ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR SIMULATION-BASED INFERENCE IN $B^0 \to K^{*0}\ell^+\ell^-$

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

MASTER OF SCIENCE

IN

PHYSICS

December, 2025

By

Ethan Lee

Thesis Committee:

Tom Browder, Chairperson
Sven Vahsen
Peter Sadowski

Keywords: artificial intelligence, machine learning, simulation-based inference, particle physics, Belle II, Wilson coefficients

Contents

1 Abstract

We investigate neural simulation-based inference approaches to fitting the deviation of Wilson Coefficient 9 (C_9) from its Standard Model value (C_9^{SM}) given $B^0 \to K^{*0}\mu^+\mu^-$ events simulated in the context of the Belle II experiment. We denote this deviation as δC_9. We compare three neural network-based approaches to this multi-dimensional fitting problem. The first approach converts the dataset into a three-dimensional grid and fits for δC_9 using computer vision techniques. The second approach uses the deep sets architecture to predict δC_9 from a dataset while enforcing the permutation invariance of events. The third approach trains a classification model to predict a binned probability distribution over δC_9 given a single event. Predictions are then aggregated using the independence of events to obtain the binned δC_9 probability distribution given the entire dataset. We train and evaluate models on simulated datasets with and without detector effects. We also train and evaluate models on a dataset that includes simulated background events from the M_{bc} sideband.

2 Introduction

Belle II is a high energy physics experiment that measures collisions at the SuperKEKB accelerator. SuperKEKB collides e^+ and e^- beams at the $\Upsilon(4S)$ resonance to produce B^0 $\bar{B}^0$ pairs and B^+ B^- pairs. A few B^0 and $\bar{B}^0$ mesons decay as $B^0 \to K^{*0}\mu^+\mu^-$ and $\bar{B}^0 \to \bar{K}^{*0}\mu^-\mu^+$. These decays are suppressed at tree level in the Standard Model and therefore are sensitive to new physics, including lepton flavor universality violation (LFUV). If new physics is present, it may appear as a deviation of Wilson Coefficient 9 (C_9) from its Standard Model value (C_9^{SM}) [1, 3, 5, 6, 7, 13], where C_9 is a coefficient of an effective field theory. With the expected 50 ab^{-1} full Belle II dataset, a significant C_9 deviation can be precisely measured. We denote the deviation of C_9 as δC_9, where $\delta C_9 \equiv C_9 - C_9^{SM}$.

In terms of C_9, the matrix element for $B^0 \to K^{*0}\ell^+\ell^-$, as in A. Sibidanov

et al. [16] and W. Altmannshofer et al. [2], is

$$\mathcal{M} = \frac{G_F \alpha}{\sqrt{2}\pi} V_{tb} V_{ts}^* \Bigg\{ \Big[\langle K^* | \bar{s} \gamma^\mu (C_9^{\text{eff}} P_L + C_9' P_R) b | \bar{B} \rangle - \frac{2 m_b}{q^2} \langle K^* | \bar{s} i \sigma^{\mu\nu} q_\nu (C_7^{\text{eff}} P_R + C_7' P_L) b | \bar{B} \rangle \Big] (\bar{\ell} \gamma_\mu \ell) + \langle K^* | \bar{s} \gamma^\mu (C_{10} P_L + C_{10}' P_R) b | \bar{B} \rangle (\bar{\ell} \gamma_\mu \gamma_5 \ell) \Bigg\} \tag{1}$$

where C_9^{eff} is

$$C_9^{\text{eff}} = C_9 + Y(q^2) = 4.211 + Y(q^2) \tag{2}$$

for the Standard Model (with NNLL accuracy) and where ℓ is μ, e, or τ and $Y(q^2)$ is a function of q^2. P_L and P_R in equation 1 are the left and right-handed projection operators and are given by

$$P_L = \frac{1}{2}(1 - \gamma^5) \tag{3}$$

$$P_R = \frac{1}{2}(1 + \gamma^5) \tag{4}$$

Current fits for δC_9 use intermediate variables calculated from distributions of measured $B^0 \to K^{*0} \mu^+ \mu^-$ angular variables, such as the forward-backward asymmetry A_{FB} or P_5'. A downside to this approach is that information is lost in computing these intermediate variables. Traditional multi-dimensional fitting techniques are also difficult because they require parameterization of detector effects and backgrounds.

Here we explore neural networks as an alternative in fitting for δC_9. Neural networks are able to fit from high dimensional inputs, enabling the usage of measured variables, q^2, $\cos\theta_\mu$, $\cos\theta_K$, and χ, rather than intermediate variables such as A_{FB} and P_5'. Neural networks are also able to approximate complicated functions, allowing us to fit from inputs that include detector effects and backgrounds, without changing the architecture of the model.

Following the established framework known as simulation-based inference [8, 9], we use simulated data to train the neural networks to approximate a solution to the inverse problem of predicting δC_9 given a simulator that produces events given δC_9. See figure 1 for a visual. Using the Sibidanov Physics Generator [16], we simulate $B^0 \to K^{*0} \mu^+ \mu^-$ and $\bar{B}^0 \to \bar{K}^{*0} \mu^- \mu^+$ events at different values of δC_9. We train models at three levels of data realism: without detector simulation, with detector simulation, and with detector simulation and background events.

We compare three different neural network based approaches. The first approach uses computer vision techniques and was developed by S. Dubey [10]. In this approach we transform a dataset of $B^0 \to K^{*0} \mu^+ \mu^-$ events into a three-dimensional 'image' by binning the dataset in its three angular variables and

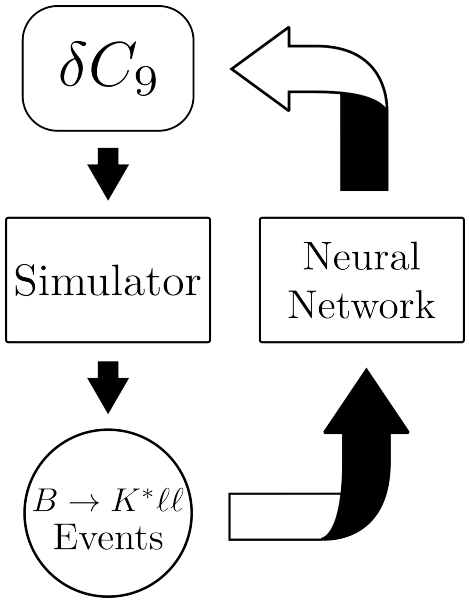

Figure 1: Diagram representing the simulation-based inference workflow for predicting δC_9 from a dataset of $B^0 \to K^{*0}\ell^+\ell^-$ events.

computing the average q^2 per bin. We then train a convolutional neural network (CNN) [12] to predict the δC_9 value used to generate the image. The second approach takes a dataset and predicts its δC_9 label using the deep sets architecture [17]. This model takes advantage of the permutation invariance of a dataset of events and does not require binning. The third approach is what we are referring to as the event-by-event approach. In this approach, we train a classification neural network to predict the binned δC_9 probability distribution given a single event. We use the independence of events to aggregate the per event predictions to obtain the binned δC_9 probability distribution given the dataset. The event-by-event approach is an example of neural posterior estimation (NPE) [9].

We compare model performances across simulation levels and input dataset sizes by evaluating different metrics such as mean squared error and mean absolute error. We also measure the sensitivity and bias of each model at a possible δC_9 new physics value. Finally, we discuss qualitative differences between the approaches such as ease of use.

3 Datasets

We train our models to predict δC_9 using simulated $B^0 \to K^{*0}\mu^+\mu^-$ and $\bar{B}^0 \to \bar{K}^{*0}\mu^-\mu^+$ signal events. These events are created at different δC_9 values using A. Sibidanov's new physics generator [16] and the Belle II basf2 software

Type	q^2 Veto	Split	Events
Signal (Generator)	Loose	Train	1.8×10^7
		Val.	1.8×10^7
	Tight	Train	1.6×10^7
		Val.	1.6×10^7
		Test	1.3×10^6
Signal (Detector)	Loose	Train	5.9×10^6
		Val.	5.9×10^6
	Tight	Train	5.3×10^6
		Val.	5.3×10^6
		Test	4.8×10^5
Background (SB)	Loose	Train	1.3×10^5
		Val.	1.3×10^5
	Tight	Train	1.2×10^5
		Val.	1.2×10^5
		Test	1.2×10^5
Background (SR)	Loose	Val.	757
	Tight	Val.	781
Signal (Generic, SR)	Loose	Val.	727
	Tight	Val.	710

Table 1: Summary of event counts. SB and SR stand for sideband and signal regions, respectively. Loose and tight q^2 vetos refer to $0 < q^2 < 20$ and $1 < q^2 < 19$ GeV^2 cuts respectively. A background suppression model is applied to the SR datasets.

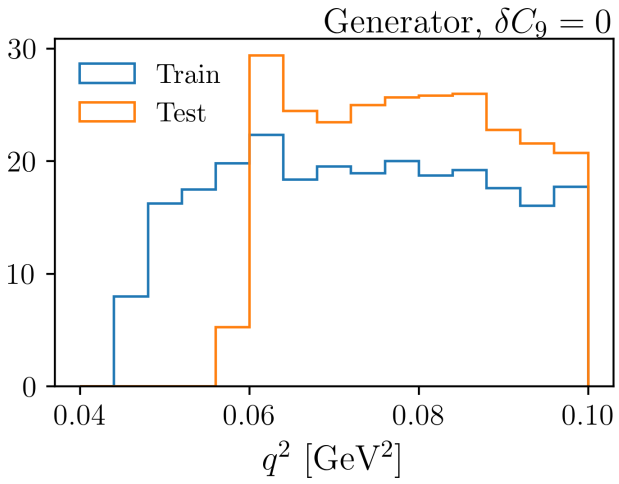

Figure 2: Visualization of the small shift in the q^2 distribution between the training and test datasets. The shift is due to changes in the simulation software pipeline. The shift is visible in the very low q^2 region and seems to result in an approximate 0.1% shift in the entire distribution.

framework [14]. Our training and validation datasets are composed of events simulated at 44 different values of δC_9 in the interval $[-2,\ 1.1]$. We assume a constant real value for δC_9. Theory fits to current data suggest a negative value for δC_9, so we focus on simulating events with negative values of δC_9. However, to improve the performance of the models near $\delta C_9 = 0$, we also simulate positive values of δC_9. Our test dataset includes events simulated at 14 values of δC_9 in the interval $[-2,\ 0.2]$. See table 1 for a summary of datasets.

We simulate two levels of signal events. The first level is the generator level, which does not incorporate detector effects. The second level is the detector level, which is created by adding detector simulation to the generator level events. We discuss detector effects in more detail in section 3.3. We also create datasets of background events, which we add to the detector level signal events to create a third dataset level which we refer to as the detector and background level.

