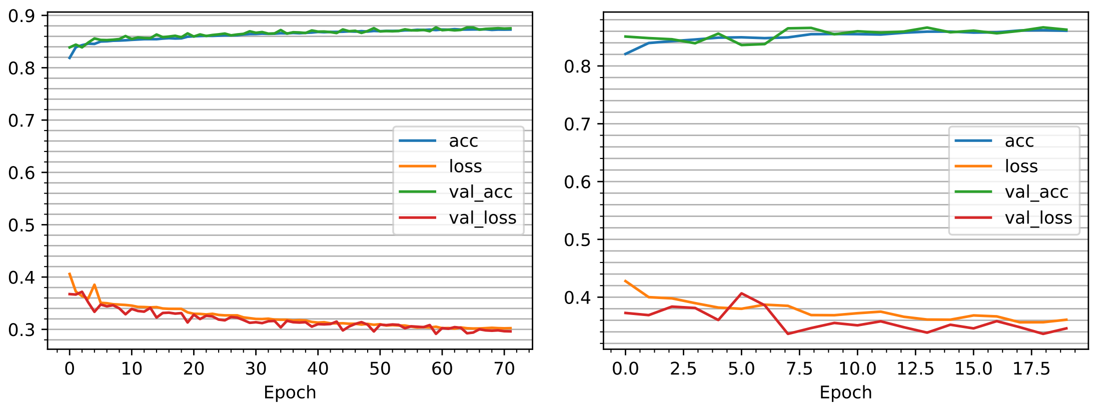
The results of all combined trainings are summarised in table 12.5. Interestingly the network using an LSTM on decay strings shows the top performance for charged trainings, though as in the decay string trainings, it suffers from very slow training times. The 1×1 convolutions again shows the most robust performance across both mixed and charged, with ResNet also showing similar results. Inspecting the training plots, the large validation metric fluctuations in early epochs from the MCParticles mixed trainings appears to occur in the combined trainings as well. I observe the same pattern, with vanilla CNN introducing the biggest fluctuations, and residual and 1×1 convolutions reducing them as before. The charged trainings show increased initial fluctuations, with all becoming stable after several iterations.

12.4.4 Training closing remarks

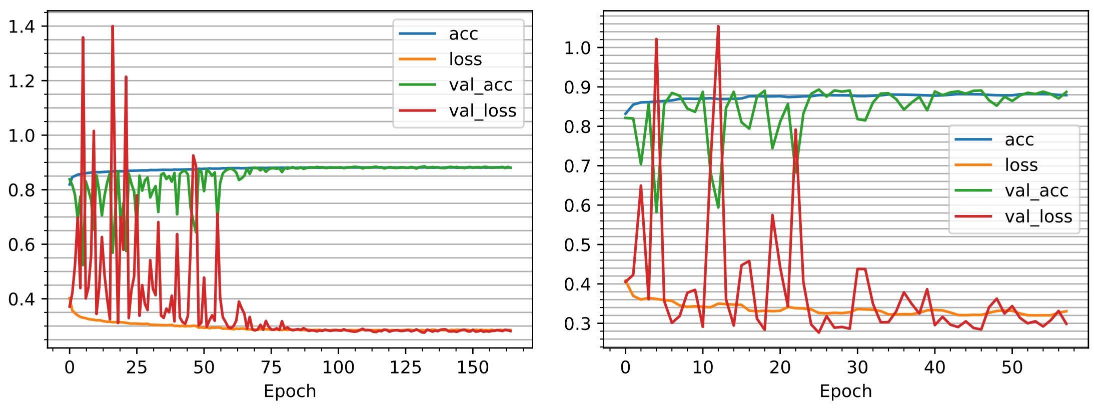
My first observation from the training results is that the combination of decay string information with the individual particle properties provides a significant improvement in classification power. Having the combination networks used in this study is therefore a warranted first step, but motivates investigation into further methods of combining and representing input data.

There are three channels through which I believe immediate improvements can be made: increasing the training data volume significantly, further augmenting the input particle properties/adding more properties (e.g. No. daughters), and implementation of more exotic or powerful architectures (e.g. graph neural networks [89], very deep CNNs [95]). Given the exceptionally large volume of simulation data that will be required as the Belle II experiment ramps up data taking over the coming year, my assessment is that the aforementioned improvement avenues offer a quick means of meeting that requirement while other more long-term solutions are developed (e.g. on-the-fly simulation).

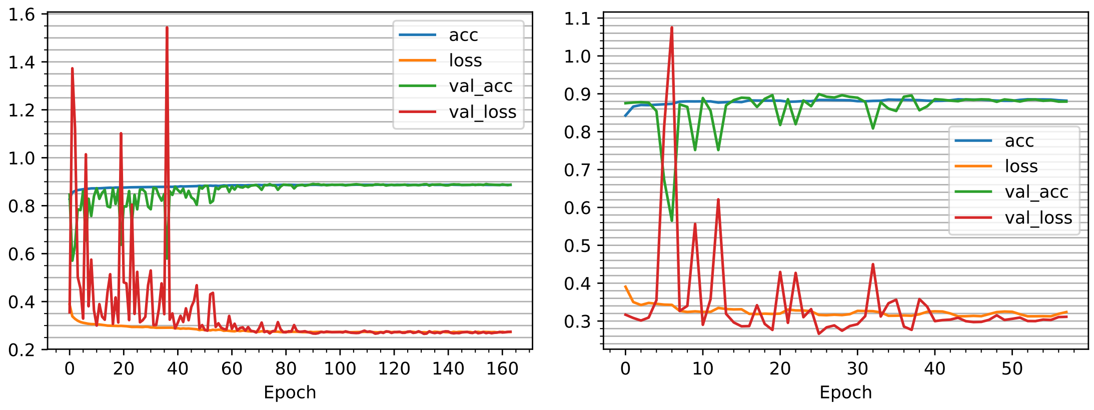
Due to the large range of possible hyperparameters to vary there is still plenty of room for the fine-tuning of the networks. The time required for each training run and limited computational power means that an exhaustive exploration of the hyperparameter space is not feasible. In the last years there have been attempts at automating the hyperparameter optimisation process, albeit for a currently limited range of applications. The two most



(a) Fully connected (MLP)

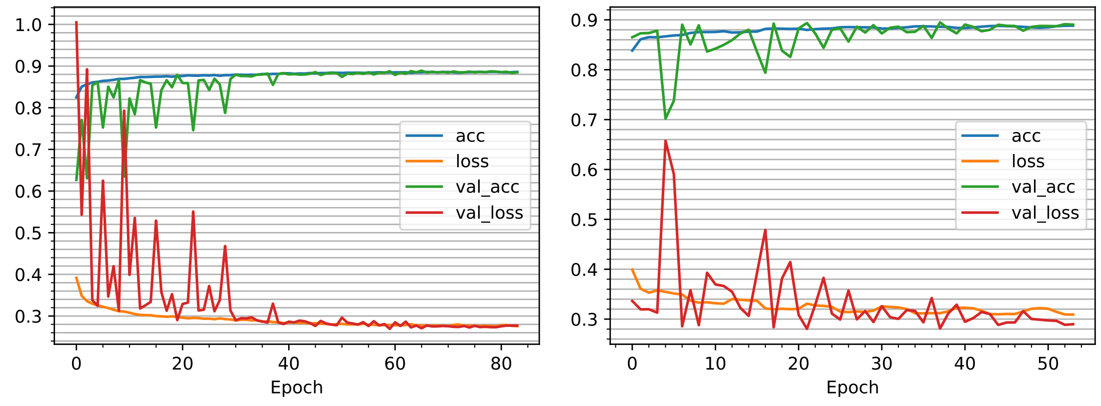


(b) Vanilla CNN (MCParticles) and wide CNN (decays string)

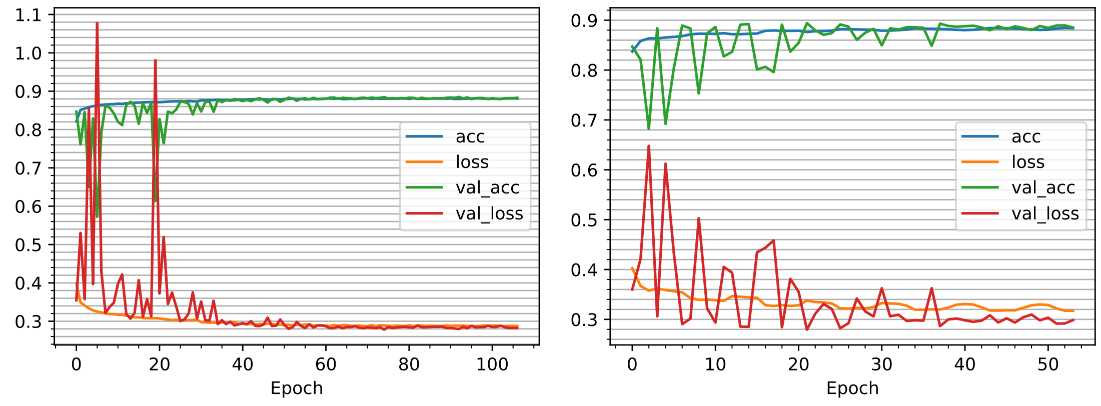


(c) Vanilla CNN (MCParticles) and LSTM (decay string)

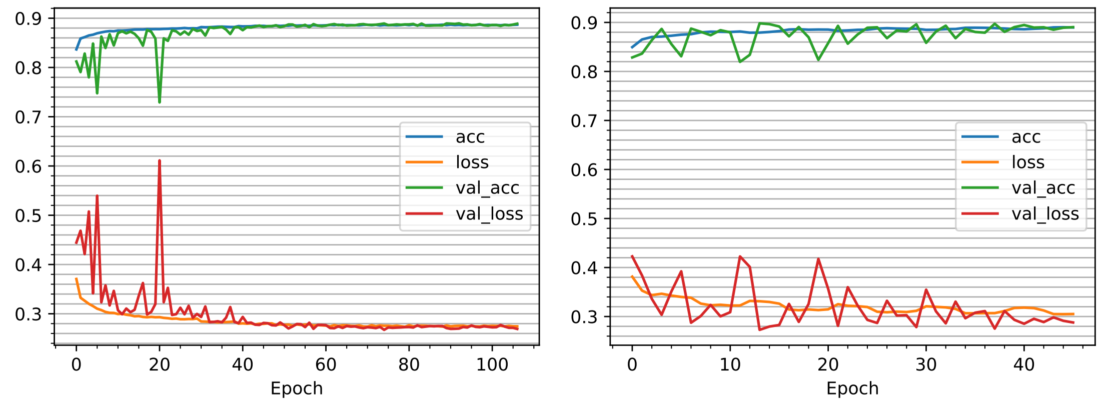
Figure 12.21: Training metrics for charged FEI B^+ (left column) and mixed FEI B^0 (right column) for the fully connected, vanilla, and LSTM based architecture combined trainings.



(a) ResNet



(b) ResNeXt



(c) 1×1

Figure 12.22: Training metrics for charged FEI B^{+} (left column) and mixed FEI B^{0} (right column) for the ResNet, ResNeXt, and 1×1 convolutional architecture combined trainings.

Table 12.5: Optimised final architectures for combined trainings final sub-networks. MCParticle and decay string sub-networks use the optimised architectures from sections 12.4.1 and 12.4.2. The input to the initial layers shown here are the concatenated outputs of the previous sub-networks. In all models the architecture for charged FEI B^+ and mixed B^0 is reused, with the validation loss values displayed as B^+/B^0.

Model	MLP	Vanilla CNN	ResNet	ResNeXt	1×1 CNN	Vanilla-LSTM

Layers	Dense(256)
	Dense(256)
	Dense(256)
		Dense(512)
		Dense(512)
		Dense(512)
			Dense(256)
			Dense(128)
				Dense(128)
				Dense(64)
				Dense(32)
					Dense(128)
					Dense(64)
					Dense(32)
						Dense(128)
						Dense(64)
						Dense(32)

Val. loss (B^+/B^0)	0.291/0.346	0.276/0.299	0.269/0.290	0.278/0.299	0.269/0.288	0.265/0.311

notable are Google's AutoML [96] and Auto-Keras [97], the latter of which is most relevant to this study[2]. This technology is worth monitoring for future use in the continuation of this study.

12.5 Evaluation

Following the results of the trainings in the previous section, I now demonstrate their application in event simulation and present a means of evaluating their selection performance. The goal here is two-fold: firstly to check that the trainings do provide a speedup in the simulation of backgrounds, and secondly to provide the tools necessary for identifying any biases introduced by the networks.

To perform the evaluation I selected two trainings from the combined trainings (section 12.4.3), one for charged and one for mixed generic backgrounds. I select the ResNet trained networks as they were one of the best performing. The 1×1 CNN would also be a valid choice as it performed excellently during training. The vanilla-LSTM training was not selected due to the slow processing time required by the LSTM sub-network for decay strings.

To apply the trained networks I created a custom basf2 module to insert into the simulation data flow as shown in figure 12.3. This module reads the output of the MC stage and applies the trained network to return a prediction of the pass likelihood. All of the preprocessing described in section 12.3 is performed on the fly before features are passed to the network. For the purposes of evaluation I keep all events, regardless of their predicted pass likelihood. The purpose of this is to allow inspection of the impact of different likelihood thresholds on the pass events. The execution time of the module is difficult to compare to the values shown in table 10.1, as the HEP-SPEC06 benchmark software is proprietary. I can, however, compare the execution over many simulated events to provide an estimate of the relative processing times required. Table 12.6 shows the mean execution times per module call ($\pm$ one standard deviation) for the mixed channel across twenty individual simulation runs with 5000 events simulated in each[3]. The *NN module* indicates the time required by the basf2 module described above to perform both feature preprocessing and inference. The times are recorded by basf2 for all 5000 events and include the initialisation of each module which is performed prior to the first event. The initialisation of the neural network includes loading of the pre-trained network and the Tensorflow/Keras libraries which causes the large standard deviation seen. Given that $num1.2e5$ events are typically simulated per job in the commissioned Monte Carlo simulation campaigns, I expect the overall impact of the initialisation time to be negligible.

[2] Auto-Keras is open source and free, whereas Google's AutoML is not.

[3] Execution time summaries are given by basf2 after all processing is finished, not recorded manually by me. To separate initialisation times from execution times requires changes to basf2, but would allow a more precise measure.

Table 12.6: Simulation execution times with trained neural network inference included. Times are given as average execution time per call (event) with one standard deviation.

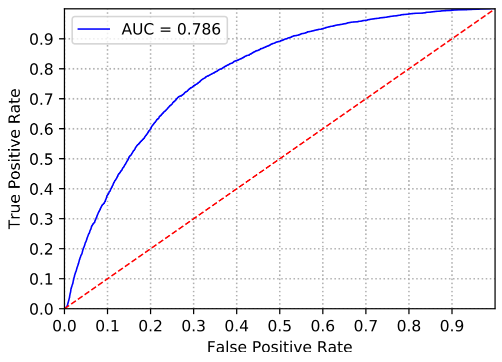
Stage	Time (ms) / Call
Event generation	5 ± 26
NN module	39 ± 201
Detector simulation	1970 ± 720
Event reconstruction	7000 ± 3000

Once the events have been simulated I repeat the FEI skim procedure performed in section 6.2.1 on all events, regardless of the neural network output. This allows an inspection of the performance of the network (how accurately it rejected the true pass and fail events) at varying thresholds applied to the pass prediction output. To demonstrate the performance I compare the true positive (correctly classified pass events) and false positive (incorrectly classified fail events) rates at varying thresholds. Figure 12.23 shows the resulting receiver operating characteristic (ROC) curves [98] for both trainings applied. The area under curve (AUC) is given to provide a quantitative means of comparing the network performances. The charged training has an AUC of 0.786 while the mixed has an AUC of 0.795, indicating similar simulation speedup provided by both. To quantify the simulation speedup, by inspecting the ROC curves and arbitrarily selecting a pass event retention rate of 80%, a fail event rejection rate of around 65% is achieved. Since this is a binary classification task (only two event options: pass or fail), and neglecting the execution time of the network needed to make the classification, this translates directly into a 65% simulation time reduction in order for the production of 80% of the background events which will survive the FEI skim procedure. While this is not an amazing jump in background simulation volumes accessible, as is required by those studying rare or low efficiency B decay channels, it is only a first attempt which can reasonably be expected to be improved as more advanced machine learning techniques are explored.