At each of these three levels, we also create datasets using two different q^2 vetos, $0 < q^2 < 20$ GeV^2 and $1 < q^2 < 19$ GeV^2. The reason for this is as follows. We notice a small shift between the distributions of the training dataset and the test dataset, probably due to differences in the simulation software pipeline that accumulated in the time between the creation of the training dataset and the creation of the test dataset. See figure 2 for a visual. The original simulation software is now unavailable. However, the majority of the effects of the distribution shift can be mitigated by training models on datasets with the tighter $1 < q^2 < 19$ GeV^2 veto. In section 5, we present validation results for models

trained using the looser q^2 veto datasets to showcase potential results given a wide q^2 range. We also present test results for models trained using the tighter q^2 veto to demonstrate generalization capabilities given a small systematic shift in the input distribution. Note that we generated a new signal test dataset after retraining models with the tighter q^2 veto. However, we were unable to generate a new background test dataset because there is no accessible background event generator.

The background events used in the detector and background level datasets are obtained from reconstructing simulated generic B^0 $\bar{B}^0$ and B^+ B^- decays in the M_{bc} sideband of $4.8 < M_{bc} < 5.26$ GeV. Here, generic simulation refers to the simulation of the entire detector given e^+ e^- collisions. Ideally, we would use background events from the M_{bc} sideband of real data to obtain a more realistic background dataset. However, because this is a proof of concept study, we stick to simulated background events. We use events from the M_{bc} sideband because it might be very difficult to collect a background dataset from the signal region of real data. Note that we leave the inclusion of continuum background events to future work. It would be possible to suppress continuum background events by training a background suppression model on a dataset of simulated continuum background events as in [4].

We use background events from the large $4.8 < M_{bc} < 5.26$ GeV sideband because there were too few background events in the small $5.2 < M_{bc} < 5.26$ GeV sideband that we also tried. This lack of events is a problem for two reasons. First, without a sufficient number of training examples, our models might not generalize to unseen data. Second, we find that an insufficient number of evaluation background events tends to bias model predictions during evaluation, leading to mischaracterization of performance. The larger number of background events from the larger M_{bc} sideband helps with these two issues. However, the distribution of the background events from the large M_{bc} sideband differs more from the distribution of signal region background events (see figures 3 and 4). This could introduce issues with model generalization to signal region backgrounds. We expect that, in the future, the full Belle II 50 ab^{-1} dataset will provide enough data to train models using the better matching smaller $5.2 < M_{bc} < 5.26$ GeV sideband.

Unfortunately, we also do not have access to a large enough generic simulation signal region dataset to obtain reliable signal region model evaluation results (with signal region backgrounds). We therefore do not present results on the generic signal region dataset. See the last two sections of table 1 for the number of signal region signal and background events. Signal region events are obtained by reconstructing generic simulation events in the $M_{bc} > 5.27$ GeV range.

3.1 Variables

Our models are fit on four variables: $\cos\theta_\mu$, $\cos\theta_K$, χ, and q^2, which we calculate for each event. Here, $\cos\theta_\mu$, $\cos\theta_K$, and χ are angular variables. The variable $\cos\theta_\mu$ is the cosine of the muon helicity angle θ_μ shown for general lepton flavor

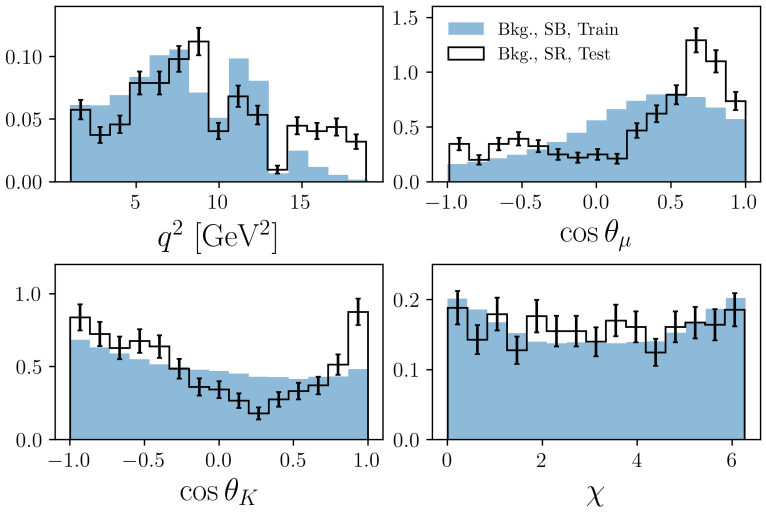

Figure 3: Comparison of signal region (SR) test and $4.8 < M_{bc} < 5.26$ GeV sideband (SB) training background distributions. We show results from the $1 < q^2 < 19$ GeV^2 dataset. There are 1.2×10^5 events in the SB training dataset and 776 events in the SR test dataset. Error bars are $\sqrt{n_i}$, where n_i is the number of events in the ith bin. Histograms are normalized.

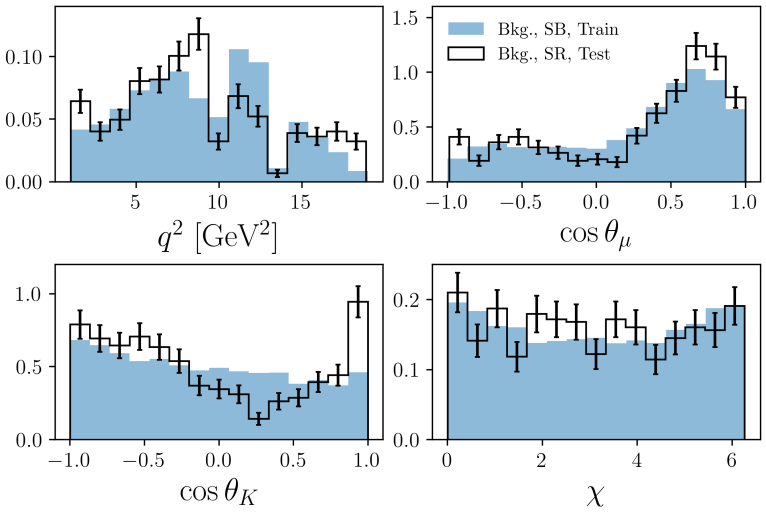

Figure 4: Same comparison as above, except the training dataset is taken from the smaller $5.2 < M_{bc} < 5.26$ GeV sideband. There are 1.6×10^4 events in the SB training dataset and 628 events in the SR test dataset.

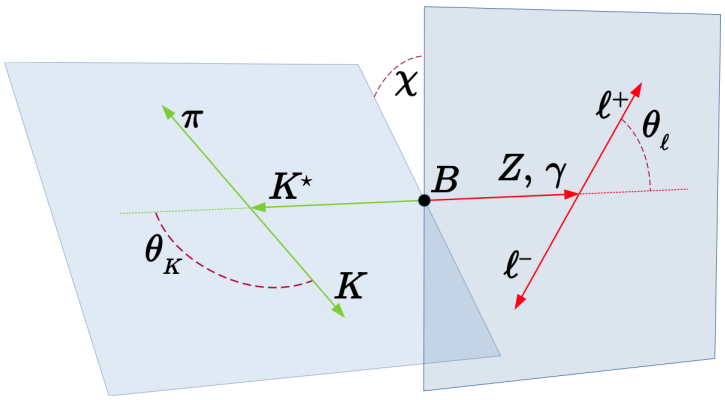

Figure 5: Diagram of a $B^0 \to K^{*0}\ell^+\ell^-$ decay. Used with permission from A. Sibidanov [16]. For the muon channel, ℓ is μ.

as θ_ℓ in figure 5. The variable θ_μ represents the angle between the momentum of the center of mass of the di-muon system in the B rest frame and the momentum of the positive muon in the di-muon system rest frame. The next variable, $\cos\theta_K$, is the cosine of the K^* helicity angle shown as θ_K in figure 5. The variable θ_K represents the angle between the momentum of the K^* in the B rest frame and the momentum of the K^*'s daughter kaon in the K^* rest frame. χ is the angle between the K^* decay plane and the di-muon decay plane as shown in figure 5. The last variable, q^2, is the squared invariant mass of the di-muon system. In figure 6 we plot the distributions of the four decay variables for simulated signal events for different values of δC_9. The figure also includes a side-by-side comparison of generator and detector level distributions.

3.2 Cuts

We apply selection requirements (cuts) to the detector simulation dataset to reduce the number of mis-reconstructed events and backgrounds. Cuts are described in table 2. Notable cuts include the J/ψ and $\psi(2S)$ q^2 vetos, which reduce the large number of background events from $B^0 \to K^{*0}J/\psi$ and $B^0 \to K^{*0}\psi(2S)$ events, respectively. Cuts in M_{bc} and ΔE are also important for reducing mis-reconstructed events and backgrounds. We show plots of the signal M_{bc} and ΔE distributions in figure 7. We note that we also examined a q^2 cut to the most sensitive q^2 region ($1 < q^2 < 8$ GeV^2). However, the tight cut reduced the performance of the models (possibly due to the reduction in training data), so we switched to the looser cuts in table 2. We also note that we do not yet apply best candidate selection and leave this to future work. However, we expect the average number of candidates per event to be close to one. After applying all cuts, the average number of candidates per event for signal, $B^0\bar{B}^0$ background, and B^+B^- background events in the signal region

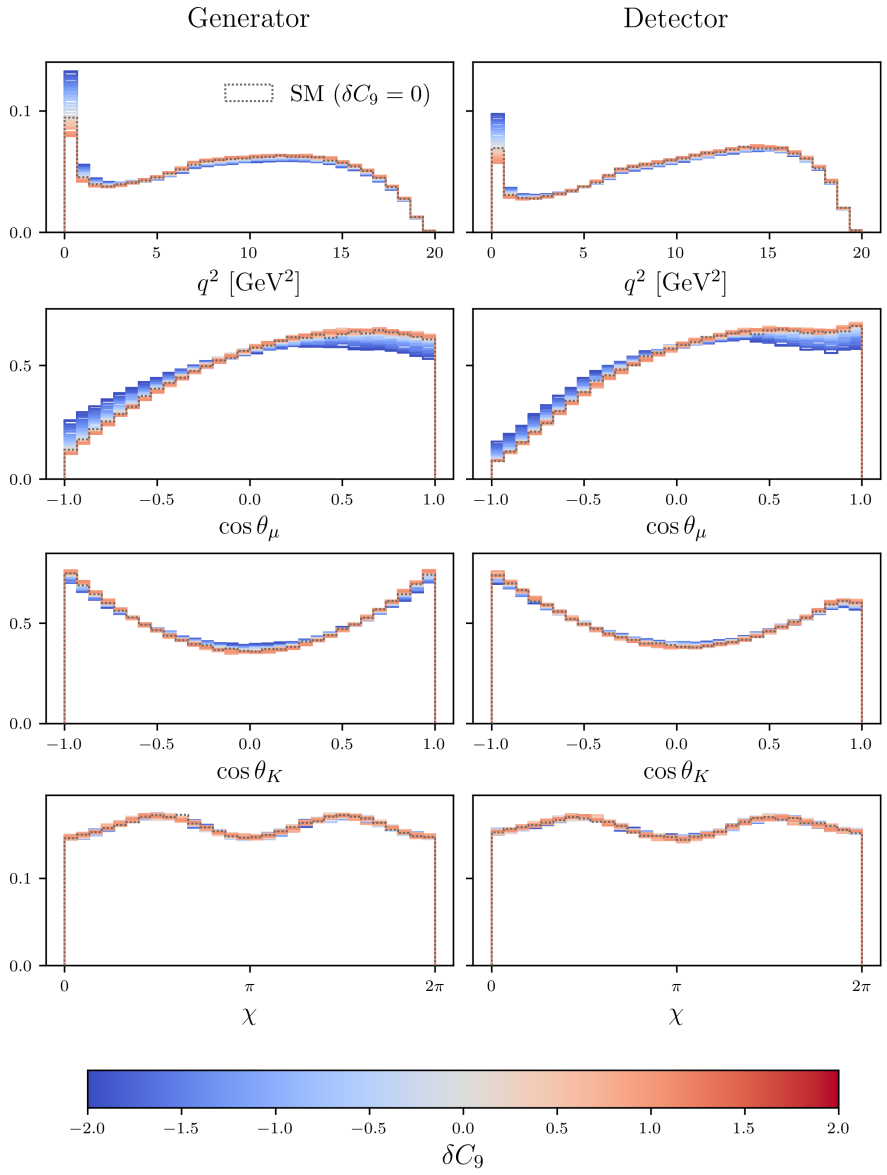

Figure 6: Normalized histograms of the decay variables. Color represents δC_9 value. We plot δC_9 values in the interval $[-2.0, 1.1]$. We show the same number of generator events as detector events.

($M_{bc} > 5.27$ GeV) is 1.00, 1.02, and 1.02 respectively in the $1 < q^2 < 19$ GeV^2 region, as calculated using our generic simulation dataset.

3.2.1 Background Suppression Model

In order to further reduce the number of background events in our generic simulation signal region dataset, we train a background suppression model using automated machine learning techniques. The resulting model is an ensemble primarily consisting of boosted decision tree (BDT) models. The model learns to classify each event as either background or signal. We only apply this model to the generic signal region datasets (last two sections of table 1). Applying this model to the M_{bc} sideband background dataset would result in too few background events, as discussed previously. We use this model to obtain a realistic signal to background ratio for creating model training and evaluation datasets using backgrounds from the M_{bc} sideband.

Our background suppression model is created using the AutoGluon automated machine learning software [11]. We train the models on a dataset composed of an equal number of background events from the M_{bc} sideband background training dataset and (Standard Model) signal events from the detector level signal training dataset. We train on sideband background events due to a lack of signal region background events. We evaluate the model on the generic signal region validation dataset. Inputs to the model are the event-by-event variables listed in table 3. The output of the model is the probability that an event is a signal event. We determine an optimal cut on the background suppression model output by maximizing the figure of merit (FOM)

$$\mathrm{FOM} = \frac{\mathrm{n_{sig}}}{\sqrt{\mathrm{n_{sig}} + \mathrm{n_{bkg}}}} \tag{5}$$

where n_{sig} and n_{bkg} are the respective numbers of signal and background events that pass the cut. Finding the maximum of the FOM for the $0 < q^2 < 20$ GeV^2 signal region validation dataset leaves us with a cut of $p(\mathrm{signal}) > 0.65$, where $p(\mathrm{signal})$ is the output of the background suppression model. The $1 < q^2 < 19$ GeV^2 dataset gives a similar cut of $p(\mathrm{signal}) > 0.64$. After applying this cut, the ratio of background to signal events of the signal region validation dataset is about 1.1 for both the $1 < q^2 < 19$ and $0 < q^2 < 20$ GeV^2 regions. We show a histogram of the background suppression model's outputs for signal region events in figure 8. A tighter cut could achieve a background to signal ratio lower bound of 0.45 for the $1 < q^2 < 19$ GeV^2 region. After applying the background suppression model cut, our final signal efficiency for the $1 < q^2 < 19$ GeV^2 region is about 0.25.

Note that after the application of our background suppression model and cuts there is a peaking background in the M_{bc} signal region (see figure 9). By reviewing the true particle identifications of reconstructed background events in the signal region, we find that this peaking background is due to the misidentification of μ as π and vice versa. We leave the reduction of this peaking background to future work, where we plan to suppress this background using a tighter

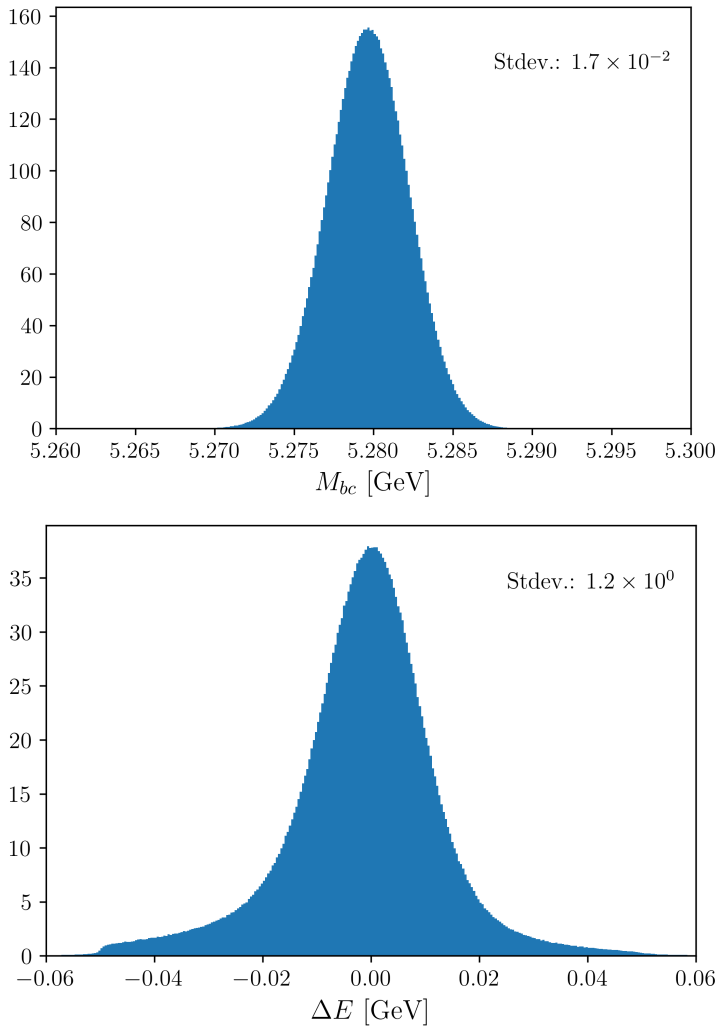

Figure 7: Signal distributions for M_{bc} and ΔE. Data includes detector effects. $-0.05 < \Delta E < 0.05$ GeV and $M_{bc} > 5.27$ GeV cuts were applied previously during reconstruction. The Y-axis is normalized.

Particle	Cut	Notes
B^0	$M_{bc} > 5.27$ GeV	Signal region
	$4.8 < M_{bc} < 5.26$ GeV	M_{bc} Sideband
	$\lvert\Delta E\rvert < 0.05$ GeV	
	p(signal) > 0.65	Applied to SR ($0 < q^2 < 20$ GeV^2)
	p(signal) > 0.64	Applied to SR ($1 < q^2 < 19$ GeV^2)
K^{*0}	$\lvert M - M_{PDG}\rvert < 2\,\Gamma$	$M_{PDG} = 0.892$ GeV, $\Gamma = 0.05$ GeV
K^+	KaonID > 0.8	
	$dr < 2$ cm	
	$\lvert dz\rvert < 4$ cm	
	$17^\circ < \theta < 150^\circ$	CDC acceptance
π^-	$dr < 2$ cm	
	$\lvert dz\rvert < 4$ cm	
	$17^\circ < \theta < 150^\circ$	CDC acceptance
μ^+, μ^-	muonID > 0.8	
	$p > 0.6$ GeV	
	$dr < 2$ cm	
	$\lvert dz\rvert < 4$ cm	
	$17^\circ < \theta < 150^\circ$	CDC acceptance
	$0 < q^2 < 20$ GeV^2	Loose q^2 veto
	$1 < q^2 < 19$ GeV^2	Tight q^2 veto
	$\neg(9 < q^2 < 10$ GeV$^2)$	J/ψ veto
	$\neg(13 < q^2 < 14$ GeV$^2)$	$\psi(2S)$ veto

Table 2: Cuts applied to detector level data. To obtain signal region events, we cut using the signal region M_{bc} cut. To obtain sideband events, we apply the M_{bc} sideband cut. The variable p(signal) is the output of the background suppression model, which is applied to the signal region datasets obtained from generic simulation. We list the background suppression model cuts for both loose and tight q^2 veto datasets. We also list both loose and tight q^2 vetos in the muon cut section. In the K^{*0} section, Γ stands for the Breit-Wigner width and M_{PDG} is the mass given by the Particle Data Group.

Particle	Variable	Notes
B^0	χ^2_{red}	Tree fitter reduced χ^2
	ΔE	
K^{*0}	$M - M_{PDG}$	$M_{PDG} = 0.892$ GeV
K^+, π^-, μ^+, μ^-	dr	
	dz	

Table 3: Input variables to background suppression model.

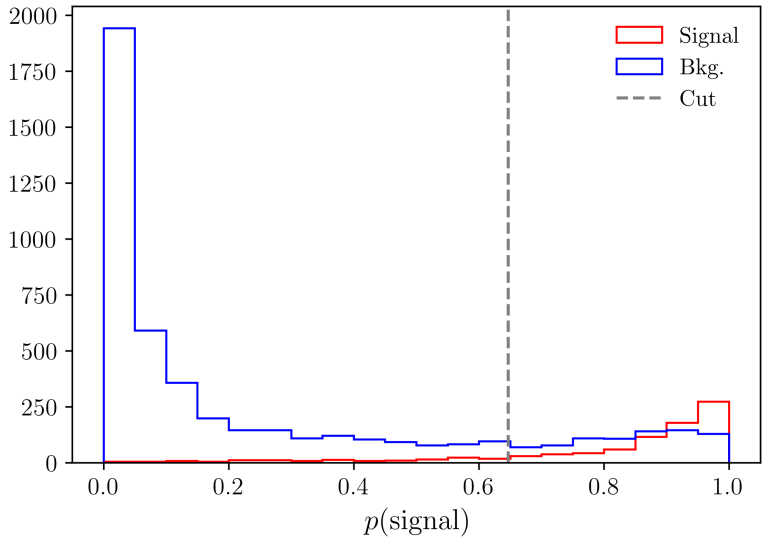

Figure 8: Background suppression model predictions on the signal region validation dataset. The dashed line indicates the lower bound of the selection cut obtained by maximizing the figure of merit. We show data from the $0 < q^2 < 20$ GeV^2 dataset.

muonID cut, a pionID cut, and the invariant mass vetos as used in [4], such as $M(\pi^-\mu^+) \notin [3.06, 3.11]$ GeV^2 to suppress $B \to J/\psi\ (\mu^+\mu^-)K^{*0}\ (K^+\pi^-)$, where the μ^- and π^- are misidentified as each other. If all μ and π misidentifications were eliminated (see the right plot in figure 9), our signal region background to signal ratio would decrease to 0.56, assuming that we apply the same cuts used for our $1 < q^2 < 19$ GeV^2 dataset.