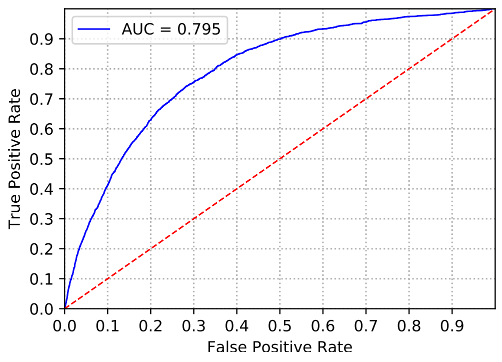
The natural question following the performance seen here is: what are those misclassified pass events (false negatives)[4]? The biggest concern is that they represent a specific part of the background phase space; either some difficult to classify kinematic regions or certain decay channels which the network tends towards rejecting. There are various potential solutions to combating any of these biases should they be found, of which the following three I have listed in order of preference:

1. Improve the classification performance to the point at which such biases disappear, i.e. near perfect classification. This is would require significant increases in the training data volumes used and may require training on specific analyses with clear kinematic

[4] I am not concerned with the details of the fail events outside of how well rejected they since they will never enter into physics analysis and introduce potential biases. Instead they will simply be rejected by the FEI skims.



(a) Charged FEI B^+



(b) Mixed FEI B^0

Figure 12.23: Receiver operating characteristic curves of applied networks during evaluation data production. The red dashed line indicates the performance of a naive untrained network (i.e. flipping a coin). The blue line shows the performance of the networks at various pass probability prediction thresholds. The area under curve (AUC) score is shown in the legend.

restrictions in place on signal candidates.

2. Construct anti-bias measures as a property of the network training. This is implemented in practice in the loss function. One common method is the inclusion of a divergence term in the loss that is a measure of how correlated the network output is with the variables being monitored for bias.

3. Weight the under-performing regions of phase space according to a baseline Monte Carlo sample that did not involve the use of the classifier. This can be performed in bins of the network output. I investigate this method in this section.

4. Identify regions of the simulation phase space which the classifier should ignore. For example if it is found that all events containing three or less charged particles significantly under-performs, while the remainder of pass events are rejected consistently, then it may be acceptable to simply not apply the classifier to such events and perform the full simulation for all of them. This results in an obvious loss in simulation efficiency but in certain cases may be the most efficient way forward which reduces bias.

To assess potential bias I inspect the changes to the kinematics of the FEI skim output. I select a moderate threshold for both channels of pass probability greater than 0.85 for the charged B^+ and 0.60 for mixed B^0. These are selected such that the pass event retention rates are roughly 78% for charged and 74% for mixed, and the corresponding fail event rejection rates are 65% and 72%. I then choose variables which describe the reconstructed FEI B meson, in particular those used typically in their selection (recall section 6.2.1): M_{bc}, $\Delta(E)$, sigProb, and nTracks. Additionally I select some rest of event (ROE) variables to represent the kinematics of what remains in the detector aside from the reconstructed B meson, namely the thrust vectors [47], and the ratio of second to zeroth Fox-Wolfram moments [46], R2, from section 6.2.4 These can help indicate any potential biases introduced in the remaining particles that would be used to reconstruct the signal decay in an analysis.

At this point it is useful to clarify several definitions before continuing:

NN This is the trained neural network applied to simulated events as shown in figure 12.3.

P (F) Pass (Fail) events.
These are events which are kept (discarded) by the FEI skim.

P^* (F^*) Pass* (Fail*) events.
Events predicted by the NN to be pass (fail) events.

TP^* True pass* events.
The set of pass events correctly predicted to pass by the NN, i.e. events which would make it through the entire simulation flow from figure 12.3.

FF^* False fail* events.
These are pass events incorrectly classified as F^* by the NN. A perfect NN would produce no false fail* events.

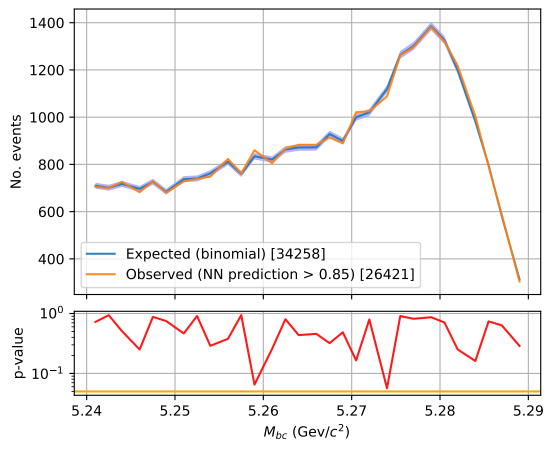
To measure the introduced bias I compare the distributions of all pass (P) events to that of the true pass* (TP^*). As the TP^* events are a subset of the P, I use a binomial distribution of the P events to quantify the agreement between the TP^* from the NN and expected TP^* from an unbiased classifier. I bin each observable, where each bin's P binomial distribution has a mean of

$$\mu_{P,i} = \epsilon N_{P,i} \,, \tag{12.2}$$

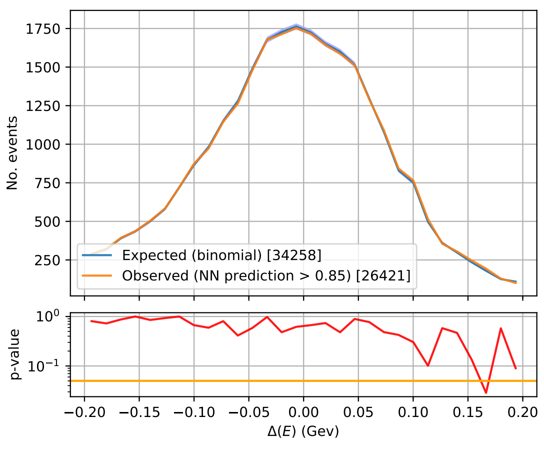
where ϵ is the total TP^* efficiency of the NN (78% for charged and 74% for mixed), and $N_{P,i}$ is the total number of P events in that bin. I calculate standard deviation as

$$\sigma_{P,i} = N_{P,i}\epsilon(1-\epsilon) \,. \tag{12.3}$$

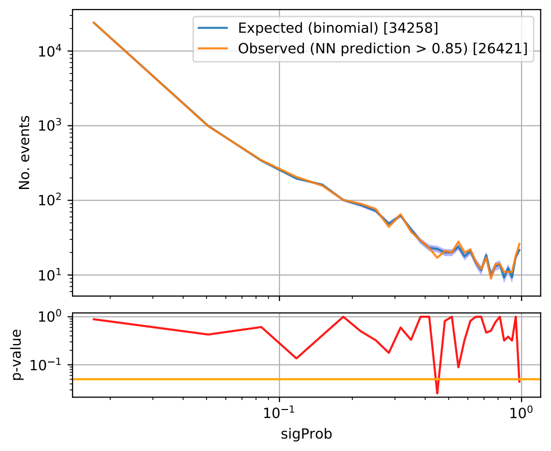
I compare the binomial distribution of P with the each bin's recorded TP^* and calculate the bin-wise p-value to check for agreement. Figures 12.24 and 12.25 show the resulting plots for the charged FEI B^+ channel. Blue shows the binomial distributions of P with the shaded region indicating one standard deviation. Orange shows the TP^* distribution. The accompanying red plot underneath each shows the p-value, with the orange line indicating a p-value of 0.05 for reference. Note the log scale on the y-axis of the p-value plots and both axes of the sigProb distributions. The corresponding plots for the mixed FEI B^0 channel are



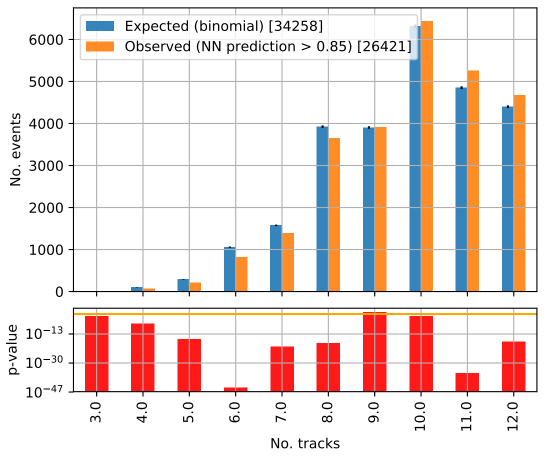
(a) M_{bc}



(b) $\Delta(E)$



(c) sigProb (log axes)

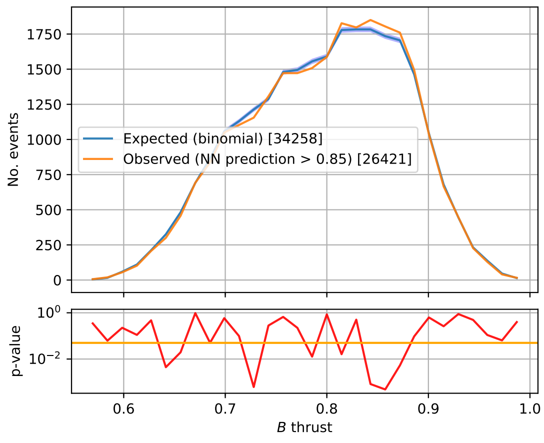


(d) nTracks

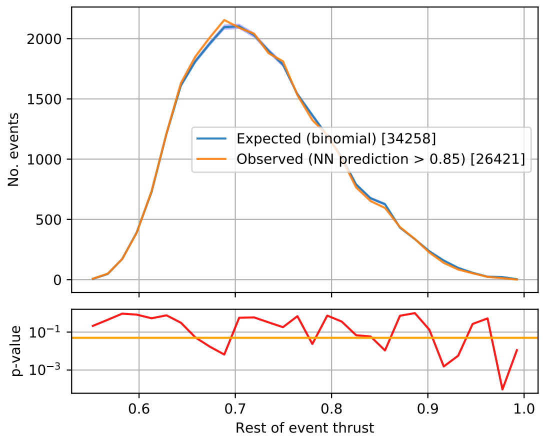
Figure 12.24: Bias evaluation plots for charged FEI B^+.

shown in figs. G.1 and G.2 in Appendix G. Inspecting these distributions, there is a significant bias introduced in the number of tracks seen in figure 12.28d with an under-representation of events with fewer tracks and over-representation of higher multiplicity events. A similar bias is seen in the mixed B^0 channel. Additionally, several bins in the ROE kinematic distributions show disagreement. Overall the resulting TP^* distributions indicate biases and motivate investigating corrections.

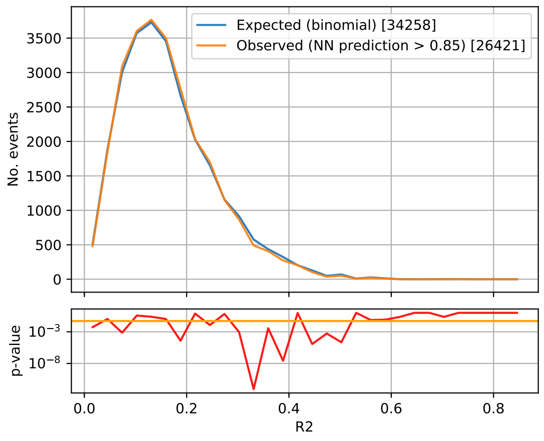
In an attempt to curb this bias I introduce a second neural network that attempts to discriminate between TP^* and FF^* events. The output of the network can then be used to weight simulated as a means of recovering the expected kinematic distributions. This second network I then refer to as the *biasNN* to distinguish it from the original *NN*. The *biasNN*



(a) B meson thrust



(b) ROE thrust



(c) R2

Figure 12.25: Bias evaluation plots for charged FEI B^+.

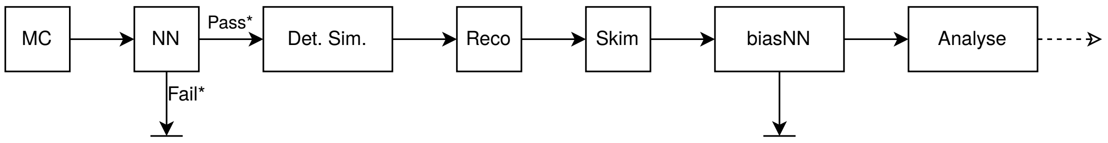


Figure 12.26: Monte Carlo simulation flow with both the selective background module and the bias correction neural network inserted.

predicted TP^* and FF^* I refer to as *bias*TP^* and *bias*FF^* respectively. Figure 12.26 shows an example of where the *biasNN* could fit into the processing flow. The network uses the variables shown in figs. 12.24 and 12.25 as inputs, along with two additional variables: the cosine of the difference between the thrust of the FEI reconstructed B meson and the beam-pipe ($\cos(B_{\text{thrust}} - z)$) and the ROE thrust ($\cos(B_{\text{thrust}} - \text{ROE}_{\text{thrust}})$). The two additional inputs were added as they improved the quality of discrimination of the network. Both are shown for the mixed and charged FEI channels in figure G.3 in Appendix G.

The network architecture used is a simple feed forward network with three hidden dense layers containing 64 neurons each. Each layer uses LeakyReLU activation, except the last which uses sigmoid, and batch normalisation. I used the Adam optimiser for training with a learning rate of 5×10^{-4}, AMSGrad, and a binary cross-entropy loss as this showed the best performance. I selected the charged FEI B^+ channel for training and evaluating the network as a test-bed for this process of bias correction. As the classification and weighting must ultimately be performed on an event basis, not on individual B candidates, I chose the reconstructed B with the highest sigProb of each event for classification. The training dataset used contains roughly 8.6×10^4 candidates, with an additional 2×10^4 candidates used for validation. An alternative approach could be to use all B meson candidates and average across the *biasNN* output to obtain a *bias*TP^* for the whole event, or to use event-level variables (number of charged tracks, ECL clusters, etc.) to make the bias prediction before FEI is performed. These alternatives are not investigated in this study and left for future work.

The resulting normalised distribution of *bias*TP^* predictions from the trained network are shown in figure 12.27, where the blue shows the subset of FF^* events, and orange the TP^*. The final accuracy of the *biasNN* is around 58% (loss of 0.67). This level of performance is reasonable given the levels of bias seen in most kinematic bins in the input variables (the number of tracks alone is not enough to decide whether a single event is likely to contribute to biases).

I applied the trained network to another independent sample of roughly 2.8×10^4 charged FEI B^+ events and calculated the ratio $\frac{P}{TP^*}$ for each bin in *bias*TP^*. The goal being to recover the original P distribution, represented by the sum of FF^* and TP^* in figure 12.27. Table 12.7 shows the corresponding bin ratios, normalised to the range $[0, 1]$. I then used the ratios as a sampling weight for each bin, e.g. for a ratio of 0.75 only a random set of 75% of events from that bin are retained. Figures 12.28 and 12.29 show the resulting corrected distributions.

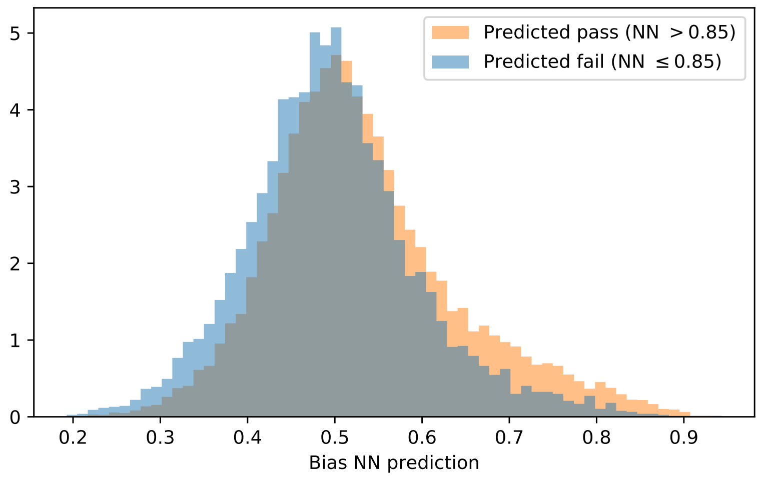
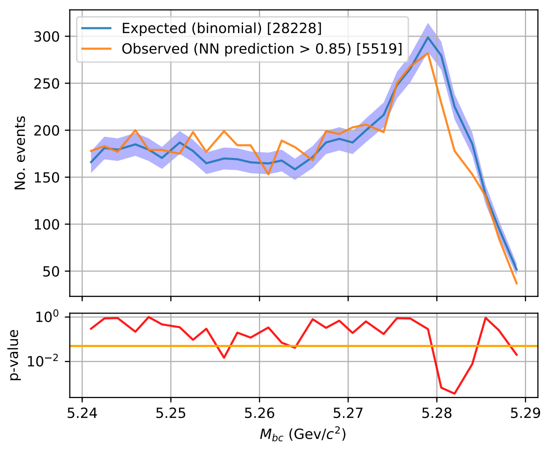


Figure 12.27: Normalised output predictions of the biasNN network on charge FEI B^+ skim events.

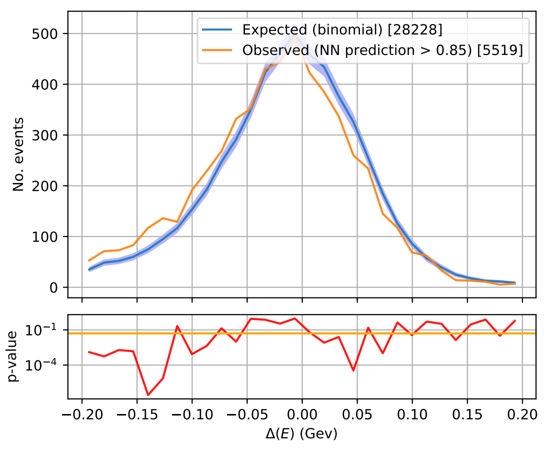
Note that the mean of the bin ratios is only 0.295, resulting in a very low final efficiency and negating much of the speedup provided by the original *NN*. Interestingly the correction significantly improved the resulting distribution of number of tracks. Unfortunately this is to the detriment of the other kinematic variables, although with the resulting low statistics it is difficult to quantify precisely. Overall it appears as though this form of post hoc bias correction is not a suitable solution. Even in the best-case scenarios involving minimal corrections it involves the discarding of a portion of simulated pass events, working against the original goal of this study. Instead, including the bias correction as a property of the network training would be more appropriate. This point I discuss further in section 13.1.

Bin in *bias*TP^*	$\frac{P}{TP^*}$ ratio
$(0.192, 0.23]$	1.000
$(0.23, 0.268]$	0.756
$(0.268, 0.305]$	0.734
$(0.305, 0.343]$	0.564
$(0.343, 0.38]$	0.453
$(0.38, 0.418]$	0.379
$(0.418, 0.455]$	0.332
$(0.455, 0.493]$	0.282
$(0.493, 0.531]$	0.256
$(0.531, 0.568]$	0.225
$(0.568, 0.606]$	0.197
$(0.606, 0.643]$	0.174
$(0.643, 0.681]$	0.138
$(0.681, 0.718]$	0.098
$(0.718, 0.756]$	0.094
$(0.756, 0.793]$	0.070
$(0.793, 0.831]$	0.057
$(0.831, 0.868]$	0.026
$(0.868, 0.906]$	0.000
$(0.906, 0.944]$	0.052

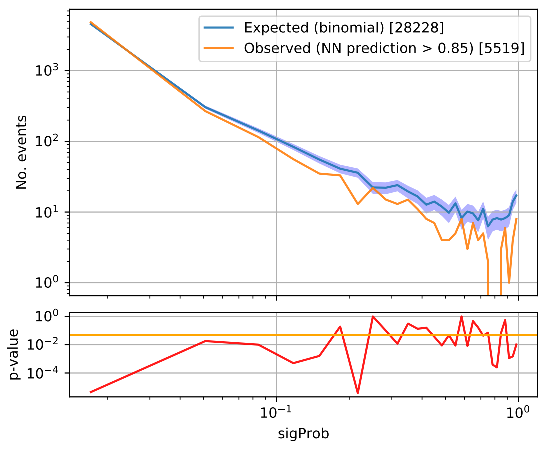
Table 12.7: Bin ratios corresponding to 12.27.



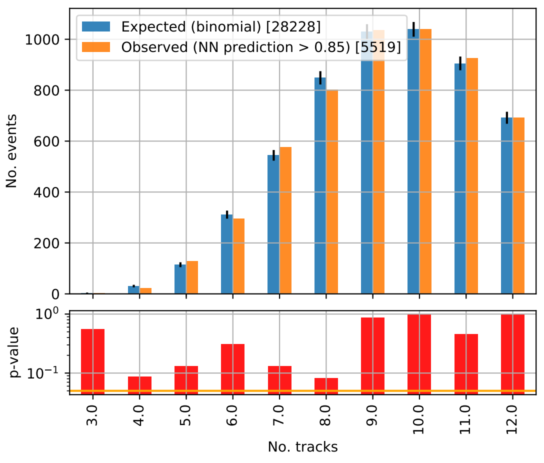
(a) M_{bc}



(b) $\Delta(E)$

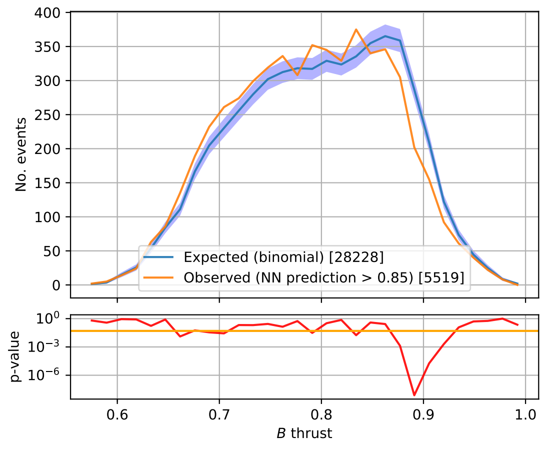


(c) sigProb (log axes)

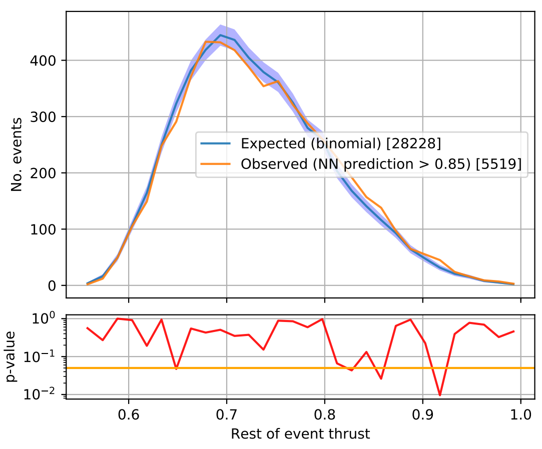


(d) nTracks

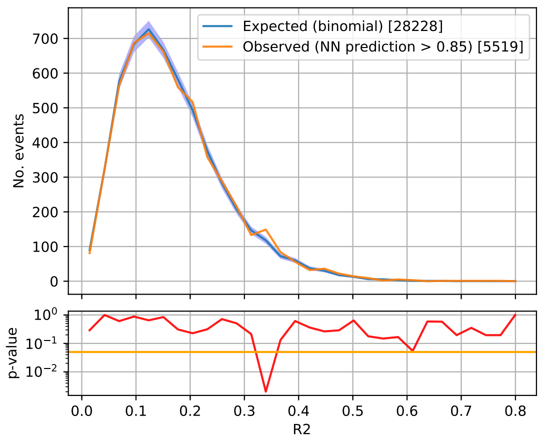
Figure 12.28: Corrected bias evaluation plots for charged FEI B^+.



(a) B meson thrust



(b) ROE thrust



(c) R2

Figure 12.29: Corrected bias evaluation plots for charged FEI B^{+}.

Chapter 13

Summary

In this study I developed a method of selective Monte Carlo simulation as a means of reducing computational requirements. With the use of neural networks I showed it is possible to predict early in the simulation procedure how likely a simulated event is to be useful in analysis. I developed the procedure for extracting information from Monte Carlo data in Belle II and performing the necessary preprocessing steps for input to a neural network. With this I investigated three different methods of using simulated particles, decays strings, and the combination of both to make predictions. The results showed that the networks with the combination input had the best predictive power. I then showed, through the simulation of an independent sample, that a trained network is able to reduce simulation time requirements. Using the output of this independent sample I outlined a method of quantifying kinematic biases introduced by the network. The biases found were non-negligible and so in the final part of this study I investigated a method of bias correction which proved unable to recover original kinematics. The results of this study are nonetheless promising and demonstrate that selective background Monte Carlo simulations targeted to specific analyses are possible. If the biases are able to be corrected then this is a powerful technique that will enable significant increases in simulation volumes for Belle II.

13.1 Outlook

In the neural network implementations used in this study, the natural graph structure of the data is never fully encapsulated. The data formats used, MCParticles equated with pixels and decay strings as sentences, effectively flatten the data structures. The networks are then left to infer the particle relations themselves, rather than having them included as an intrinsic property of the data format itself. An alternative approach would involve the use of graph neural networks [89] or graph convolutional networks [88] which are designed specifically to handle graph-like structures. Investigation of their uses in other fields such as social network analysis and molecular chemistry would allow an analogous application here that may yield a better predictive power.

The network architectures I used in this work all had architectures which required a fixed length input and necessitated padding of the input data. The padding scheme applied used zero-padding, where all padded input values were set to zero. Zero, however, is still a valid input and may inhibit the discriminating power of the networks. Fully convolutional networks [99] offer a means of handling arbitrary sized inputs with a fixed size output. This removes entirely the need for padding or truncating data during preprocessing.

The ideal handling of the kinematic biases discussed in section 12.5 would be to include their mitigation in the training of the initial neural network. One method of achieving this would be to introduce a penalty to the training loss. The penalty should be a measure of the correlation between the network output and a collection of selected kinematic variables deemed to be representative of the entire simulated event. For example, including the Kullback-Leibler divergence [100] in the loss in a similar manner to its usage in variational autoencoders would allow a controlled penalisation of induced bias.

Hyperparameter optimisation of neural networks is a time-consuming task with no guarantee of yielding better results. To automate this process when using Keras the Auto-Keras [97] library could be used. This will remove the need for supervision of the training procedure, creating extra time for other tasks and speeding up the entire investigative process in general.

The normalisation scheme I used in this study was min-max scaling which performs a linear rescaling of continuous variables to the range $[0, 1]$. An alternative normalisation using for example tanh may be better suited to the distribution of values in MCParticles. As a majority of the simulated particles are produced and decay very close to the interaction point of the Belle II detector differences between small values should be emphasised.

Glossary

$E_{\mathbf{ECL}}$ The unreconstructed energy remaining in the electromagnetic calorimeter in GeV. Defined as $E_{\text{ECL}} = E_{\text{obs}} - E_{\text{rec}}$, where E_{obs} is the total energy observed, and E_{rec} is the total energy used in particle candidate reconstruction. . 38, 39, 51, 55–60, 64, 66, 74–76, 164–166, 168, 169

$E_{\mathbf{miss}}$ Missing energy of the $\Upsilon(4S)$ in the event centre of mass frame. Defined as the difference between initial $\Upsilon(4S)$ energy and the sum of the energy of all reconstructed daughters. . 50, 162

K**daughter angle** Cosine of the angle between the daughter particles of the signal side Kaon. Expected to peak at 1 for correctly reconstructed Kaons. . 49, 50, 162

$M_{\mathbf{bc}}$ Beam-constrained mass in GeV/c^2. Defined as $M_{\text{bc}} = \sqrt{E^2_{\text{beam}} - p^2_B}$, where E_{beam} is half of the initial energy of the e^+e^- collision in the centre of mass frame, and p_B the three-momentum of the reconstructed B meson. . 35, 50, 51, 75, 104, 135, 137, 142, 157, 159, 161, 162, 180

$P_{\mathbf{miss}}$ Missing magnitude of the three-momentum of the reconstructed $\Upsilon(4S)$. Defined as the difference between the initial and reconstructed $\Upsilon(4S)$ momentum. In the event centre of mass frame this is trivially the reconstructed momentum of the $\Upsilon(4S)$. . 50, 162

$\Delta(E)$ Beam-constrained energy in GeV. Defined as $\Delta(E) = E_B - E_{\text{beam}}$, where E_{beam} is half of the initial energy of the e^+e^- collision in the centre of mass frame, and E_B the reconstructed B meson energy. . 35, 50, 104, 135, 137, 142, 180

cos($B_{\mathbf{thrust}}$ − $\mathbf{ROE}_{\mathbf{thrust}}$) Cosine of the angle between the thrust of the given B meson and the thrust of the rest of event. . 139, 182

cos($B_{\mathbf{thrust}} - z$) Cosine of the angle between the thrust of the given B meson and the z-axis (beam-pipe). . 139, 182

ROE(E) Rest of event energy in GeV. The energy remaining in the detector once the energy associated with the reconstructed particle is removed. Restrictions are places on the tracks and ECL hits included in the measurement to remove likely contributions to the total energy from beam-background related sources. . 49, 162

$fBDT$ Fast Boosted Decision Tree (fBDT) calculated probability of an event originating from continuum or not. 0 indicates that the input was likely a continuum event, 1 a

$\Upsilon(4S) \rightarrow B\bar{B}$ event. . 50, 51, 162

$p_{\textbf{CMS}}$ Momentum in GeV/c of the signal side kaon in the centre of mass reference frame. . 49, 162

array index Unique, per event index number of the Monte Carlo simulated particle (zero-based). . 108–111

charge Electric charge of the particle.. 108, 109, 111

decay time Time stamp of the particle's decay in ns relative to $\Upsilon(4S)$ production time.. 107, 111

decay vertex Coordinates of the particle's decay vertex.. 107, 111

dr Transverse (radial) distance of closest approach of the reconstructed track to the interaction point in cm. . 38

dz Distance of closest approach of the reconstructed track to the interaction point in the z-axis (along the beam-pipe) in cm. . 38

E Reconstructed energy of the particle in GeV. . 38–40, 50, 162

E1E9 The ratio of energies for a given ECL cluster of the central crystal and 3x3 crystals around the central crystal. . 39

ECL cluster error timing ECL cluster timing uncertainty range that contains 99% of true photons. . 38–41

ECL cluster timing Difference in nano-seconds between the bunch crossing (e^+e^- collision) and activation of the highest energy crystal for the given particle's ECl hit. Photons from beam-related backgrounds are not associated with the bunch crossing and are expected to show a uniform distribution. . 38–42

energy Particle energy in GeV.. 108, 111

FSP Final state particles. These are the eventual decay daughters of the initial e^+e^- collision that have a lifetime long enough to be detected by Belle II. This is a catch-all term to describe all the particles that can be combined in various ways to reconstruct the particular decay being searched for. . 34, 107

lifetime Difference between decay and production time of the particle.. 111

mass Particle mass in GeV.. 111

MDST Mini-DST (data summary tapes). The event storage format used in Belle II for direct user analysis. Combines the online reconstruction of raw subdetetector output with calibration constants. . 81, 108

momentum Three-momentum of the particle in Gev/c.. 108, 109, 111

mother PDG code PDG code of the particle's parent.. 108, 111, 121, 127

No. daughters Number of decay daughter particles.. 108, 109, 111, 129

nTracks Total number of charged tracks in the event. When working with simulations this is the true number of tracks in the event. For reconstructed particles it is only the number of tracks reconstructed from detector hits. . 35, 104, 136, 137, 142, 180

PDG code Unique identifier for that particle type and charge. [9] contains the rules necessary for translating each code to it's particle type.. 98, 108–112, 121, 127

Production time Production time in ns relative to $\Upsilon(4S)$ production.. 108, 111

production vertex Coordinates of the particle's production vertex.. 108, 109, 111

sigProb Signal probability output by the Full Event Interpretation software to indicate the confidence that the associated B meson was correctly reconstructed. 0 indicates that the B meson was not correctly reconstructed, 1 means a high level of confidence that it was. . 30, 35, 50, 51, 104, 136, 137, 139, 142, 162, 180

status bit Bitmask representing how the particle was created during simulation, e.g. by EvtGen, whether it is a virtual particle, whether it originates from initial or final state radiation. . 80, 108, 111

Bibliography

[1] T. Kuhr, C. Pulvermacher, M. Ritter, T. Hauth and N. Braun, 'The Belle II Core Software', Comput. Softw. Big Sci. **3**, 1 (2019).

[2] J. Grygier et al., 'Search for $B \to h\nu\bar{\nu}$ decays with semileptonic tagging at Belle', Phys. Rev. **D96**, 091101 (2017).

[3] R. Aaij et al., 'Test of lepton universality using $B^+ \to K^+\ell^+\ell^-$ decays', Phys. Rev. Lett. **113**, 151601 (2014).

[4] R. Aaij et al., 'Test of lepton universality with $B^0 \to K^{*0}\ell^+\ell^-$ decays', JHEP **08**, 055 (2017).

[5] A. J. Buras, J. Girrbach-Noe, C. Niehoff and D. M. Straub, '$B \to K^{(*)}\nu\bar{\nu}$ decays in the Standard Model and beyond', JHEP **02**, 184 (2015).

[6] S. Pohl, 'Track Reconstruction at the First Level Trigger of the Belle II Experiment', PhD thesis (Munich, Ludwig-Maximilians-Universität, Munich, Apr. 2018).

[7] N. Cabibbo, 'Unitary symmetry and leptonic decays', Phys. Rev. Lett. **10**, 531–533 (1963).

[8] M. Kobayashi and T. Maskawa, 'CP Violation in the Renormalizable Theory of Weak Interaction', Prog. Theor. Phys. **49**, 652–657 (1973).

[9] M. Tanabashi et al., 'Review of Particle Physics', Phys. Rev. **D98**, 030001 (2018).

[10] L. Wolfenstein, 'Parametrization of the kobayashi-maskawa matrix', Phys. Rev. Lett. **51**, 1945–1947 (1983).