3.3 Detector Effects

3.3.1 Efficiencies

When applying detector effects to generated data, events are lost due to event reconstruction issues and interactions with detector geometry. In order to investigate the extent of these effects, we define the efficiency as the number of detector level events divided by the number of generator level events. We bin the dataset in each decay variable and compute the efficiency for every bin. As an equation, if we refer to the efficiency of bin i of variable x as $\epsilon_{x,\,i}$, we have that

$$\epsilon_{x,\,i} \equiv \frac{n_{x,\,i,\,\text{det}}}{n_{x,\,i,\,\text{gen}}} \tag{6}$$

where $n_{x,\,i,\,\text{det}}$ is the number of reconstructed events in bin i and $n_{x,\,i,\,\text{gen}}$ is the number of generated events in bin i. The resulting plots are shown in figure 10.

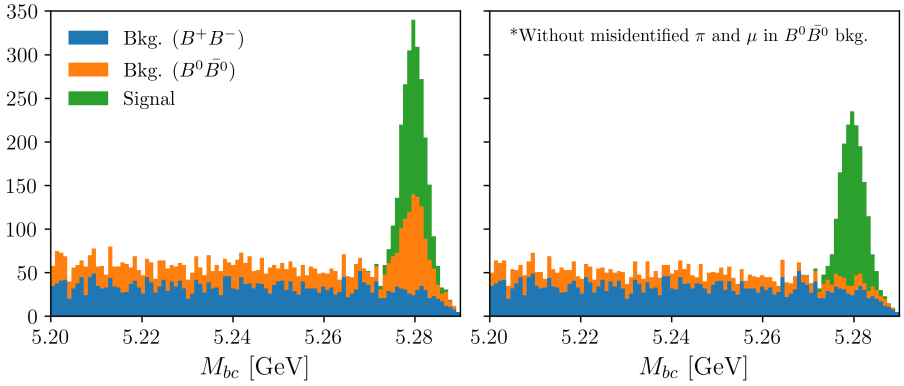

Figure 9: The M_{bc} distribution of reconstructed generic B^0 $\bar{B}^0$ and B^+ B^- decays after applying a background suppression model and cuts. In the right plot, we show the distribution without misreconstructed μ and π. We show the background and signal distributions as a stacked histogram.

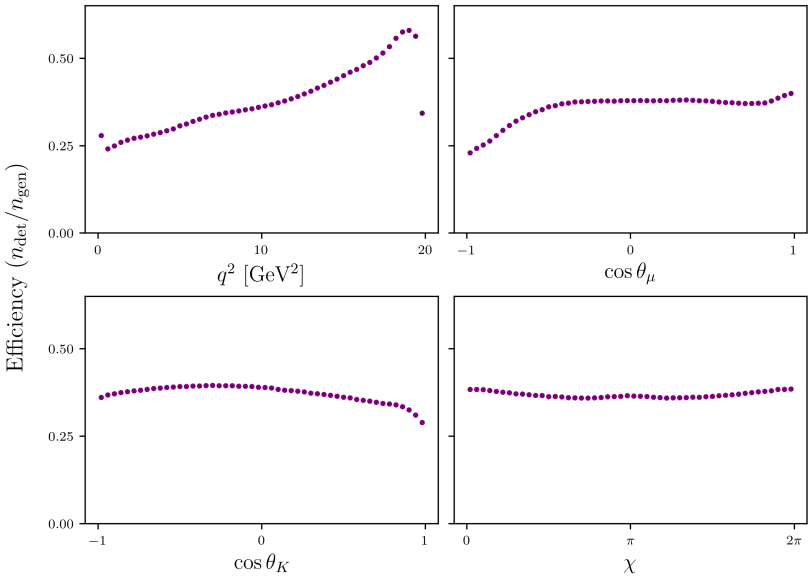

Figure 10: There are about 4.2×10^7 generator level events and 1.5×10^7 detector level events, which gives an overall efficiency of 0.37. The data shown here is before J/ψ and $\psi(2S)$ q^2 vetos and background suppression model cuts are applied, which inflates the overall efficiency. These plots include events for all values of δC_9. We use data with $0 < q^2 < 20$ GeV^2.

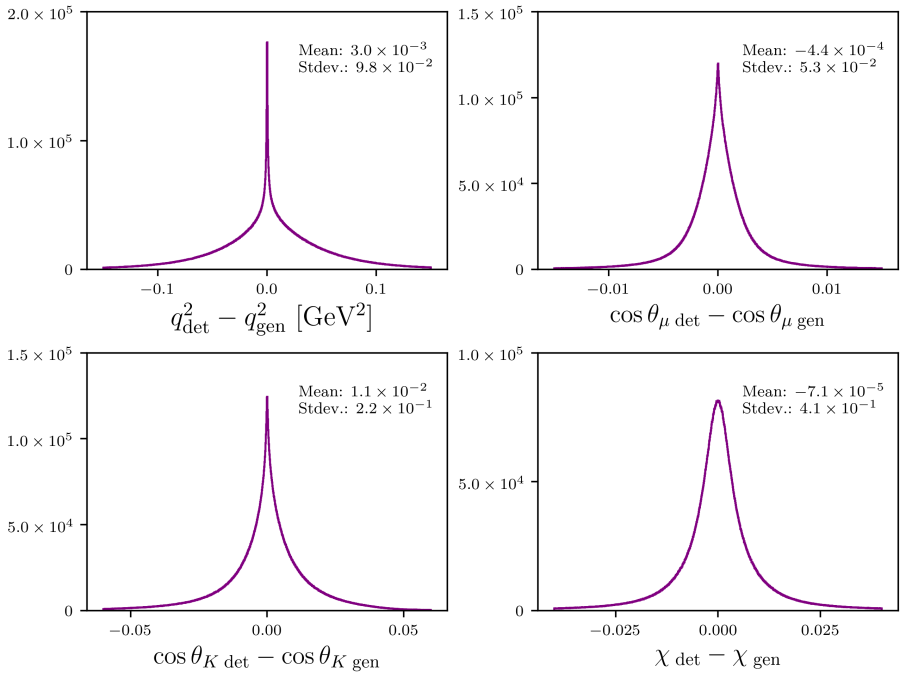

Figure 11: The y-axis represents the number of events per bin. There are about 1.5×10^7 events per histogram. We show resolutions of events from all values of δC_9. The data shown is before J/ψ and $\psi(2S)$ q^2 vetos and background suppression model cuts are applied. Note that the scale of the x-axis and y-axis are different for each plot. We use data with $0 < q^2 < 20$ GeV^2.

3.3.2 Resolutions

When detector effects are applied to generator level events, the measured values of variables change from their original generator level values. In order to investigate this difference, we define the resolution as the difference between the value of a measurement after detector effects are applied and its generator level value. For variable x of a certain event, the resolution would be $x_{\text{det}} - x_{\text{gen}}$. We calculate the resolution of each decay variable for all events that pass reconstruction and plot histograms of the results in figure 11.

3.4 Asymmetries

Asymmetries such as A_{FB} and S_5 depend on beyond the Standard Model (BSM) physics parameters. Although we do not input these asymmetries directly into the models, they are sensitive intermediate variables that a neural network could learn to compute as functions of its inputs. In figure 12 we plot A_{FB} and S_5 over multiple δC_9 values to show that these variables are sensitive to changes

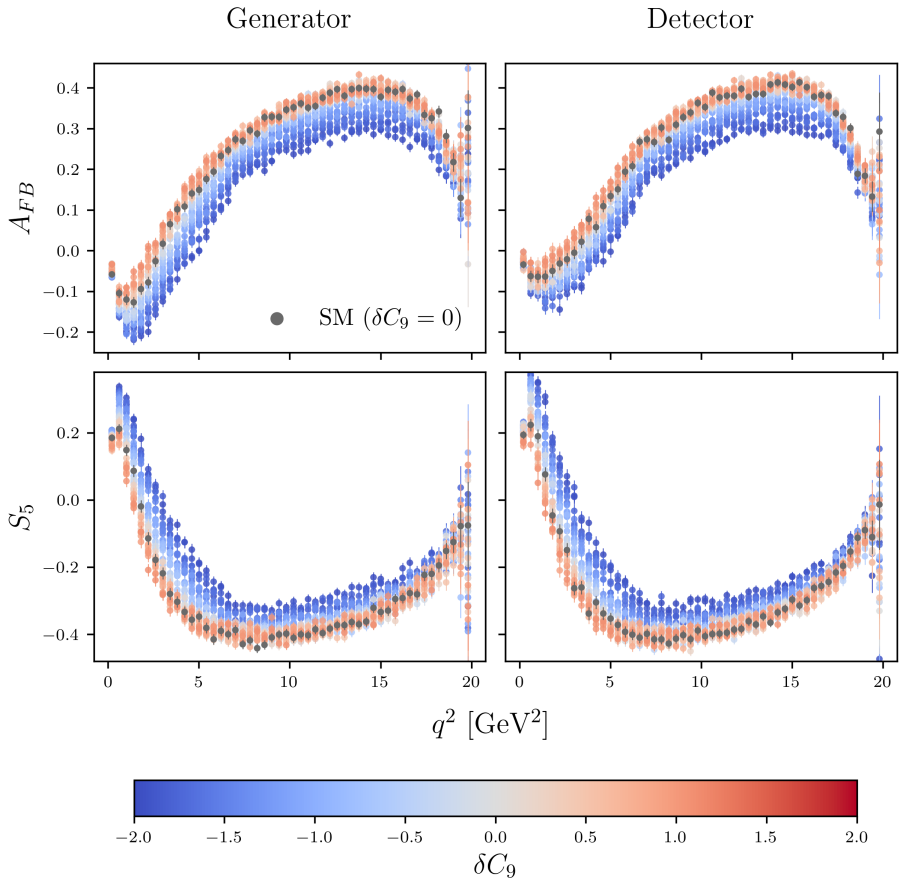

Figure 12: Asymmetries A_{FB} and S_5 are plotted here. Color represents δC_9 value. Asymmetries are computed over q^2 bins (x-axis). The same number of generator and detector level events are used. The deviation $\delta C_9 \in [-2.0, 1.1]$ is plotted. We use data with $0 < q^2 < 20$ GeV^2. We also note that the A_{FB} and S_5 shifts are anti-correlated.