[11] S. L. Glashow, J. Iliopoulos and L. Maiani, 'Weak Interactions with Lepton-Hadron Symmetry', Phys. Rev. **D2**, 1285–1292 (1970).

[12] T. Inami and C. S. Lim, 'Effects of Superheavy Quarks and Leptons in Low-Energy Weak Processes $K_L \to \mu\mu$, $K^+ \to \pi^+\nu\nu$ and $K^0 \leftrightarrow \bar{K}^0$', Progress of Theoretical Physics **65**, 297–314 (1981).

[13] G. Buchalla and A. J. Buras, 'QCD corrections to rare K and B decays for arbitrary top quark mass', Nucl. Phys. **B400**, 225–239 (1993).

[14] M. Misiak and J. Urban, 'QCD corrections to FCNC decays mediated by Z penguins and W boxes', Phys. Lett. **B451**, 161–169 (1999).

[15] G. Buchalla and A. J. Buras, 'The rare decays $K \to \pi\nu\bar{\nu}$, $B \to X\nu\bar{\nu}$ and $B \to \ell^+\ell^-$: An Update', Nucl. Phys. **B548**, 309–327 (1999).

[16] J. Brod, M. Gorbahn and E. Stamou, 'Two-Loop Electroweak Corrections for the $K \to \pi\nu\bar{\nu}$ Decays', Phys. Rev. **D83**, 034030 (2011).

[17] D. M. Straub, '$B \to K^{(*)}\nu\bar{\nu}$ – SM predictions', in Belle2-memo-2016 (Dec. 2015).

[18] W. Altmannshofer, A. J. Buras, D. M. Straub and M. Wick, 'New strategies for New Physics search in $B \to K^*\nu\bar{\nu}$, $B \to K\nu\bar{\nu}$ and $B \to X_s\nu\bar{\nu}$ decays', JHEP **04**, 022 (2009).

[19] D. M. Straub, 'flavio: a Python package for flavour and precision phenomenology in the Standard Model and beyond', arXiv e-prints, arXiv:1810.08132 (2018).

[20] S. Cunliffe, 'Prospects for rare B decays at Belle II', in Proceedings, Meeting of the APS Division of Particles and Fields (DPF 2017): Fermilab, Batavia, Illinois, USA, July 31 - August 4, 2017 (2017).

[21] M. Feindt, F. Keller, M. Kreps, T. Kuhr, S. Neubauer, D. Zander and A. Zupanc, 'A Hierarchical NeuroBayes-based Algorithm for Full Reconstruction of B Mesons at B Factories', Nucl. Instrum. Meth. **A654**, 432–440 (2011).

[22] M. Feindt and U. Kerzel, 'The NeuroBayes neural network package', Nucl. Instrum. Meth. **A559**, 190–194 (2006).

[23] O. Lutz et al., 'Search for $B \to h^{(*)}\nu\bar{\nu}$ with the full Belle $\Upsilon(4S)$ data sample', Phys. Rev. **D87**, 111103 (2013).

[24] O. Lutz, 'Search for $B \to h^{(*)}\nu\bar{\nu}$ Decays at Belle and Development of Track Finding for Belle II', PhD thesis (KIT, Karlsruhe, 2012).

[25] J. P. Lees et al., 'Search for $B \to K^{(*)}\nu\bar{\nu}$ and invisible quarkonium decays', Phys. Rev. **D87**, 112005 (2013).

[26] P. del Amo Sanchez et al., 'Search for the Rare Decay $B \to K\nu\bar{\nu}$', Phys. Rev. **D82**, 112002 (2010).

[27] B. Aubert et al., 'Measurement of the branching fraction of $\Upsilon(4S) \to B^0\overline{B}^0$', Phys. Rev. Lett. **95**, 042001 (2005).

[28] J. F. Kamenik and C. Smith, 'FCNC portals to the dark sector', JHEP **03**, 090 (2012).

[29] S. Renner and P. Schwaller, 'A flavoured dark sector', JHEP **08**, 052 (2018).

[30] F. Sala and D. M. Straub, 'A New Light Particle in B Decays?', Phys. Lett. **B774**, 205–209 (2017).

[31] S. K. Choi et al., 'Observation of a resonance-like structure in the $\pi^{+-}\psi'$ mass distribution in exclusive $B \to Kpi^{+-}\psi'$ decays', Phys. Rev. Lett. **100**, 142001 (2008).

[32] T. Abe et al., 'Belle II Technical Design Report', arXiv e-prints, arXiv:1011.0352 (2010).

[33] P. M. Lewis et al., 'First Measurements of Beam Backgrounds at SuperKEKB', Nucl. Instrum. Meth. **A914**, 69–144 (2019).

[34] *Belle II website*, (2018) `https://belle2.org/` (visited on 05/12/2018).

[35] E. Kou et al., 'The Belle II Physics Book', arXiv e-prints, arXiv:1808.10567 (2018).

[36] J. Karancsi, 'Operational experience with the CMS pixel detector', Journal of Instrumentation **10**, C05016 (2015).

[37] Y. Takubo, 'The Pixel Detector of the ATLAS experiment for the Run2 at the Large Hadron Collider', JINST **10**, C02001 (2015).

[38] G. B. Mohanty, 'Belle II Silicon Vertex Detector', Nucl. Instrum. Meth. **A831**, 80–84 (2016).

[39] B. Wang, 'The Belle II Experiment and SuperKEKB Upgrade', J. Univ. Sci. Tech. China **46**, 617–624 (2016).

[40] A. Moll, 'The software framework of the Belle II experiment', J. Phys. Conf. Ser. **331**, 032024 (2011).

[41] T. Keck, 'FastBDT: A Speed-Optimized Multivariate Classification Algorithm for the Belle II Experiment', Comput. Softw. Big Sci. **1**, 2 (2017).

[42] T. Keck et al., 'The Full Event Interpretation – An exclusive tagging algorithm for the Belle II experiment', arXiv e-prints, arXiv:1807.08680 (2018).

[43] B. Kronenbitter et al., 'Measurement of the branching fraction of $B^+ \to \tau^+\nu_\tau$ decays with the semileptonic tagging method', Phys. Rev. **D92**, 051102 (2015).

[44] I. Adachi et al., 'Evidence for $B^- \to \tau^-\bar{\nu}_\tau$ with a Hadronic Tagging Method Using the Full Data Sample of Belle', Phys. Rev. Lett. **110**, 131801 (2013).

[45] D. M. Asner et al., 'Search for exclusive charmless hadronic B decays', Phys. Rev. **D53**, 1039–1050 (1996).

[46] A. J. Bevan et al., 'The Physics of the B Factories', Eur. Phys. J. **C74**, 3026 (2014).

[47] E. Farhi, 'Quantum chromodynamics test for jets', Phys. Rev. Lett. **39**, 1587–1588 (1977).

[48] S. S. Wilks, 'The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses', Annals Math. Statist. **9**, 60–62 (1938).

[49] J. Ocariz, 'Probability and Statistics for Particle Physicists', in Proceedings, 1st Asia-Europe-Pacific School of High-Energy Physics (AEPSHEP): Fukuoka, Japan, October 14-27, 2012 (2014), pp. 253–280.

[50] A. Wald, 'Tests of statistical hypotheses concerning several parameters when the number of observations is large', Transactions of the American Mathematical Society **54**, 426–482 (1943).

[51] T. Abe et al., 'Achievements of KEKB', Progress of Theoretical and Experimental Physics **2013**, 03A001, 03A001 (2013).

[52] D. J. Lange, 'The EvtGen particle decay simulation package', Nucl. Instrum. Meth. **A462**, 152–155 (2001).

[53] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen and P. Z. Skands, 'An Introduction to PYTHIA 8.2', Comput. Phys. Commun. **191**, 159–177 (2015).

[54] S. Agostinelli et al., 'GEANT4: A Simulation toolkit', Nucl. Instrum. Meth. **A506**, 250–303 (2003).

[55] T. Kuhr, *Software resource estimates*, tech. rep., Internal communication (Belle II, Aug. 2016).

[56] C. Aguado Sanchez, J. Bloomer, P. Buncic, L. Franco, S. Klemer and P. Mato, 'CVMFS - a file system for the CernVM virtual appliance', in Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research (2008), p. 52.

[57] W. McKinney, ' Data Structures for Statistical Computing in Python ', in Proceedings of the 9th Python in Science Conference, edited by S. van der Walt and J. Millman (2010), pp. 51–56.

[58] T. E. Oliphant, *Guide to NumPy*, 2nd (CreateSpace Independent Publishing Platform, USA, 2015).

[59] T. Kuhr, 'First production with the Belle II distributed computing system', Journal of Physics: Conference Series **513**, 032050 (2014).

[60] T. Hara, 'Computing at the Belle II experiment', Journal of Physics: Conference Series **664**, 012002 (2015).

[61] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, `http://www.deeplearningbook.org` (MIT Press, 2016).

[62] K. Hornik, 'Approximation capabilities of multilayer feedforward networks', Neural Networks **4**, 251–257 (1991).

[63] 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, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, 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 Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng, *TensorFlow: large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[64] F. Chollet et al., *Keras*, `https://keras.io`, 2015.

[65] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization', CoRR **abs/1412.6980** (2014).

[66] S. J. Reddi, S. Kale and S. Kumar, 'On the Convergence of Adam and Beyond', in International conference on learning representations (2018).

[67] M. D. Zeiler, 'ADADELTA: an adaptive learning rate method', CoRR **abs/1212.5701** (2012).

[68] J. Duchi, E. Hazan and Y. Singer, 'Adaptive subgradient methods for online learning and stochastic optimization', J. Mach. Learn. Res. **12**, 2121–2159 (2011).

[69] M. A. Nielsen, *Neural networks and deep learning*, misc, 2018.

[70] Y. LeCun, L. Bottou and P. Haffner, 'Gradient-Based Learning Applied to Document Recognition', in (1998).

[71] A. Karpathy, *Stanford University CS231n: Convolutional Neural Networks for Visual Recognition*.

[72] K. He, X. Zhang, S. Ren and J. Sun, 'Deep Residual Learning for Image Recognition', arXiv e-prints, arXiv:1512.03385 (2015).

[73] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, 'Aggregated Residual Transformations for Deep Neural Networks', arXiv e-prints, arXiv:1611.05431 (2016).

[74] M. Lin, Q. Chen and S. Yan, 'Network In Network', arXiv e-prints, arXiv:1312.4400 (2013).

[75] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, 'Going Deeper with Convolutions', arXiv e-prints, arXiv:1409.4842 (2014).

[76] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev and J. Yosinski, 'An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution', arXiv e-prints, arXiv:1807.03247 (2018).

[77] A. Trask, F. Hill, S. Reed, J. Rae, C. Dyer and P. Blunsom, 'Neural Arithmetic Logic Units', arXiv e-prints, arXiv:1808.00508 (2018).

[78] S. Hochreiter and J. Schmidhuber, 'Long short-term memory', Neural computation **9**, 1735–80 (1997).

[79] C. Olah, *Understanding lstm networks*, `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, Accessed: 2018-11-26, Aug. 2018.

[80] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford and A. Smola, 'Feature Hashing for Large Scale Multitask Learning', ArXiv e-prints, arXiv:0902.2206 (2009).

[81] T. Mikolov, K. Chen, G. Corrado and J. Dean, 'Efficient Estimation of Word Representations in Vector Space', ArXiv e-prints, arXiv:1301.3781 (2013).

[82] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: a simple way to prevent neural networks from overfitting', Journal of Machine Learning Research **15**, 1929–1958 (2014).

[83] S. Ioffe and C. Szegedy, 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift', ArXiv e-prints, arXiv:1502.03167 (2015).

[84] A. Krogh and J. A. Hertz, 'A simple weight decay can improve generalization', in *Advances in neural information processing systems 4*, edited by J. E. Moody, S. J. Hanson and R. P. Lippmann (Morgan-Kaufmann, 1992), pp. 950–957.

[85] M. Ritter, L. Wood, T. Kuhr, M. Bracko, T. Elsethagen, K. Fox, J. Hall, C. Pulvermacher, B. Raju, M. Schram and E. Stephan, 'Belle II conditions database', Journal of Physics: Conference Series **1085**, 032032 (2018).

[86] P. Frasconi, M. Gori and A. Sperduti, 'A general framework for adaptive processing of data structures', IEEE Transactions on Neural Networks **9**, 768–786 (1998).

[87] C. Goller and A. Kuchler, 'Learning task-dependent distributed representations by backpropagation through structure', in In proc. of the icnn-96 (July 1996), 347–352 vol.1.

[88] T. N. Kipf and M. Welling, 'Semi-Supervised Classification with Graph Convolutional Networks', arXiv e-prints, arXiv:1609.02907 (2016).

[89] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, 'The graph neural network model', Undetermined, IEEE Transactions on Neural Networks **20**, 61–80 (2009).

[90] T. Sainath, O. Vinyals, A. Senior and H. Sak, 'Convolutional, long short-term memory, fully connected deep neural networks', in (Apr. 2015), pp. 4580–4584.

[91] M. Zain Amin and N. Nadeem, 'Convolutional Neural Network: Text Classification Model for Open Domain Question Answering System', ArXiv e-prints, arXiv:1809.02479 (2018).

[92] Y. Kim, 'Convolutional Neural Networks for Sentence Classification', ArXiv e-prints, arXiv:1408.5882 (2014).

[93] Y. Zhang and B. Wallace, 'A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification', ArXiv e-prints, arXiv:1510.03820 (2015).

[94] K. He, X. Zhang, S. Ren and J. Sun, 'Identity Mappings in Deep Residual Networks', arXiv e-prints, arXiv:1603.05027 (2016).

[95] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition', CoRR **abs/1409.1556** (2014).

[96] B. Zoph and Q. V. Le, 'Neural Architecture Search with Reinforcement Learning', arXiv e-prints, arXiv:1611.01578 (2016).

[97] H. Jin, Q. Song and X. Hu, *Auto-keras: efficient neural architecture search with network morphism*, (27th June 2018)

[98] T. Fawcett, 'An introduction to roc analysis', in Proc. natl. acad (2006), pp. 10–1016.

[99] J. Long, E. Shelhamer and T. Darrell, 'Fully convolutional networks for semantic segmentation', CoRR **abs/1411.4038** (2014).

[100] S. Kullback and R. A. Leibler, 'On information and sufficiency', Ann. Math. Statist. **22**, 79–86 (1951).

Appendix A

Cut Selection Variables

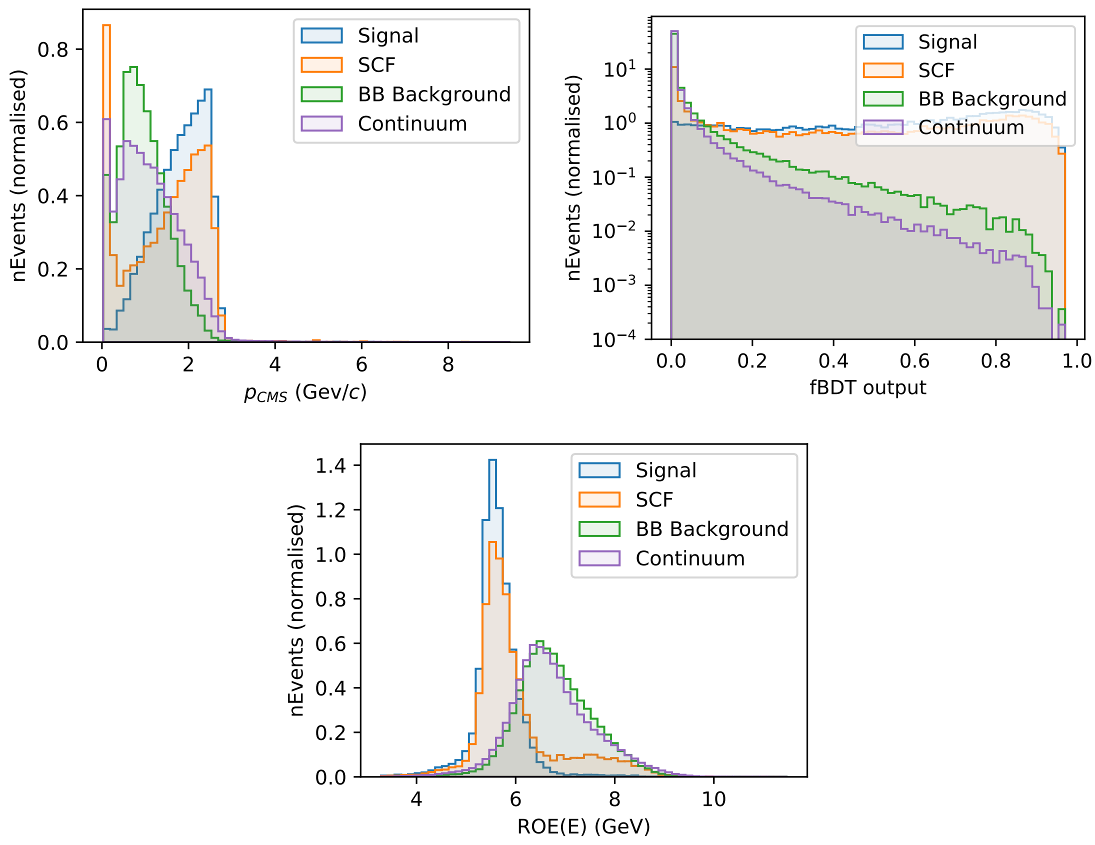


Figure A.1: K^+ hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots have been normalised to show relative number of events across the full range.

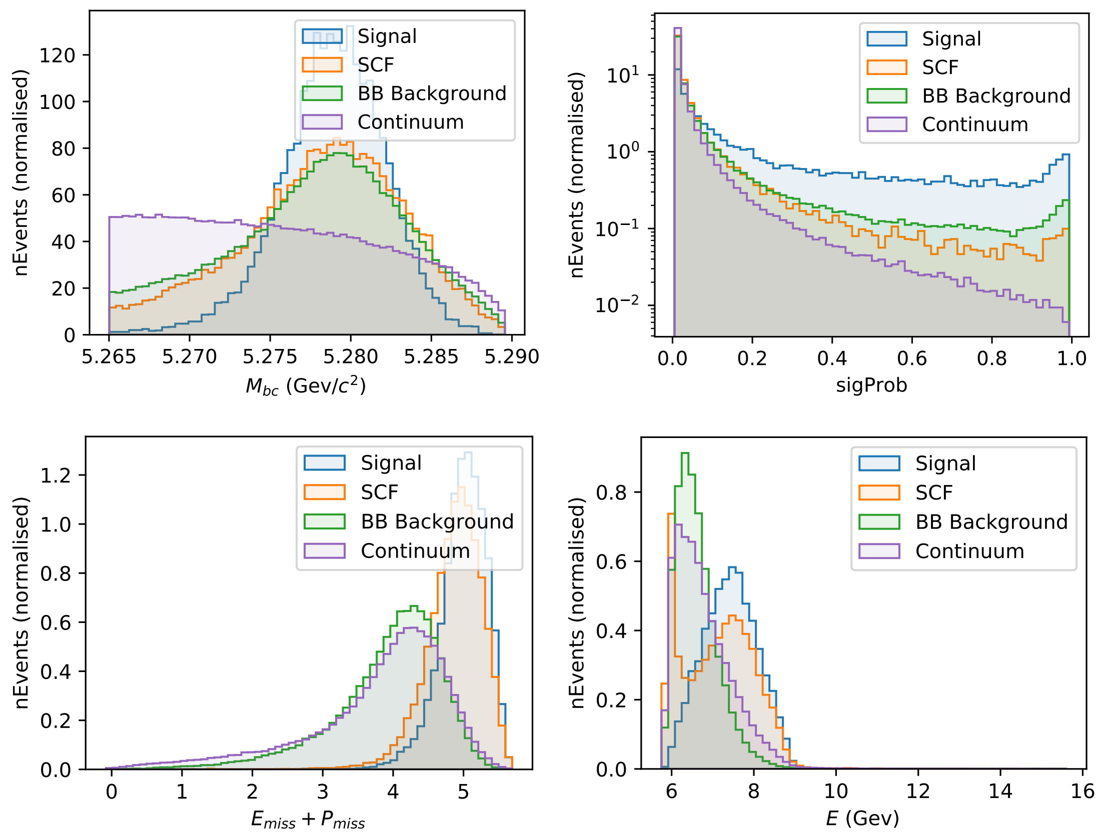


Figure A.2: K^+ hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots (except M_{bc}) have been normalised to show relative number of events across the full range.

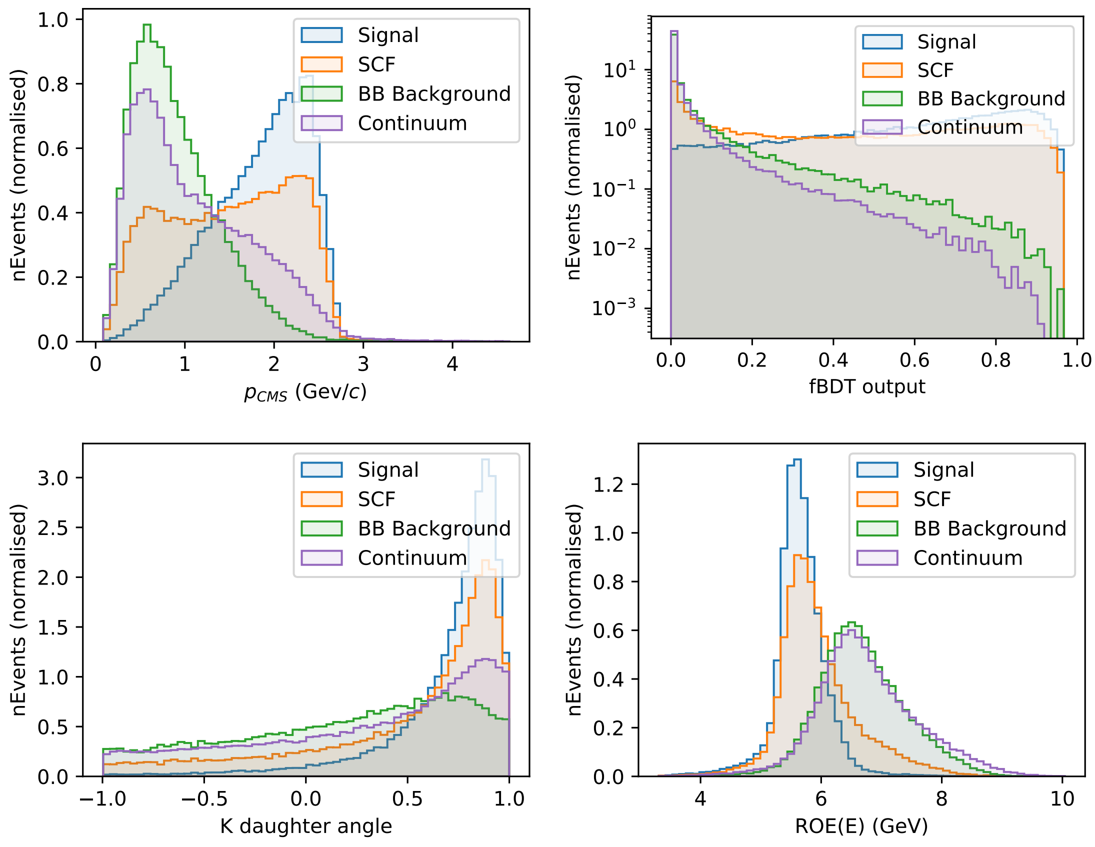


Figure A.3: K^{*0} hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots have been normalised to show relative number of events across the full range.

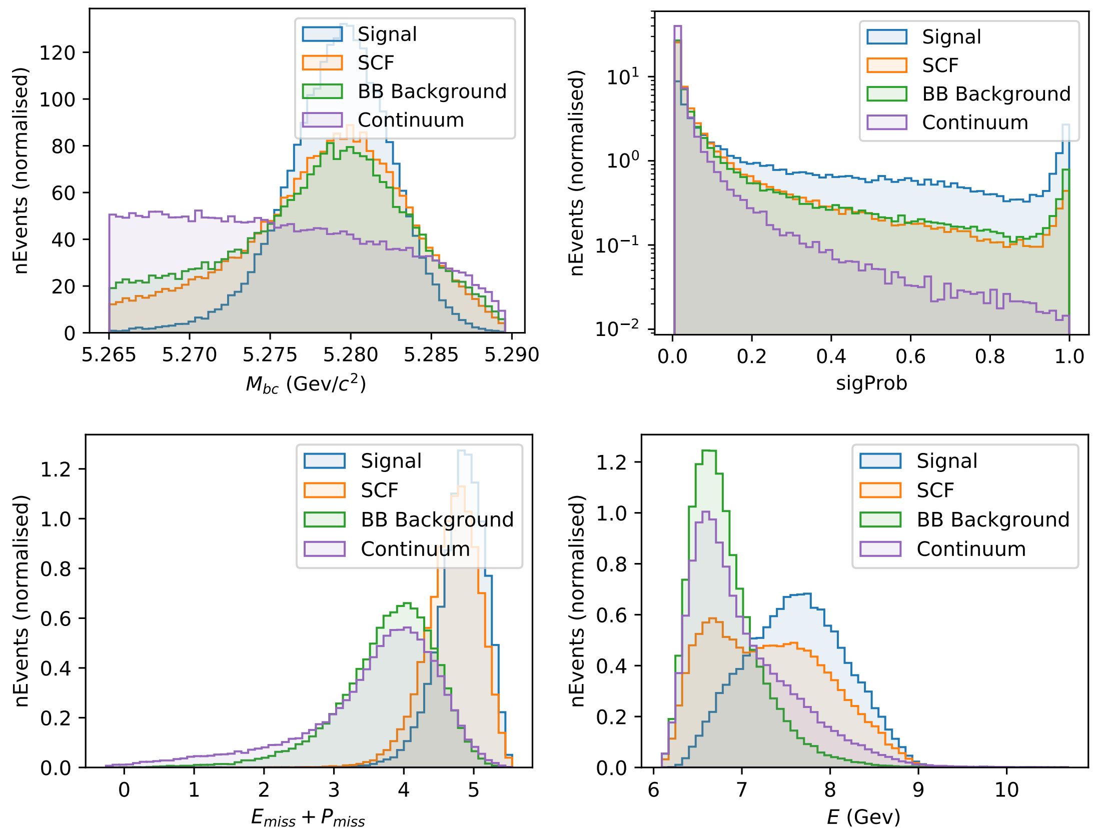


Figure A.4: K^{*0} hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots (except M_{bc}) have been normalised to show relative number of events across the full range.

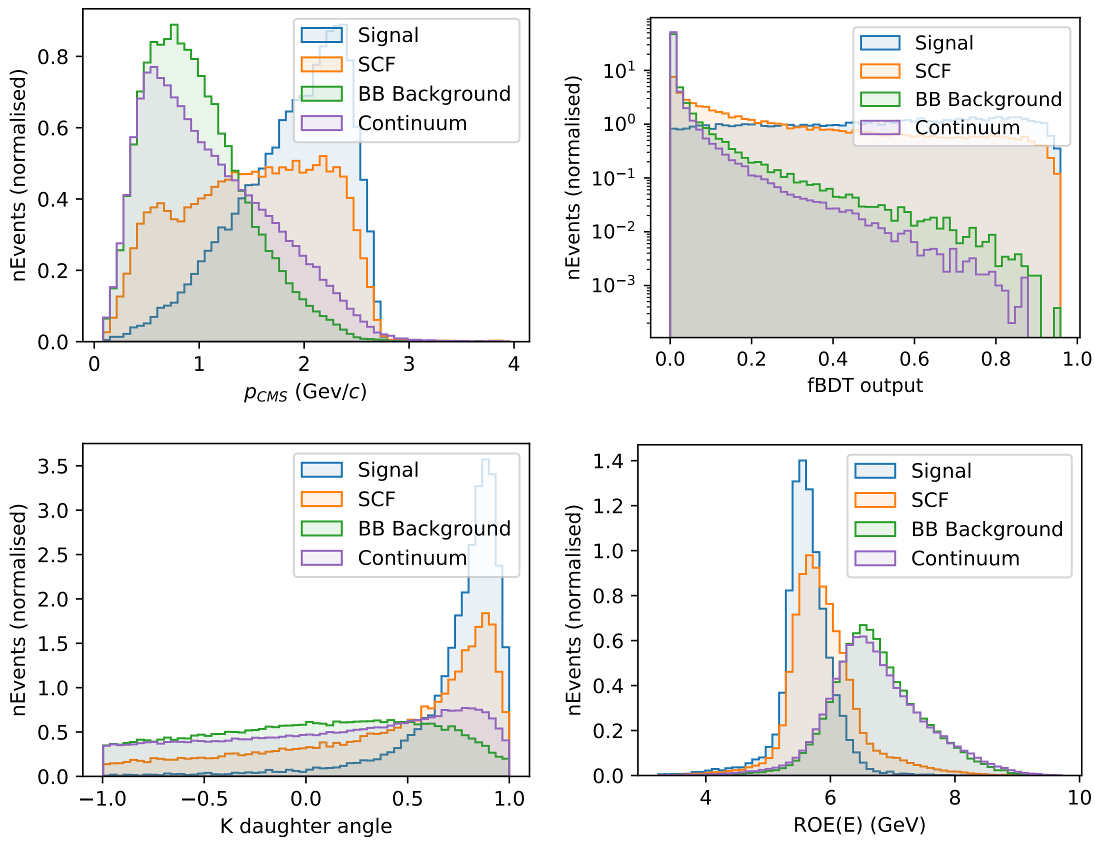


Figure A.5: K^{*+} hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots have been normalised to show relative number of events across the full range.

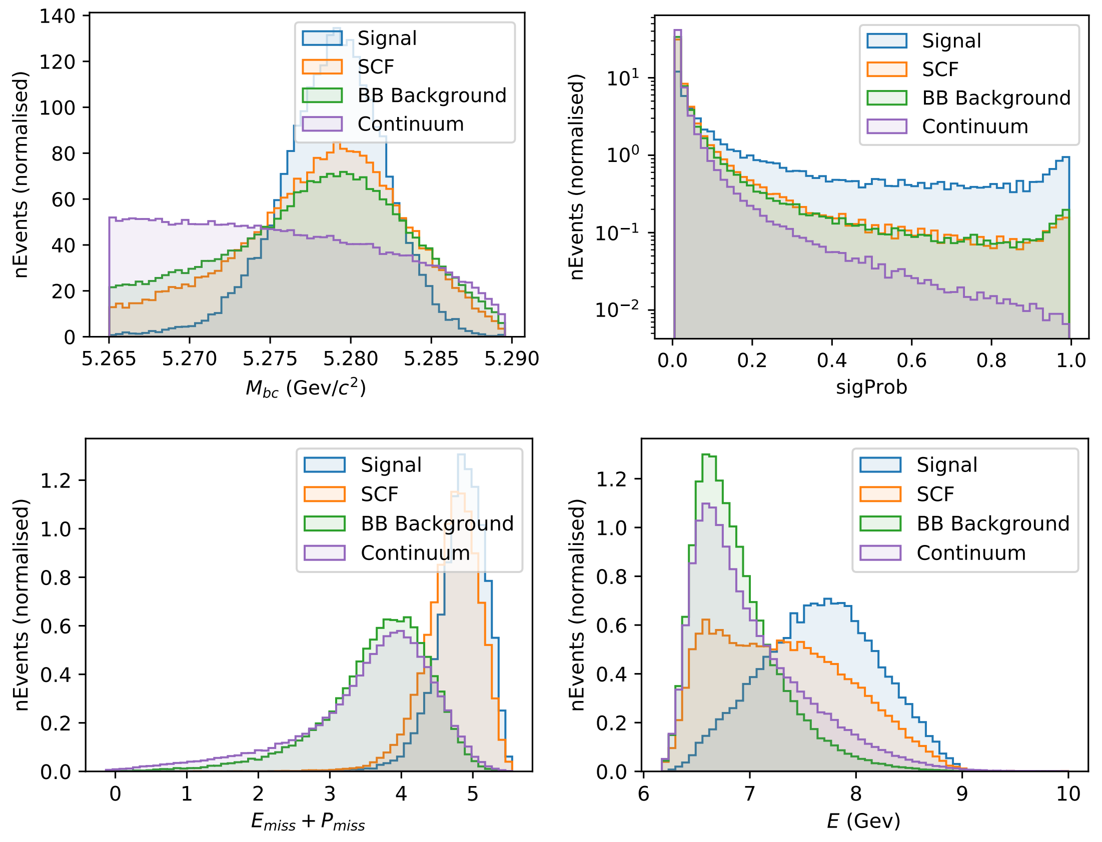


Figure A.6: K^{*+} hadronic reconstruction selection variables with preliminary selections from section 6.2.5 applied. All plots (except M_{bc}) have been normalised to show relative number of events across the full range.