in δC_9 at both the generated level and after detector effects are added. Here, we compute A_{FB} as

$$A_{FB} = \frac{n_{\text{forward}} - n_{\text{backward}}}{n_{\text{forward}} + n_{\text{backward}}} \tag{7}$$

where n_{forward} is the number of events with $0 < \cos\theta_\mu < 1$ and n_{backward} is the number of events with $-1 < \cos\theta_\mu < 0$. Similarly, we compute S_5 as

$$S_5 = \frac{4}{3}\,\frac{n_{\text{forward}} - n_{\text{backward}}}{n_{\text{forward}} + n_{\text{backward}}} \tag{8}$$

where n_{forward} is given by

$$\begin{aligned} n_{\text{forward}} = |[&(0 < \cos\theta_K < 1) \cap (0 < \chi < \frac{\pi}{2})] \\ &\cup [(0 < \cos\theta_K < 1) \cap (\frac{3\pi}{2} < \chi < 2\pi)] \\ &\cup [(-1 < \cos\theta_K < 0) \cap (\frac{\pi}{2} < \chi < \frac{3\pi}{2})]| \end{aligned} \tag{9}$$

and n_{backward} is given by

$$\begin{aligned} n_{\text{backward}} = |[&(0 < \cos\theta_K < 1) \cap (\frac{\pi}{2} < \chi < \frac{3\pi}{2})] \\ &\cup [(-1 < \cos\theta_K < 0) \cap (0 < \chi < \frac{\pi}{2})] \\ &\cup [(-1 < \cos\theta_K < 0) \cap (\frac{3\pi}{2} < \chi < 2\pi)]| \end{aligned} \tag{10}$$

where the vertical lines denote the number of elements in a set.

4 Models

We train three types of neural networks to fit for δC_9 given a dataset of $B^0 \to K^{*0}\mu^+\mu^-$ (and $\bar{B}^0 \to \bar{K}^{*0}\mu^-\mu^+$) events. The three networks are, (1) the residual convolutional neural network [12], (2) the Deep Sets neural network [17], and, (3) what we refer to as the event-by-event neural network. In this section, we describe each model in detail, including any preprocessing of the model's inputs.

4.1 Overview

4.1.1 Convolutional Neural Network

S. Dubey et al. showed [10] that we can predict the δC_9 value used to generate a $B^0 \to K^{*0}\mu^+\mu^-$ dataset by transforming the dataset into a three-dimensional image and operating on the image using a residual convolutional neural network [12] (CNN) regression model. Here, we extend the work of S. Dubey et al. by training and evaluating CNNs on data that include detector simulation and backgrounds.

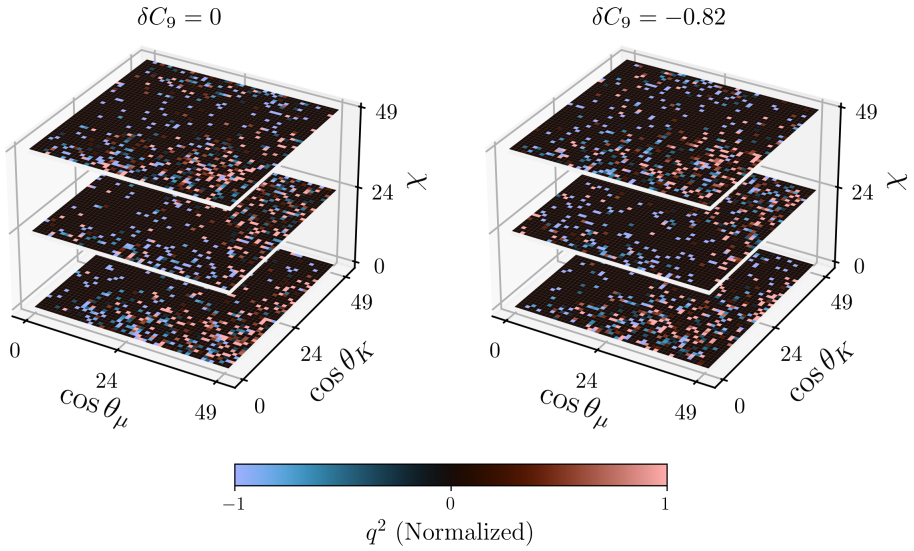

Figure 13: Visualizations of example images. Color represents the average q^2 value per angular bin. Images have 32,000 generator level signal events. Only three slices in χ are shown. Axes are labeled with bin indices. The q^2 values are standard scaled.

An image is created by binning a dataset in its three angular variables, $\cos\theta_\mu$, $\cos\theta_K$, and χ, and computing the average q^2 per bin. Example images are visualized in figure 13. Angular bins without any events are set to zero. We create images with 50 bins per dimension. Before image creation, q^2 values are standard scaled by subtracting the training dataset q^2 mean and dividing by the training dataset q^2 standard deviation. In order to create an image dataset, we bootstrap (sample with replacement) sets of events from the corresponding source dataset.

Our CNN model architecture is as close as possible to the architecture used by S. Dubey et al. This is for ease of comparison. We also find that this architecture performs better than the shallower alternative architectures that we tried. The model consists of 34 total layers, including three-dimensional convolutional layers that range from 64 output channels to 512 output channels. We use skip connections and batch normalization to train this deep network. We use ReLU activations after each hidden layer. We use a kernel size of 3 for the majority of the network, except for the initial convolution layer where we use a kernel size of 7. We train using a mean squared error loss (in contrast to the mean absolute error loss used by S. Dubey et al. [10]).

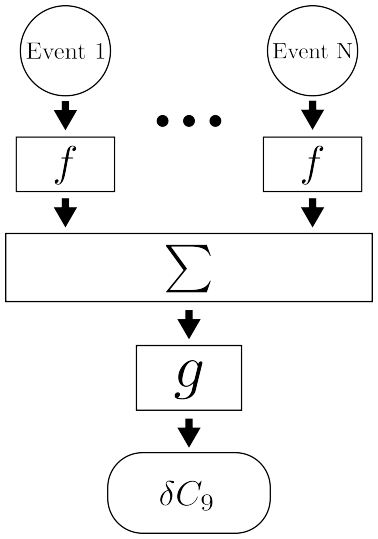

Figure 14: Diagram of the deep sets neural network operating on a set of N events. f and g are neural networks.

4.1.2 Deep Sets

The second type of network we try is the Deep Sets neural network [17]. See figure 14 for a diagram. One way to think about this model is as an unbinned version of the previous CNN method. The Deep Sets model takes as input a set of events, where each event is represented as a vector of its variables q^2, $\cos\theta_\mu$, $\cos\theta_K$, and χ. The model is composed of two neural networks, which we call f and g. If the ith event of an input dataset of N events is x_i, the deep sets model outputs $g(\sum_i^N f(x_i))$. Network f takes an event as input and outputs a new vector representation of that event. These new representations are then summed over the set of events. The sum is input into network g which outputs a regression prediction for δC_9. The summation of event representations enforces invariance over permutations of the input dataset, which eliminates the need for the model to learn that permutations of an input set have the same δC_9 value. Each input variable is standard scaled by subtracting the training dataset average and dividing by the training dataset standard deviation. As with the CNN, to create a dataset of sets of events, we bootstrap from a source dataset.

Our Deep Sets neural network contains 16 total layers. Networks f and g (as discussed previously) both have 8 layers. We begin network f with a layer of output size 32 and scale to a final output layer size of 128. Network g begins with a layer of output size 128 and scales down to a final output size of 1 (the δC_9 prediction). We use a ReLU activation after each hidden layer. We train using a mean squared error loss.

4.1.3 Event-by-event

The third network is what we are calling the event-by-event neural network. It is a simple multilayer perceptron that takes as input a single event represented by variables $\cos\theta_\mu$, $\cos\theta_K$, χ, and q^2. The network is trained to output a discrete probability distribution over binned δC_9 values, $p(\delta C_9 \mid x)$ where x is an event. This is a classification task, as opposed to the CNN and Deep Sets models, which perform regression. In order to obtain a prediction given multiple events, we combine the per event probability distributions using the following equation

$$p(\delta C_9 \mid x_1, ..., x_N) = \frac{\prod_{i=1}^{N} p(\delta C_9 \mid x_i)}{\sum_{\delta C_9'} \prod_{i=1}^{N} p(\delta C_9' \mid x_i)} \tag{11}$$

Here, x_i represents the ith event and δC_9 represents a value of δC_9 from the training dataset. We show the derivation of equation 11 in section 4.2.3. The derivation involves applying Bayes' rule, assuming events from the same δC_9 are independent, and assuming a uniform prior. However, non-uniform priors are also possible. We can also obtain a point prediction of δC_9 by calculating the expected value of the predicted probability distribution. The event-by-event network is trained on a dataset of events (bootstrapping is unnecessary).

We standard scale each input variable by subtracting its training dataset average and dividing by its training dataset standard deviation. We also find that because we assume a uniform prior distribution, it is necessary to have the same number of examples per δC_9 label in the training dataset. If the dataset is unbalanced, the model learns to predict higher probabilities for more often occurring δC_9 values and lower probabilities for less frequent δC_9 values. To balance the training dataset we reduce the number of examples per δC_9 label to the minimum number of examples over δC_9 labels. Because we simulate approximately the same number of events per δC_9 label, this does not have a large effect on our dataset. However, in a case where the original dataset is very unbalanced, scaling the gradients of examples during training might be a better solution.

The event-by-event model consists of a total of 7 layers. The first layer has an output size of 16. The layer output sizes increase until the second to last layer, which has an output size of 64. The final layer has an output size of 44 (one probabilistic prediction per δC_9 bin). Each hidden layer is followed by a ReLU activation. We train using a categorical cross entropy loss.

Note that current neural simulation-based inference approaches [9] tend to predict an unbinned per event probability distribution, using, for example, normalizing flow models [15]. However, unbinned predictions change the denominator of equation 11 into an intractable integral. We present the binned approach as a simple alternative.

4.1.4 Set-based vs. Event-based

We can further categorize the three models into two useful categories, which we call set-based models and event-based models. See figure 15 for a diagram. Set-

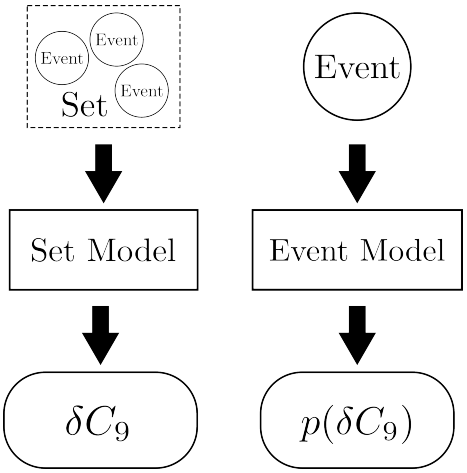

Figure 15: Set-based neural networks (CNN, Deep Sets) versus event-based neural networks.

based models take as input a set of events. Event-based models take as input a single event. Both the CNN and Deep Sets neural networks fall into the set-based category because they are trained to predict δC_9 given a representation of an entire dataset. The event-by-event neural network is trained to predict a probability distribution over δC_9 given a single event, so it falls into the event-based category.