Appendix B

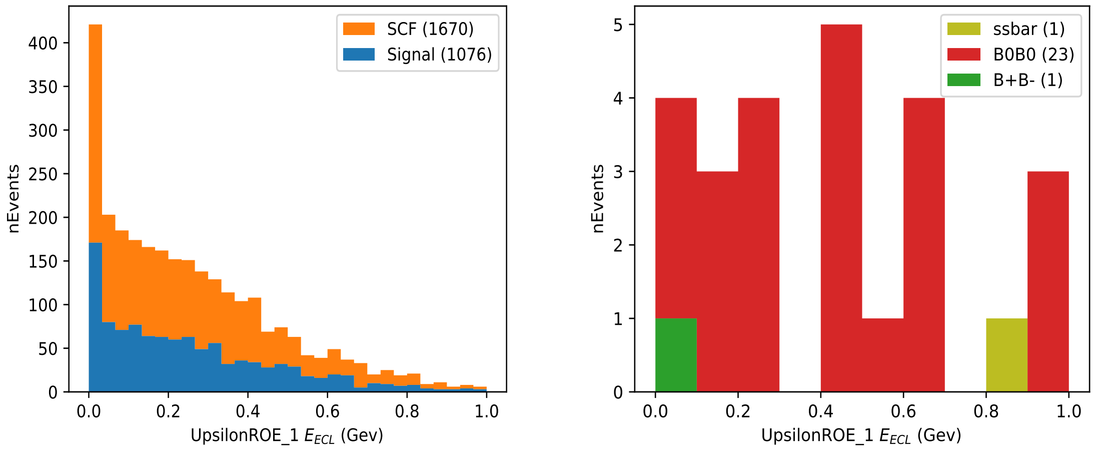
Optimised cut values

Table B.1: Cut selections obtained from optimisation performed in section 6.2.5.

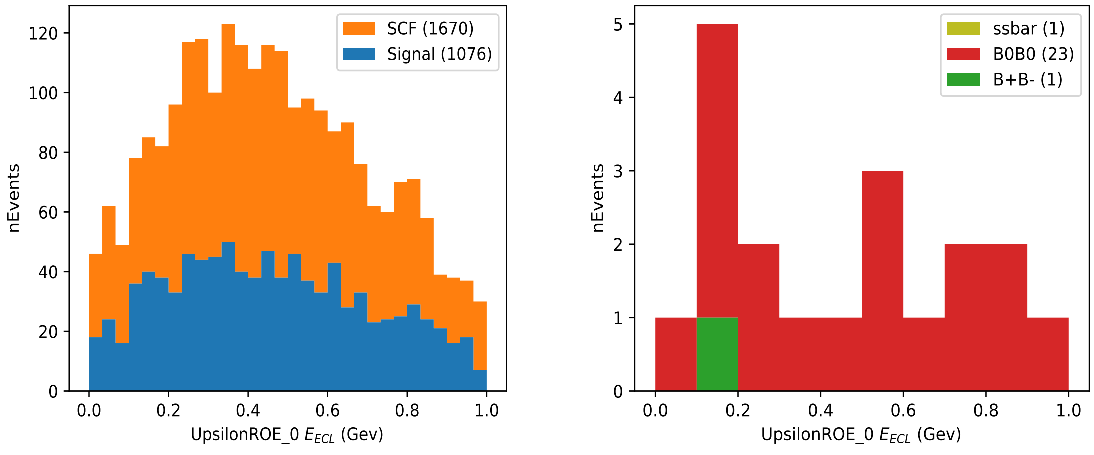
Variable	K^0	K^+	K^{*0}	K^{*+}
$p_{\text{CMS}}(\text{Gev/c}) >$	0.50	0.53	0.76	0.57
Kdaughter angle(rad) $>$	0.63	–	0.11	0.49
ROE(E)(GeV) $<$	6.42	6.04	6.08	5.91
$M_{\text{bc}}(\text{Gev/c}^2) >$	5.2738	5.2730	5.2742	5.2735
sigProb $>$	3.43×10^{-2}	2.64×10^{-2}	4.04×10^{-2}	8.17×10^{-2}
E(GeV) $>$	7.40	7.18	7.41	7.36
$fBDT >$	0.28	0.39	0.39	0.32
$E_{\text{miss}} + P_{\text{miss}} >$	3.71	4.12	4.13	4.14

Appendix C

Rest of event filter checks

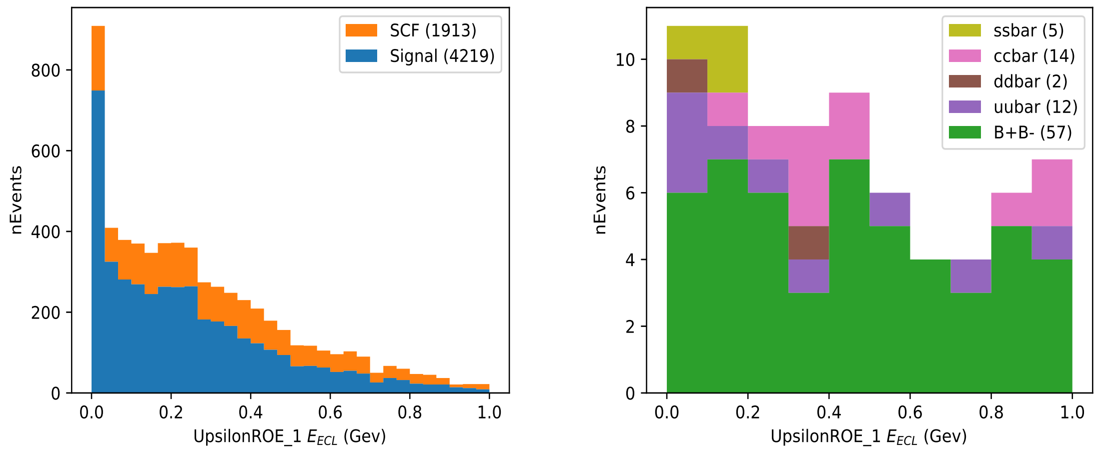


(a) UpsilonROE_1

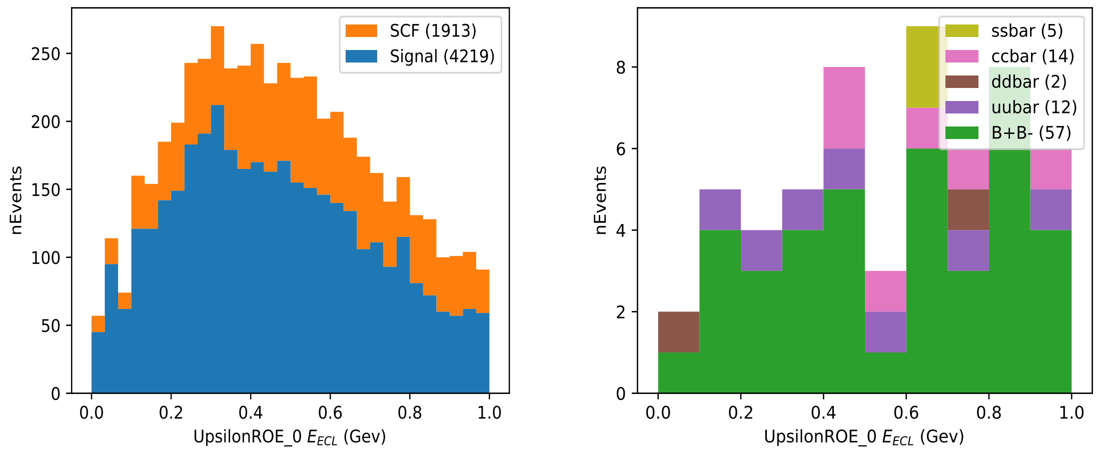


(b) UpsilonROE_0

Figure C.1: Example of changes to E_{ECL} in the K^0 channel for different rest of event restrictions in the signal region ($E_{\text{ECL}} < 1\,\text{GeV}$). All histograms shown here are stacked. The top row shows the second best performing ROE filter from the tests in section 6.2.3. The bottom row shows the baseline filter. The left column shows the change to signal and self-crossfeed, and the right to the individual background channels.

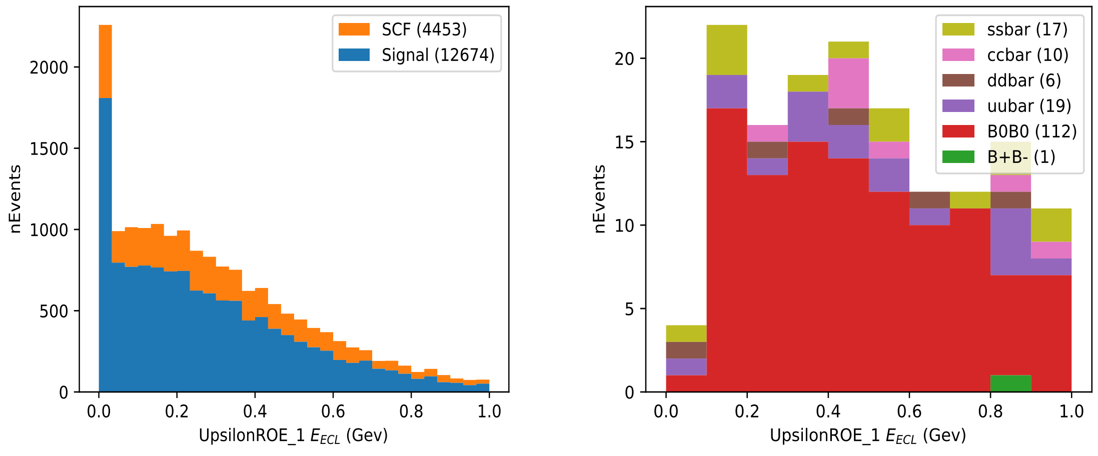


(a) UpsilonROE_1

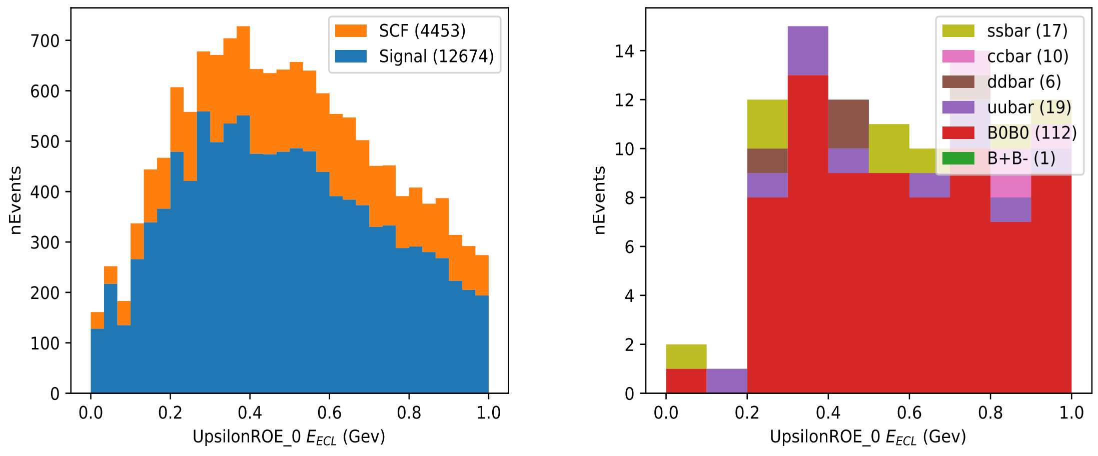


(b) UpsilonROE_0

Figure C.2: Example of changes to E_{ECL} in the K^{*+} channel for different rest of event restrictions in the signal region ($E_{\text{ECL}} < 1$ GeV). All histograms shown here are stacked. The top row shows the second best performing ROE filter from the tests in section 6.2.3. The bottom row shows the baseline filter. The left column shows the change to signal and self-crossfeed, and the right to the individual background channels.



(a) UpsilonROE_1

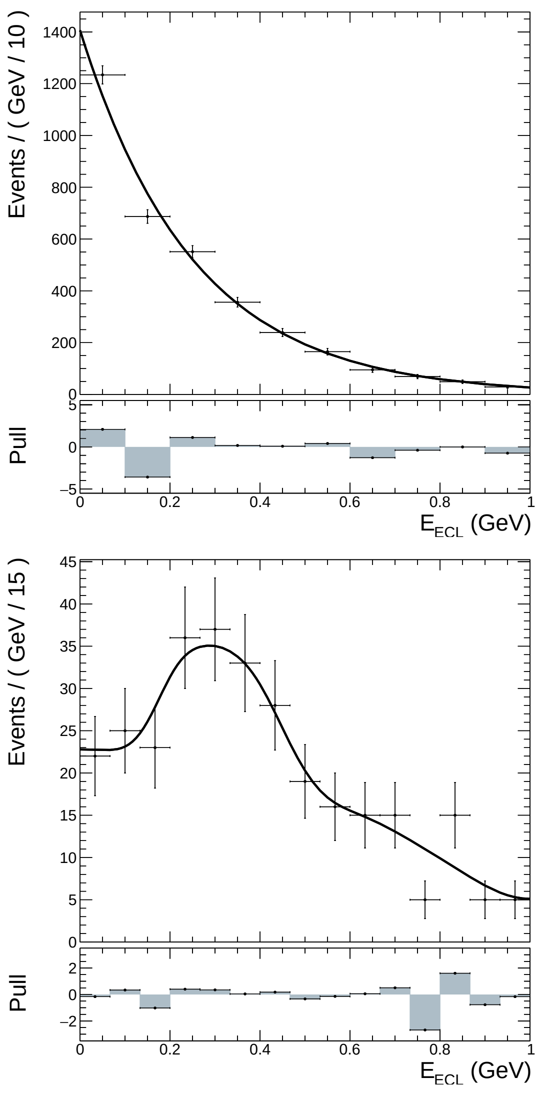


(b) UpsilonROE_0

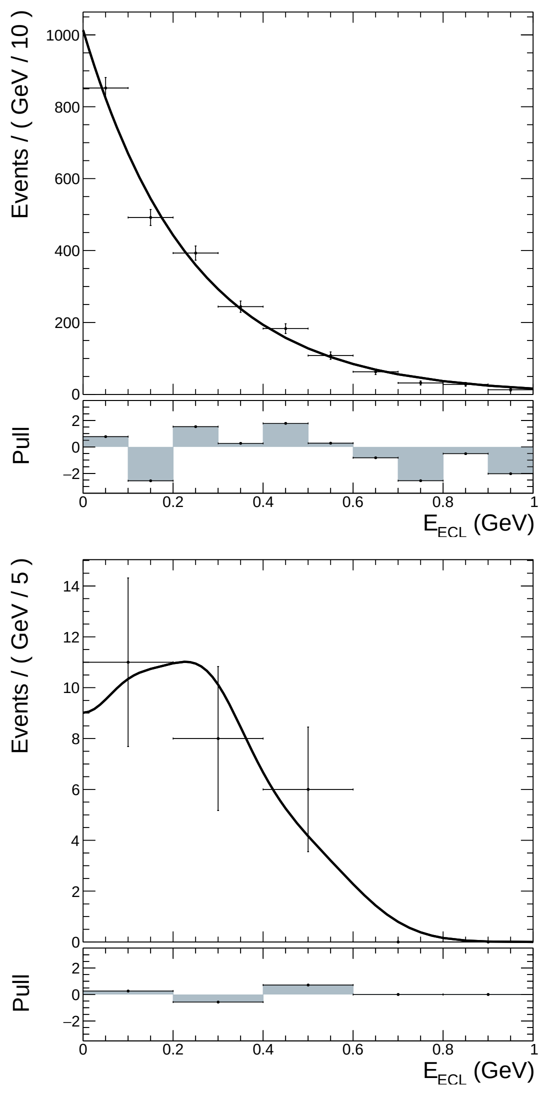
Figure C.3: Example of changes to E_{ECL} in the K^{*0} channel for different rest of event restrictions in the signal region ($E_{\mathrm{ECL}} < 1\,\mathrm{GeV}$). All histograms shown here are stacked. The top row shows the second best performing ROE filter from the tests in section 6.2.3. The bottom row shows the baseline filter. The left column shows the change to signal and self-crossfeed, and the right to the individual background channels.

Appendix D

E_{ecl} fit components

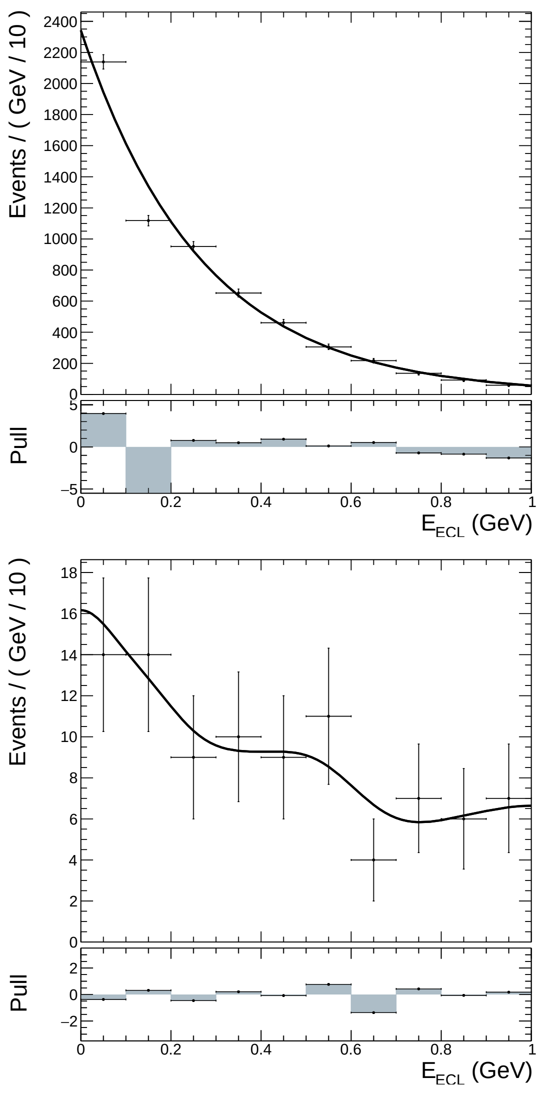


(a) K^+

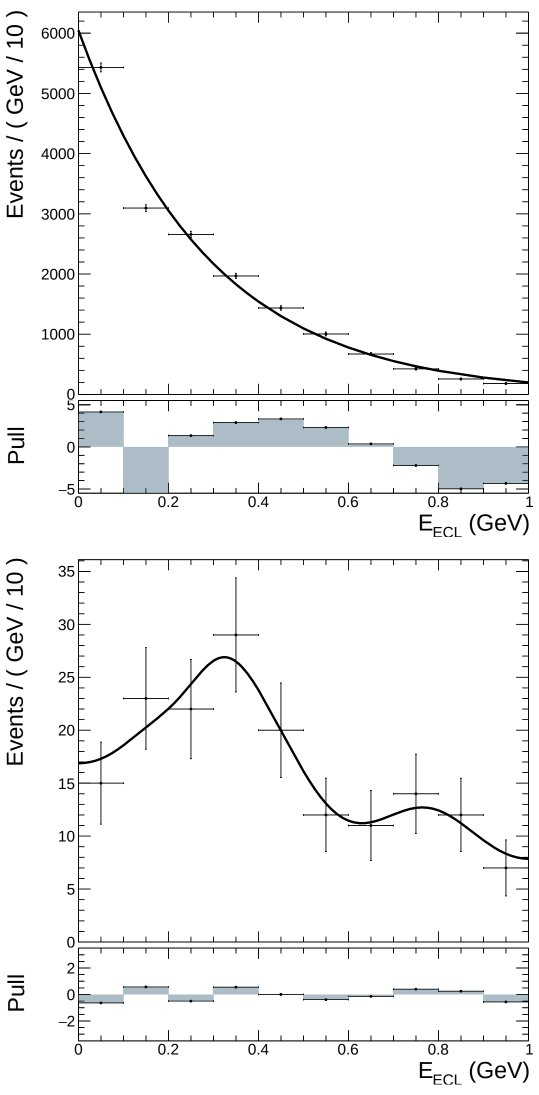


(b) K^0

Figure D.1: Individual signal exponential (top) and background KDE (bottom) fit shapes to E_{ECL} obtained from Monte Carlo simulations.



(a) K^+

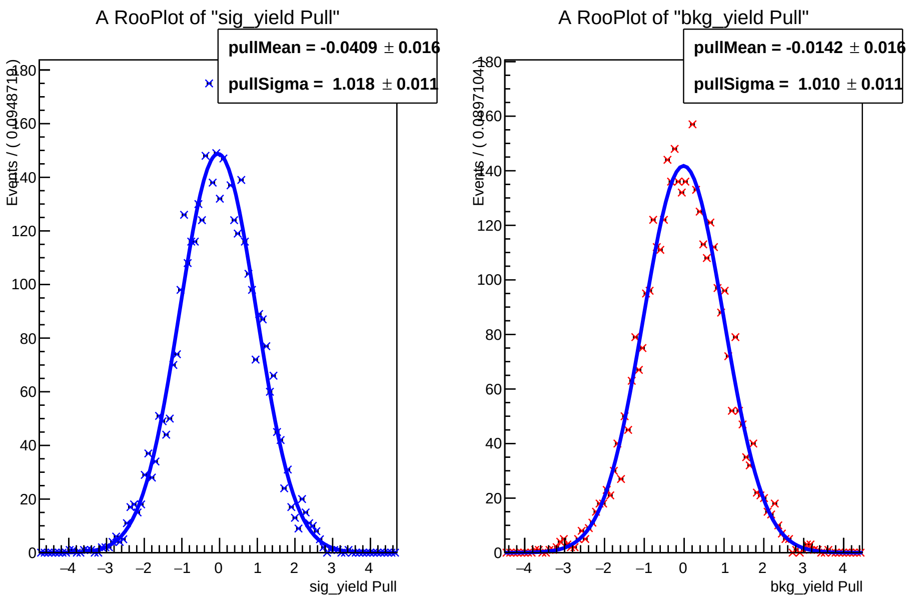


(b) K^0

Figure D.2: Individual signal exponential (top) and background KDE (bottom) fit shapes to E_{ECL} obtained from Monte Carlo simulations.

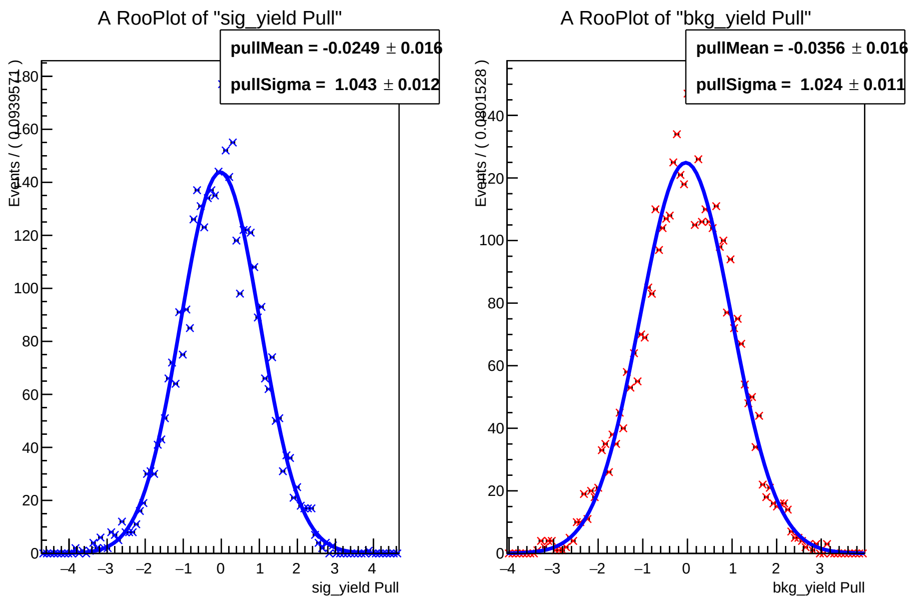
Appendix E

Pull distributions

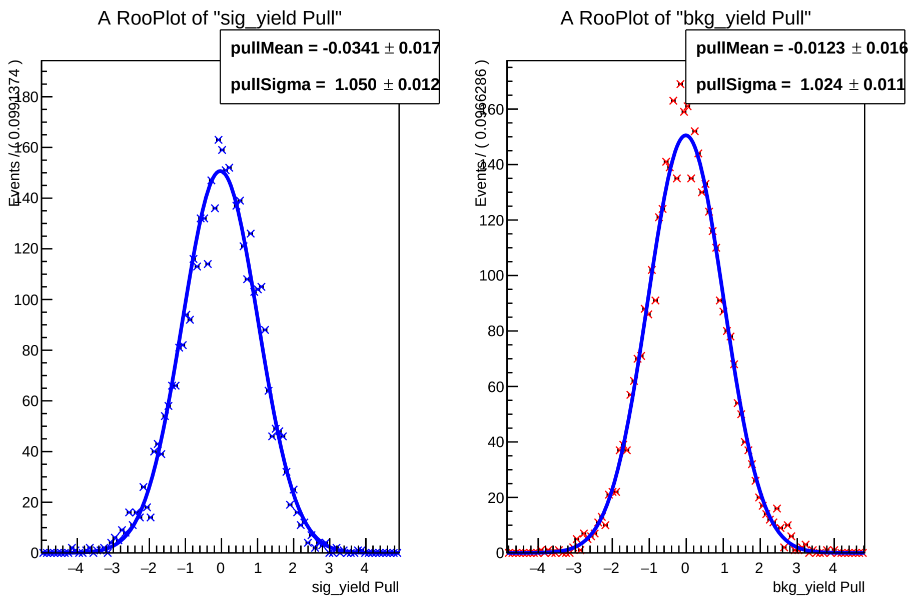


(a) K^+

Figure E.1: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 2 ab^{-1} luminosity equivalent using histogram template models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits.

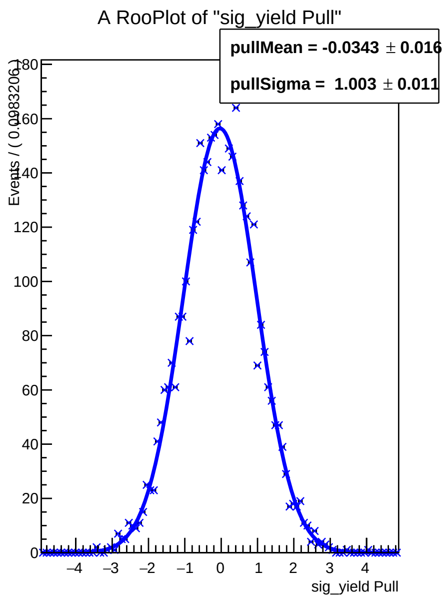


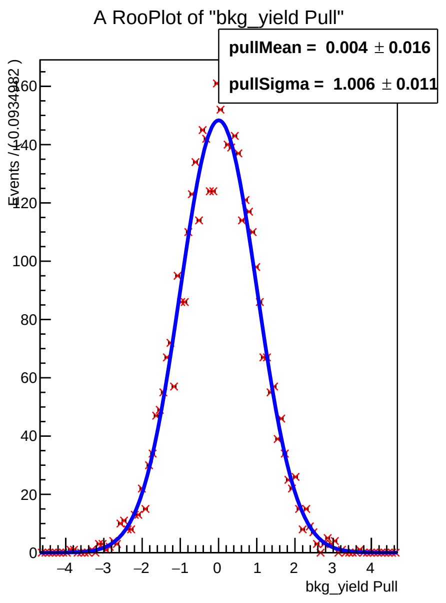
(a) K^{*+}



(b) K^{*0}

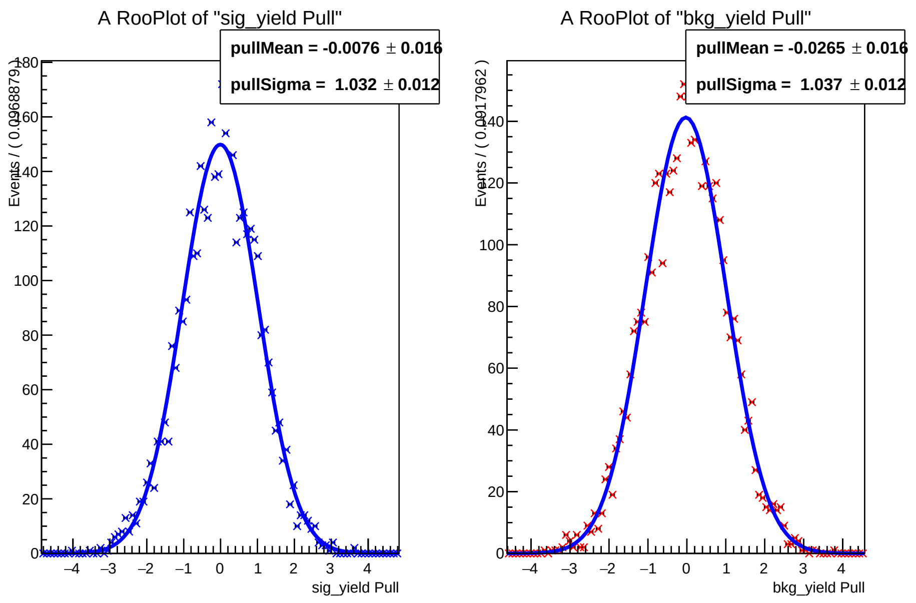
Figure E.2: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 2 ab^{-1} luminosity equivalent using histogram template models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits.



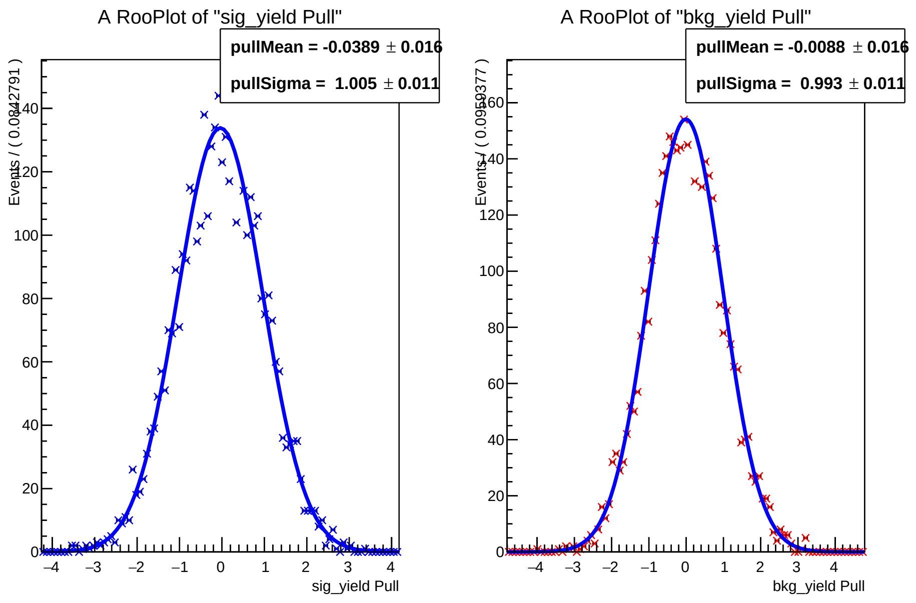


(a) K^+

Figure E.3: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 5 ab^{-1} luminosity equivalent using histogram template models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits.



(a) K^{*+}



(b) K^{*0}

Figure E.4: Pulls and corresponding Gaussian fits of the set of fits to toyMC tests at 5 ab^{-1} luminosity equivalent using histogram template models. The left plots with blue data points are the results of the signal yield fits, and the right plots with red data points show the background yield fits.

Appendix F

Upper limit scaling test values

Luminosity	Upper limit		$\Delta(-\log(L))$		$\mathcal{B}$ upper limit	
equiv. (ab^{-1})	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
1	45	26	0.7	0.9	3.23×10^{-5}	1.92×10^{-5}
2	70	36	0.8	1.0	2.53×10^{-5}	1.31×10^{-5}
3	93	44	0.9	1.2	2.24×10^{-5}	1.07×10^{-5}
4	114	50	1.1	1.3	2.06×10^{-5}	9.12×10^{-6}
5	134	56	1.2	1.4	1.93×10^{-5}	8.07×10^{-6}
6	153	63	1.3	1.5	1.83×10^{-5}	7.57×10^{-6}
7	172	67	1.4	1.5	1.77×10^{-5}	6.88×10^{-6}
8	192	72	1.6	1.7	1.72×10^{-5}	6.53×10^{-6}
9	209	75	1.7	1.7	1.66×10^{-5}	6.03×10^{-6}
10	229	79	1.9	1.9	1.64×10^{-5}	5.72×10^{-6}
11	247	83	2.0	2.0	1.61×10^{-5}	5.47×10^{-6}
12	266	88	2.2	2.0	1.59×10^{-5}	5.28×10^{-6}
13	282	88	2.3	2.0	1.56×10^{-5}	4.90×10^{-6}
14	297	95	2.4	2.2	1.52×10^{-5}	4.89×10^{-6}
15	318	96	2.6	2.2	1.52×10^{-5}	4.62×10^{-6}
16	332	99	2.7	2.2	1.49×10^{-5}	4.47×10^{-6}
17	348	104	2.9	2.3	1.47×10^{-5}	4.40×10^{-6}
18	366	106	3.0	2.4	1.46×10^{-5}	4.23×10^{-6}
19	384	106	3.2	2.4	1.45×10^{-5}	4.03×10^{-6}
20	400	113	3.3	2.5	1.43×10^{-5}	4.05×10^{-6}
21	415	115	3.4	2.5	1.42×10^{-5}	3.93×10^{-6}
22	432	118	3.6	2.6	1.41×10^{-5}	3.86×10^{-6}
23	450	117	3.8	2.6	1.40×10^{-5}	3.67×10^{-6}
24	464	120	3.8	2.7	1.39×10^{-5}	3.59×10^{-6}
25	482	123	4.0	2.7	1.38×10^{-5}	3.54×10^{-6}

Table F.1: K^+ upper limit scaling test values from toyMC tests performed in section 7.2.2.