4.2 Details

4.2.1 Hyperparameters

Here we discuss the hyperparameters of each neural network. Important hyperparameters are listed in table 4, including the number of trainable parameters per model. For the sake of simplifying comparison, we use these same hyperparameters when training models on data at all three levels of simulation (generator, detector, and detector and background). The training of all models is stabilized using a learning rate scheduler that decreases the learning rate when the validation loss has plateaued or started to increase.

For set-based models, a different model is required for each input dataset size. This is because each model is trained on examples that contain a certain number of events. There is no guarantee that a model trained to predict δC_9 given inputs of 20,000 events would generalize to inputs of 10,000 events. Here, we train CNN and Deep Sets models at input sizes of 8,000, 16,000, and 32,000 signal events, which approximately correspond to Belle II integrated luminosities of 25 ab^{-1}, 50 ab^{-1}, and 100 ab^{-1}. When training these models, we vary the batch size and number of bootstrapped examples over the input dataset sizes, and we list these values in table 5.

Hyperparameter	CNN	DS	EBE
Parameters	6.8×10^7	1.0×10^5	1.1×10^4
Layers	34	16	7
Epochs	50	50	300
Learning Rate (LR)	1×10^{-3}	1×10^{-3}	3×10^{-3}
LR Reduction Factor	0.2	0.2	0.95
LR Reduction Patience	5	5	0
Batch Size	See table 5	See table 5	1×10^4

Table 4: Comparison of training hyperparameters across the three models. LR Reduction Patience refers to the number of epochs the validation loss can increase before the learning rate scheduler applies a learning rate reduction. LR Reduction Factor refers to the fraction by which the learning rate is multiplied when it is reduced.

Hyperparameter	8×10^3 events	1.6×10^4 events	3.2×10^4 events
Examples/δC_9	400	200	100
Batch Size	128	64	32

Table 5: Hyperparameters of set-based (CNN and Deep Sets) models that vary over input dataset size (the number of signal events per bootstrapped set). Examples/δC_9 refers to the number of bootstrapped sets per δC_9 label.

Hyperparameter	CNN	DS	EBE
Layers	[9, 34]	[10, 16]	[3, 7]
Epochs	[20, 100]	[20, 100]	[100, 300]
Learning Rate (LR)	[3e-4, 1e-3]	[3e-4, 1e-3]	[1e-5, 1e-2]
LR Reduction Factor	[0.1, 0.9]	[0.1, 0.9]	[0.5, 0.95]
LR Reduction Patience	[0, 5]	[0, 5]	[0, 3]
Batch Size	[32, 128]	[32, 128]	[32, 1e6]

Table 6: Approximate hyperparameter search intervals. Hyperparameter optimization was performed by hand. Batch size search intervals were limited by amount of hardware memory. Scientific notation is abbreviated for compactness (e.g. 3e-4 is 3×10^{-4}).

We conduct all hyperparameter optimization by hand. See table 6 for a list of approximate search intervals. Further hyperparameter optimization could lead to improved model performance. We find that for the CNN and Deep Sets models, a high initial learning rate and aggressive learning rate scheduler lead to a lower validation loss than a slower learning rate and more gradual scheduler. This is at the cost of a higher rate of model training failure. For the event-by-event network, using a gradual learning rate scheduler to slowly decrease the learning rate, while training for an extended number of epochs, is necessary to achieve fit convergence. A large training batch size (i.e. number of simulation samples per parameter update) also seems important for the model to have enough information to learn.

4.2.2 Backgrounds

When training on detector and background level datasets, we create set-based model training examples by bootstrapping sets of background events and adding these background events to detector level signal sets. For ease of comparison, we keep the number of signal events per set constant with and without backgrounds. In other words, models trained with backgrounds also have 8,000, 16,000, and 32,000 signal events per set in addition to a certain number of background events. We use a background to signal ratio of approximately 1.1. We obtain this value from the background to signal ratio of the signal region validation dataset after background suppression requirements are applied. We train on backgrounds from the M_{bc} sideband.

When training the event-based event-by-event model on detector and background level data, we upsample (sample with replacement) our M_{bc} sideband background dataset so that it has the same number of events as our signal dataset. We then combine the background and signal datasets. In order to teach the model that it should predict a uniform distribution given a background event, we label background events with random δC_9 bin labels.

4.2.3 Event-by-event Derivation

The derivation of equation 11 is as follows. We can rewrite $p(\delta C_9 \,|\, x_1, ..., x_N)$ using Bayes' rule.

$$p(\delta C_9 \,|\, x_1, ..., x_N) = \frac{p(x_1, ..., x_N \,|\, \delta C_9) p(\delta C_9)}{p(x_1, ..., x_N)} \tag{12}$$

Now, since events are sampled independently given that we know the value of δC_9, the likelihood becomes:

$$\begin{aligned} p(x_1, ..., x_N \,|\, \delta C_9) &= p(x_1 \,|\, \delta C_9) p(x_2 \,|\, x_1, \delta C_9) ... && (13) \\ &= p(x_1 \,|\, \delta C_9) p(x_2 \,|\, \delta C_9) ... && (14) \\ &= \prod_{i=1}^{N} p(x_i \,|\, \delta C_9) && (15) \end{aligned}$$

Applying Bayes' rule again gives

$$p(x_1, ..., x_N \mid \delta C_9) = \prod_{i=1}^{N} \frac{p(\delta C_9 \mid x_i) p(x_i)}{p(\delta C_9)} \tag{16}$$

Applying the conditional independence to the normalization term gives

$$\begin{aligned} p(x_1, ..., x_N) &= \int_{\delta C_9'} p(x_1, ..., x_N, \delta C_9') && (17)\\ &= \int_{\delta C_9'} p(\delta C_9') p(x_1 \mid \delta C_9') p(x_2 \mid x_1, \delta C_9') ... && (18)\\ &= \int_{\delta C_9'} p(\delta C_9') p(x_1 \mid \delta C_9') p(x_2 \mid \delta C_9') ... && (19)\\ &= \int_{\delta C_9'} p(\delta C_9') \prod_{i=1}^{N} p(x_i \mid \delta C_9') && (20)\\ & && (21) \end{aligned}$$

And applying Bayes' rule again we obtain

$$p(x_1, ..., x_N) = \int_{\delta C_9'} p(\delta C_9') \prod_{i=1}^{N} \frac{p(\delta C_9' \mid x_i) p(x_i)}{p(\delta C_9')} \tag{22}$$

Now inserting the likelihood and normalization terms into our original equation gives

$$p(\delta C_9 \mid x_1, ..., x_N) = \frac{(\prod_{i=1}^{N} \frac{p(\delta C_9 \mid x_i) p(x_i)}{p(\delta C_9)}) p(\delta C_9)}{\int_{\delta C_9'} p(\delta C_9') \prod_{i=1}^{N} \frac{p(\delta C_9' \mid x_i) p(x_i)}{p(\delta C_9')}} \tag{23}$$

We can factor $\prod_{i=1}^{N} p(x_i)$ out of the numerator and denominator to get

$$\begin{aligned} p(\delta C_9 \mid x_1, ..., x_N) &= \frac{(\prod_{i=1}^{N} p(x_i))(\prod_{i=1}^{N} \frac{p(\delta C_9 \mid x_i)}{p(\delta C_9)}) p(\delta C_9)}{(\prod_{i=1}^{N} p(x_i)) \int_{\delta C_9'} p(\delta C_9') \prod_{i=1}^{N} \frac{p(\delta C_9' \mid x_i)}{p(\delta C_9')}} && (24)\\ &= \frac{p(\delta C_9)(\prod_{i=1}^{N} \frac{p(\delta C_9 \mid x_i)}{p(\delta C_9)})}{\int_{\delta C_9'} p(\delta C_9') \prod_{i=1}^{N} \frac{p(\delta C_9' \mid x_i)}{p(\delta C_9')}} && (25) \end{aligned}$$

Assuming that the output is a binned probability distribution over δC_9, the integral in the denominator turns into a sum.

$$p(\delta C_9 \mid x_1, ..., x_N) = \frac{p(\delta C_9)(\prod_{i=1}^{N} \frac{p(\delta C_9 \mid x_i)}{p(\delta C_9)})}{\sum_{\delta C_9'} p(\delta C_9') \prod_{i=1}^{N} \frac{p(\delta C_9' \mid x_i)}{p(\delta C_9')}} \tag{26}$$

Finally, assuming a uniform prior gives

$$p(\delta C_9 \mid x_1, ..., x_N) = \frac{\prod_{i=1}^{N} p(\delta C_9 \mid x_i)}{\sum_{\delta C_9'} \prod_{i=1}^{N} p(\delta C_9' \mid x_i)} \tag{27}$$

5 Results

Here we discuss results obtained for each of the three types of neural network models. As discussed in section 3, we show validation results for models trained on the $0 < q^2 < 20$ GeV^2 veto dataset and test results for models trained on the $1 < q^2 < 19$ GeV^2 veto dataset. The results for the looser q^2 veto validation dataset demonstrate the potential of models when given a wider q^2 distribution. The results for the tighter q^2 veto test dataset show how each model performs under a small systematic distribution shift. We expect that although physics and detector simulations can be accurate, the true distributions of the input variables, as measured in a real experiment, will be different from the simulated distributions seen by a model during training. The robustness of a model to small distribution shifts or systematic errors is therefore critical when choosing a model for simulation based inference.

5.1 Linearity Plots

Here, we show plots of predictions as a function of true labels. We refer to these plots as linearity plots because a perfect predictor would have a slope of one and a y-intercept of zero. In order to make predictions, we bootstrap datasets from each simulated value of δC_9, and apply the models to these sampled datasets. We do this for the three levels of simulation (generator, detector, and detector and background), as well as for three dataset sizes. For dataset sizes of 32,000 , 16,000, and 8,000 signal events (about 100 ab^{-1}, 50 ab^{-1}, and 25 ab^{-1}), we bootstrap 100, 200, and 400 datasets per δC_9 label respectively.

Note that for the CNN and Deep Sets models, each plot represents a different model because set-based models are trained to receive a certain dataset size. In contrast, each row of the event-by-event plot grid represents a different model, since the event-by-event model can input a dataset of any size.

In addition, the event-by-event model outputs binned probability distributions. In order to compare this model with the CNN and Deep Sets models, we plot the expectation value of the binned probability distribution.