Luminosity	Upper limit		$\Delta(-\log(L))$		$\mathcal{B}$ upper limit	
equiv. (ab^{-1})	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
1	24	16	0.5	0.8	1.44×10^{-4}	9.85×10^{-5}
2	35	23	0.6	0.8	1.06×10^{-4}	7.01×10^{-5}
3	46	28	0.6	0.8	9.17×10^{-5}	5.54×10^{-5}
4	54	32	0.6	0.8	8.05×10^{-5}	4.82×10^{-5}
5	63	36	0.6	0.9	7.49×10^{-5}	4.37×10^{-5}
6	70	40	0.7	0.9	7.00×10^{-5}	3.95×10^{-5}
7	78	43	0.7	0.9	6.65×10^{-5}	3.71×10^{-5}
8	84	45	0.7	0.9	6.23×10^{-5}	3.38×10^{-5}
9	91	48	0.7	1.0	6.05×10^{-5}	3.17×10^{-5}
10	99	51	0.8	1.0	5.88×10^{-5}	3.05×10^{-5}
11	104	54	0.8	1.0	5.64×10^{-5}	2.91×10^{-5}
12	111	54	0.8	1.0	5.52×10^{-5}	2.71×10^{-5}
13	119	57	0.8	1.1	5.45×10^{-5}	2.62×10^{-5}
14	123	59	0.8	1.1	5.22×10^{-5}	2.53×10^{-5}
15	132	62	0.9	1.1	5.23×10^{-5}	2.45×10^{-5}
16	137	64	0.9	1.1	5.09×10^{-5}	2.39×10^{-5}
17	142	66	0.9	1.2	4.99×10^{-5}	2.30×10^{-5}
18	148	68	1.0	1.2	4.90×10^{-5}	2.27×10^{-5}
19	153	69	1.0	1.2	4.80×10^{-5}	2.18×10^{-5}
20	160	71	1.0	1.2	4.75×10^{-5}	2.13×10^{-5}
21	166	73	1.0	1.2	4.70×10^{-5}	2.08×10^{-5}
22	172	75	1.1	1.3	4.65×10^{-5}	2.03×10^{-5}
23	177	77	1.1	1.3	4.57×10^{-5}	1.99×10^{-5}
24	182	79	1.1	1.3	4.50×10^{-5}	1.97×10^{-5}
25	189	81	1.2	1.4	4.50×10^{-5}	1.93×10^{-5}

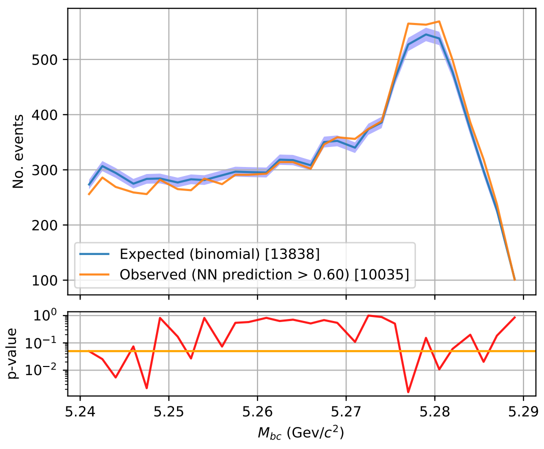
Table F.2: K^{*+} upper limit scaling test values from toyMC tests performed in section 7.2.2.

Luminosity	Upper limit		$\Delta(-\log(L))$		$\mathcal{B}$ upper limit	
equiv. (ab^{-1})	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
1	30	19	0.6	0.8	6.41×10^{-5}	4.17×10^{-5}
2	46	27	0.7	0.9	4.89×10^{-5}	2.95×10^{-5}
3	58	33	0.7	1.0	4.17×10^{-5}	2.35×10^{-5}
4	70	37	0.7	1.0	3.76×10^{-5}	1.98×10^{-5}
5	82	41	0.8	1.0	3.49×10^{-5}	1.76×10^{-5}
6	93	44	0.8	1.1	3.30×10^{-5}	1.59×10^{-5}
7	104	48	0.9	1.2	3.17×10^{-5}	1.48×10^{-5}
8	114	53	1.0	1.2	3.03×10^{-5}	1.41×10^{-5}
9	125	55	1.1	1.3	2.96×10^{-5}	1.30×10^{-5}
10	133	57	1.1	1.3	2.83×10^{-5}	1.22×10^{-5}
11	143	61	1.2	1.4	2.77×10^{-5}	1.19×10^{-5}
12	152	62	1.2	1.3	2.70×10^{-5}	1.10×10^{-5}
13	160	66	1.2	1.4	2.62×10^{-5}	1.09×10^{-5}
14	172	68	1.4	1.5	2.62×10^{-5}	1.04×10^{-5}
15	178	70	1.4	1.4	2.53×10^{-5}	9.99×10^{-6}
16	187	73	1.4	1.5	2.49×10^{-5}	9.76×10^{-6}
17	197	75	1.5	1.6	2.46×10^{-5}	9.41×10^{-6}
18	209	76	1.6	1.7	2.47×10^{-5}	9.07×10^{-6}
19	215	80	1.6	1.7	2.41×10^{-5}	9.01×10^{-6}
20	224	82	1.7	1.8	2.39×10^{-5}	8.72×10^{-6}
21	234	85	1.8	1.8	2.37×10^{-5}	8.62×10^{-6}
22	239	84	1.8	1.8	2.31×10^{-5}	8.19×10^{-6}
23	249	86	1.8	1.8	2.30×10^{-5}	8.03×10^{-6}
24	257	90	1.9	1.9	2.28×10^{-5}	7.99×10^{-6}
25	265	90	2.0	1.8	2.26×10^{-5}	7.69×10^{-6}

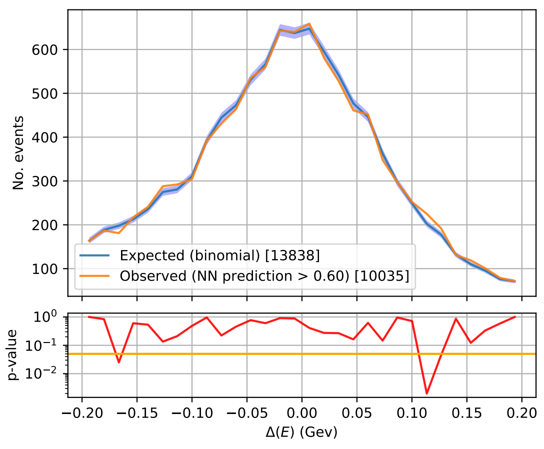
Table F.3: K^{*0} upper limit scaling test values from toyMC tests performed in section 7.2.2.

Appendix G

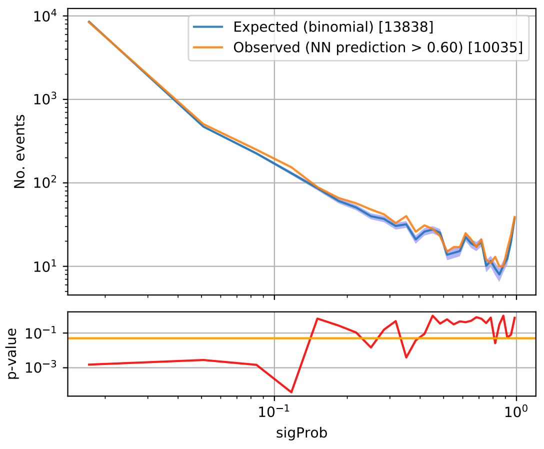
Bias evaluation



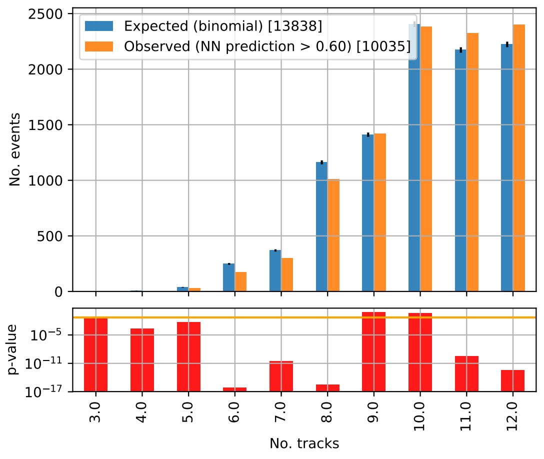
(a) M_{bc}



(b) $\Delta(E)$

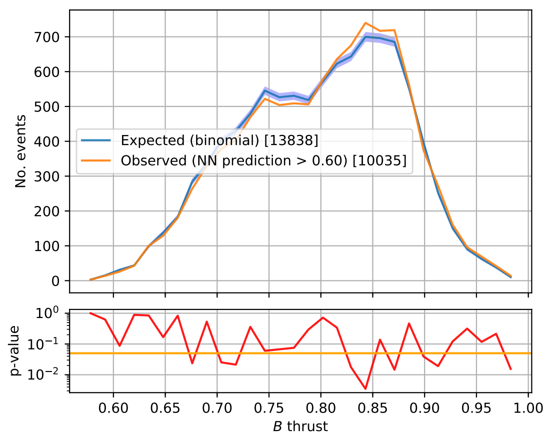


(c) sigProb (log axes)

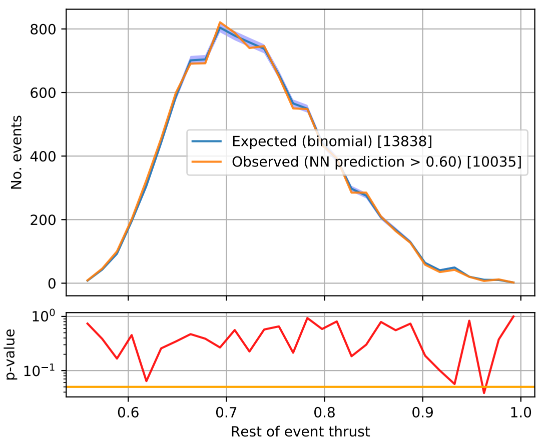


(d) nTracks

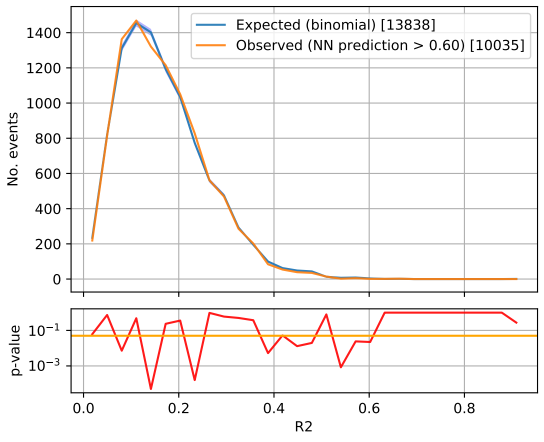
Figure G.1: Bias evaluation plots for mixed FEI B^0.



(a) B meson thrust

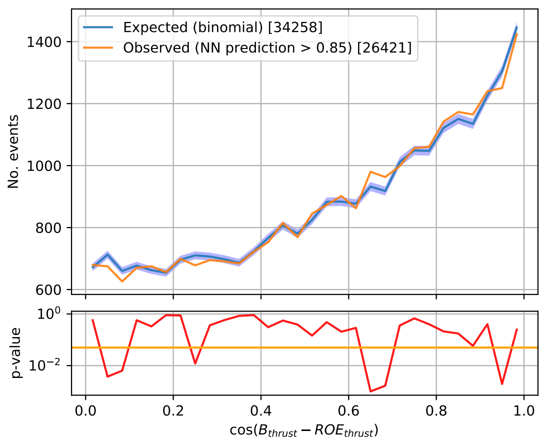


(b) ROE thrust

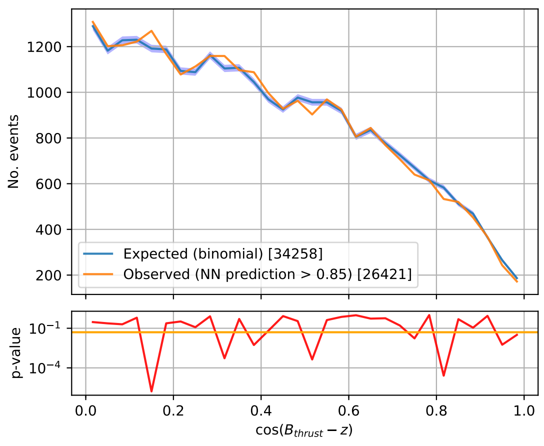


(c) R2

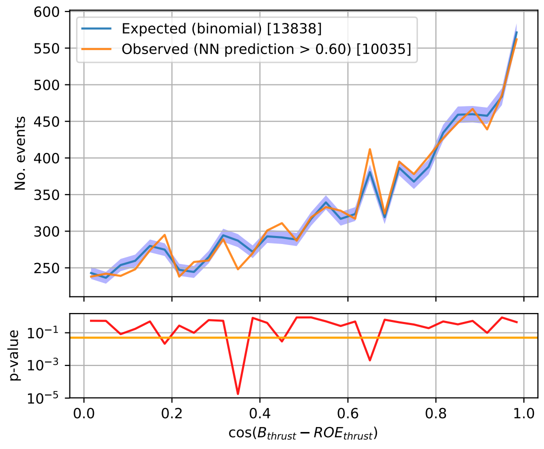
Figure G.2: Bias evaluation plots for mixed FEI B^0.



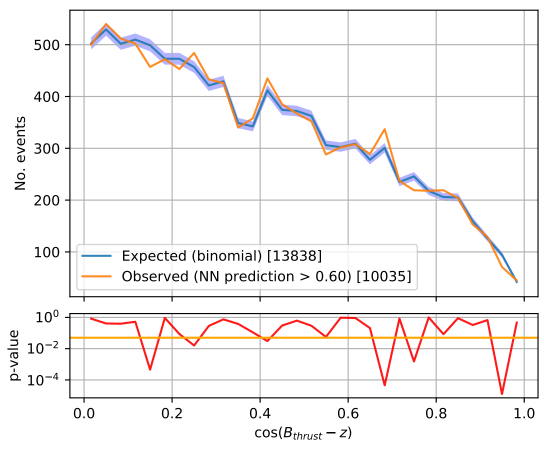
(a) $\cos(B_{\text{thrust}} - \text{ROE}_{\text{thrust}})$



(b) $\cos(B_{\text{thrust}} - z)$



(c) $\cos(B_{\text{thrust}} - \text{ROE}_{\text{thrust}})$



(d) $\cos(B_{\text{thrust}} - z)$

Figure G.3: Additional thrust vector inputs to biasNN for charged FEI B^+ (top) and mixed FEI B^0 (bottom).

Acknowledgements

Firstly I wish to thank my supervisor, Prof. Dr. Thomas Kuhr, for giving me the incredible opportunity to come to Germany and be a part of the historical physics research community here in Munich. His support and encouragement throughout my PhD gave me both the freedom to grow as a researcher and guidance when I needed it most. Thank you for sharing your enthusiasm and vast physics knowledge!

I would like to thank Dr. Martin Ritter, who was always happy to share his wisdom about work, programming, and life in general. He helped immensely with the process of moving and settling in to a new country, as well as with settling into the Belle II group in general. For his advice and guidance I am endlessly thankful.

I want to thank the other students in AG-Kuhr: Maryam for always being there for a chat when I needed a break and dropped by; Pascal for making the last months of late nights writing something to look forward to, and for introducing me to the music scene of Munich; Kilian and Matej for the office discussions; and the students who have been part of the group throughout my PhD that let me learn to be a mentor as well as a student.

My master's supervisor, Prof. Martin Sevior, who I wish to thank for first telling me about this position when I was already on my way to leaving the field of physics. He encouraged me to take the opportunity and without him I would not be here.

I also want to thank my girlfriend, Ying, for supporting me in my studies, especially over the last few months of long work hours.

To the lunchtime coffee crew, I look forward to our discussions each day. It's been somehow reminiscent of a sitcom on the couches in the cluster, and with Danny's daily facts I learnt something new every day. I owe a special thanks to David for taking the time to go through the theory section of my thesis and answering the questions I had about theory in general. Also to Dan, thank you for the English advice, despite not speaking the correct kind.

I would also like to acknowledge the support by the DFG Cluster of Excellence "Origin and Structure of the Universe". Portions of the simulations in this study have been carried out on the computing facilities of the Computational Center for Particle and Astrophysics (C2PAP).

Finally, to my family, I owe you the most thanks of all. To my parents, your support, advice, and encouragement has been endless. Not just through my PhD and moving overseas, but throughout my whole life. It's what has enabled me to follow my passion. And to my sister, thank you for coming to visit and making travelling Europe so much fun.