5.1.1 $0 < q^2 < 20$ GeV^2 Validation Results

In figures 16-18 we show linearity plots for predictions made on the $0 < q^2 < 20$ GeV^2 validation dataset.

5.1.2 $1 < q^2 < 19$ GeV^2 Test Results

In figures 19-21 we show linearity plots for predictions made on the $1 < q^2 < 19$ GeV^2 test dataset.

5.2 Errors

We calculate both the mean squared error and mean absolute error of a model's predictions and plot the errors as functions of dataset size. Here we define mean

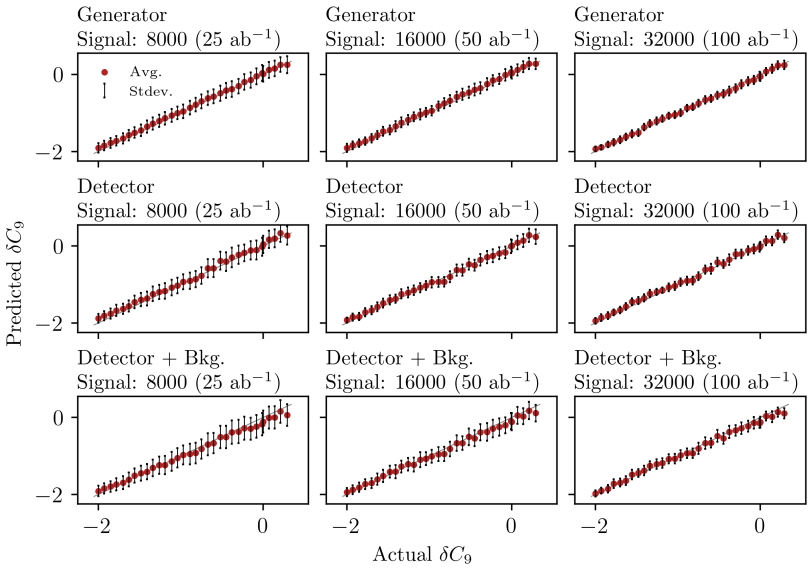

Figure 16: CNN, $0 < q^2 < 20$ GeV^2 validation linearity plots.

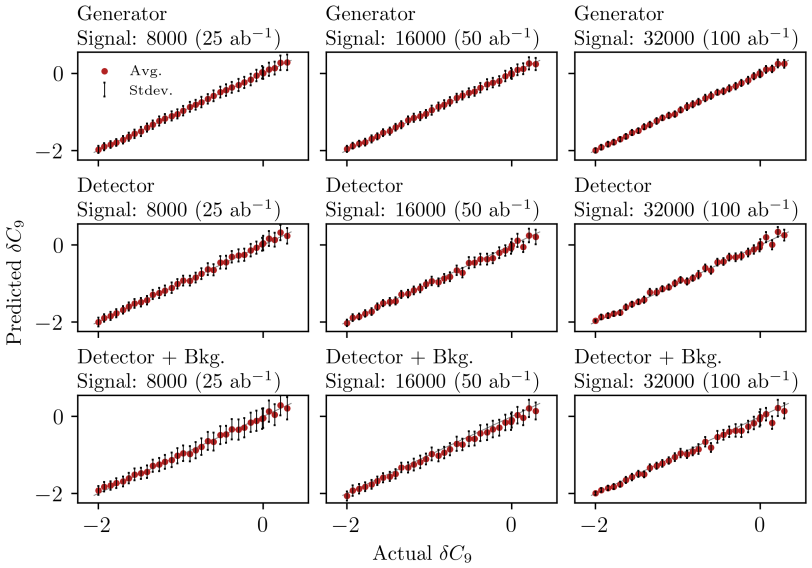

Figure 17: Deep Sets, $0 < q^2 < 20$ GeV^2 validation linearity plots.

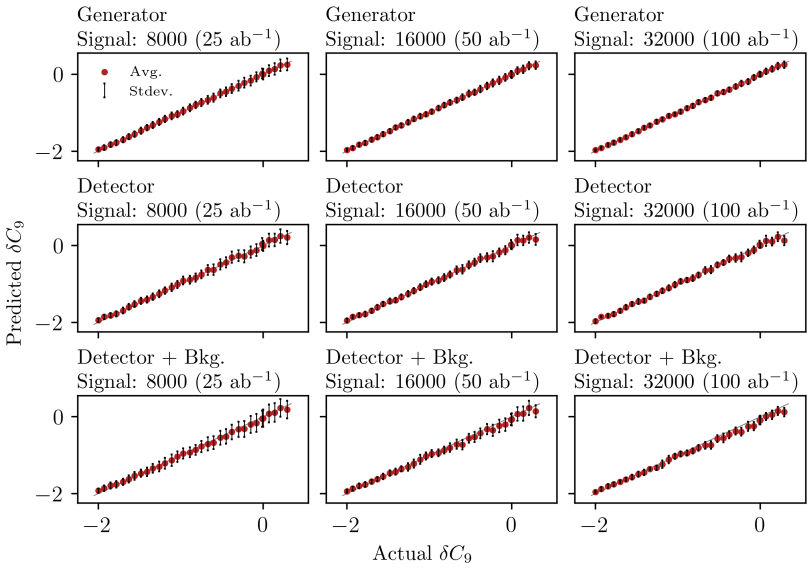

Figure 18: Event-by-event, $0 < q^2 < 20$ GeV^2 validation linearity plots.

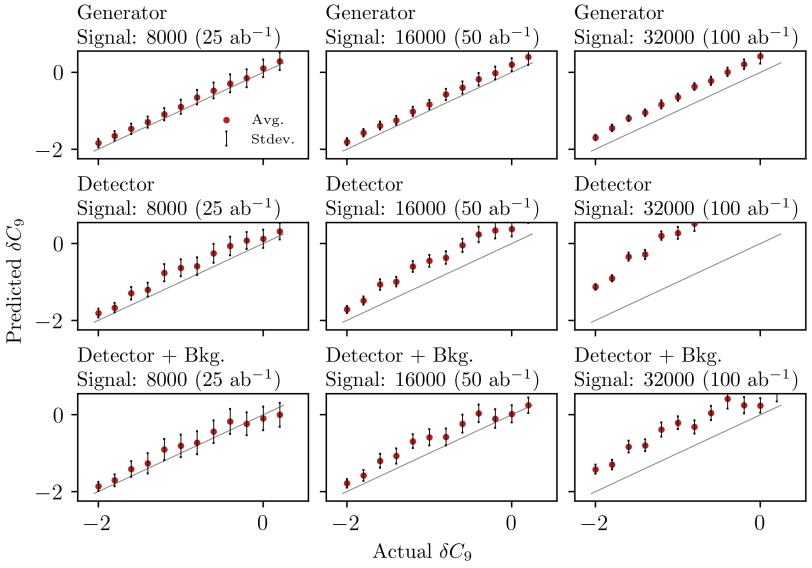

Figure 19: CNN, $1 < q^2 < 19$ GeV^2 test linearity plots. We attribute the large biases to a systematic shift between the q^2 training and test distributions. We discuss this further in section 6.

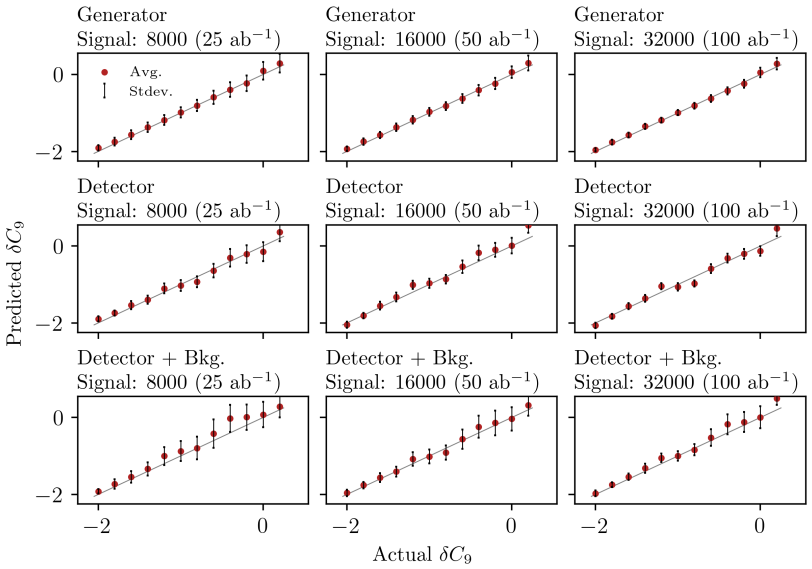

Figure 20: Deep Sets, $1 < q^2 < 19$ GeV^2 test linearity plots.

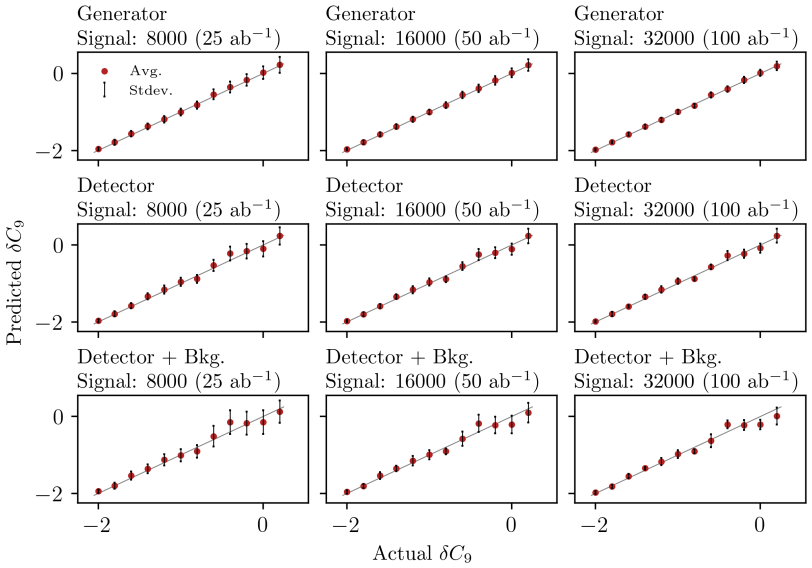

Figure 21: Event-by-event, $1 < q^2 < 19$ GeV^2 test linearity plots.

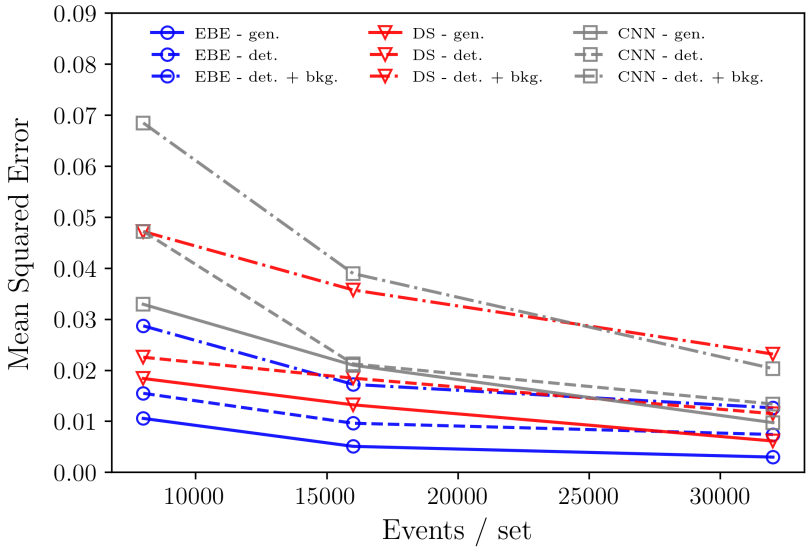

Figure 22: Mean squared error, $0 < q^2 < 20$ GeV^2 validation plot. Adding detector smearing and background reduces the sensitivity of each method.

squared error as

$$\mathrm{MSE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}}^{\mathrm{N}} (\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i})^2 \tag{28}$$

and mean absolute error as

$$\mathrm{MAE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}}^{\mathrm{N}} |\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i}| \tag{29}$$

for a dataset of N prediction-label pairs, where $\hat{y}_i$ is the ith prediction and y_i is the ith label.

5.2.1 $0 < q^2 < 20$ GeV^2 Validation Results

In figures 22-23 we show error plots for predictions on the $0 < q^2 < 20$ GeV^2 validation dataset.

5.2.2 $1 < q^2 < 19$ GeV^2 Test Results

In figures 24-25 we show error plots for predictions on the $1 < q^2 < 19$ GeV^2 test dataset.

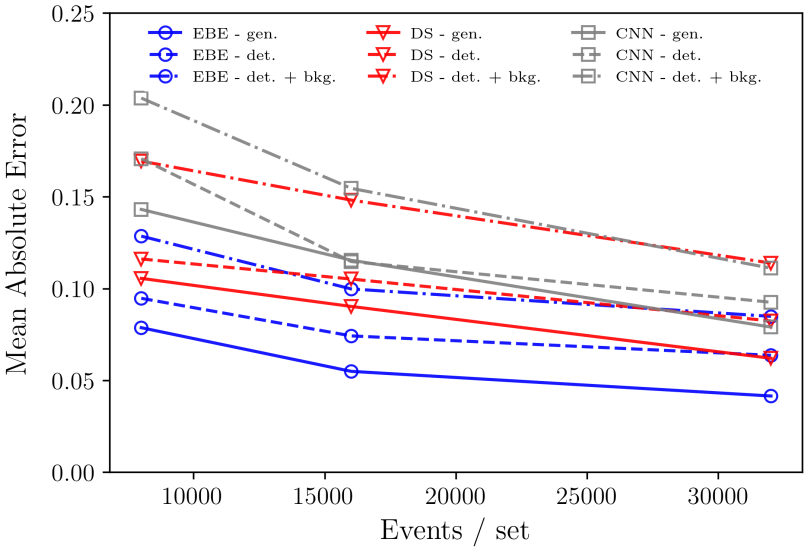

Figure 23: Mean absolute error, $0 < q^2 < 20$ GeV^2 validation plot.

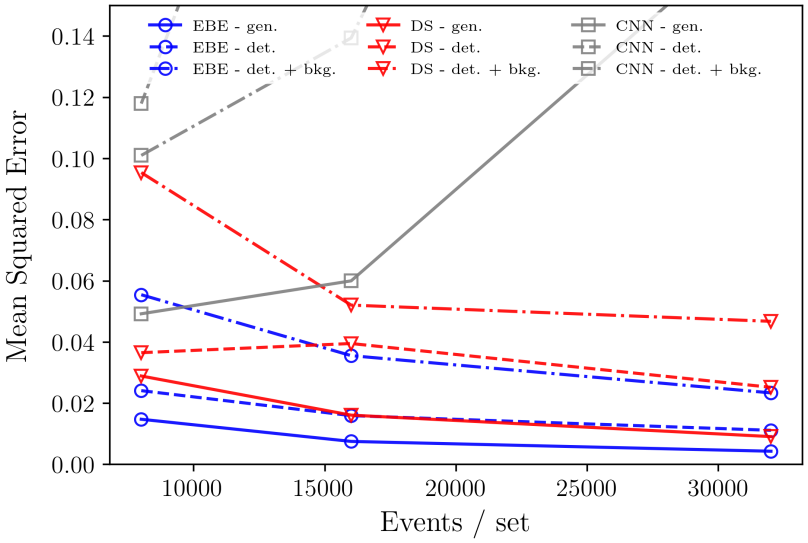

Figure 24: Mean squared error, $1 < q^2 < 19$ GeV^2 test plot.

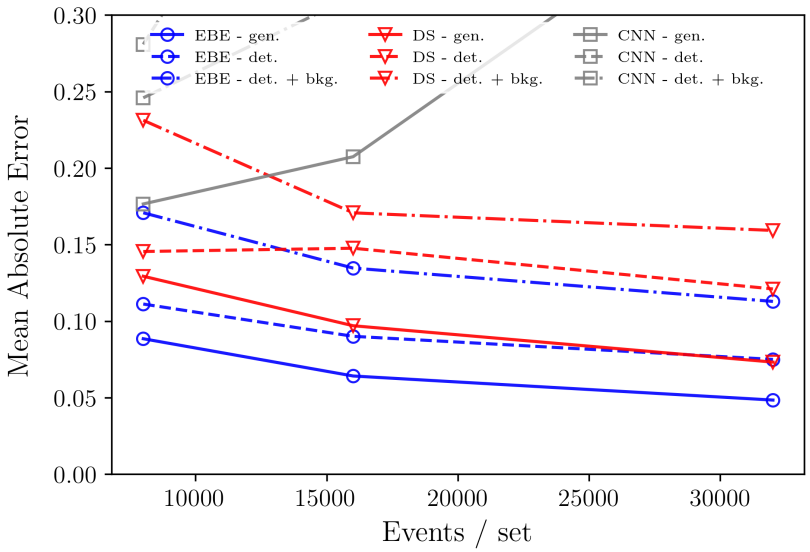

Figure 25: Mean absolute error, $1 < q^2 < 19$ GeV^2 test plot.

5.3 Predictions for New Physics Scenarios

Here we show results of model predictions at a possible δC_9 new physics (NP) value. For the $0 < q^2 < 20$ GeV^2 validation dataset, we use a new physics value of $\delta C_9 = -0.82$, and for the $1 < q^2 < 19$ GeV^2 test dataset, we use $\delta C_9 = -0.8$. In order to make predictions, we bootstrap datasets from the corresponding source dataset, and we make predictions on the bootstrapped datasets. We bootstrap 2,000 datasets at each dataset size and simulation level.

We show histograms of predictions, and we plot the standard deviation and bias of the predictions as functions of dataset size. We calculate the bias of a prediction distribution as the difference of its average and the true δC_9 label.

5.3.1 $0 < q^2 < 20$ GeV^2 Validation Results

In figures 26-28, we plot prediction distributions for $\delta C_9 = -0.82$ for the validation dataset. In figures 29 and 30, we plot the standard deviation (sensitivity) and bias of the predictions as functions of dataset size.

5.3.2 $1 < q^2 < 19$ GeV^2 Test Results

In figures 31-33, we plot prediction distributions for $\delta C_9 = -0.8$ for the test dataset. In figures 34 and 35, we plot the standard deviation (sensitivity) and bias of the predictions as functions of dataset size.

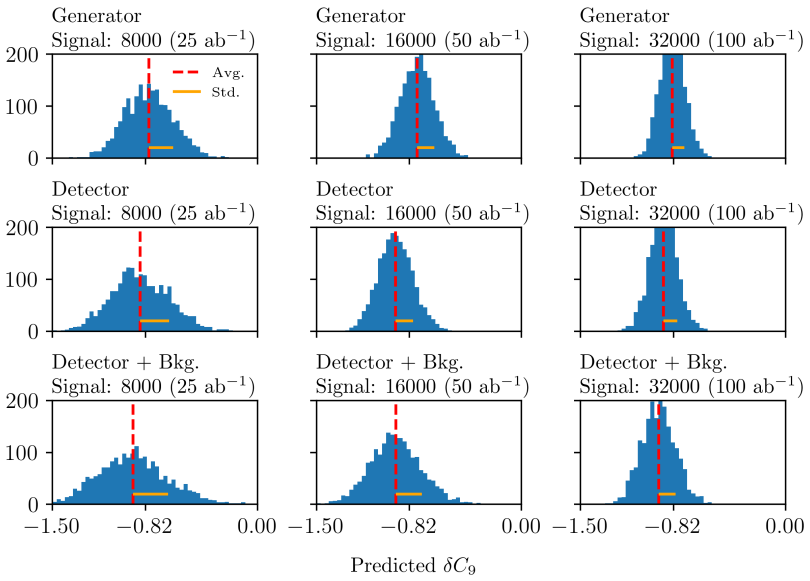

Figure 26: CNN, $0 < q^2 < 20$ GeV^2 validation NP prediction plots.

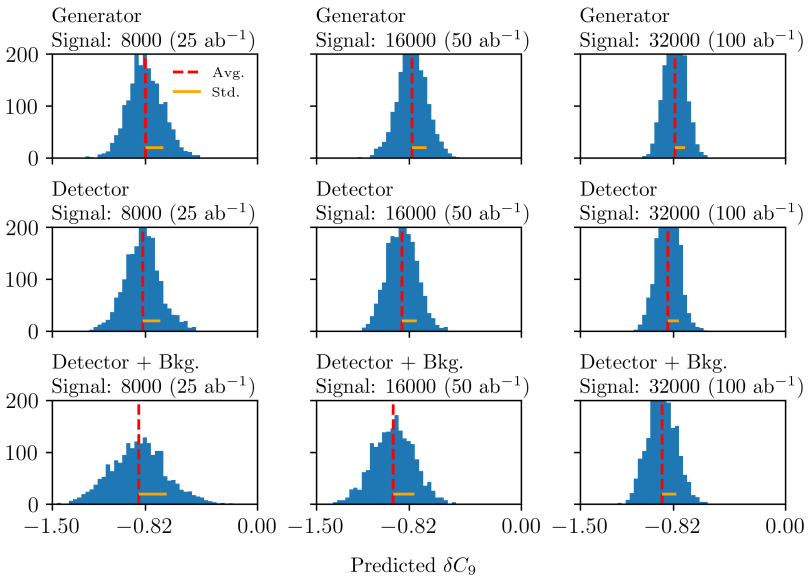

Figure 27: Deep Sets, $0 < q^2 < 20$ GeV^2 validation NP prediction plots.

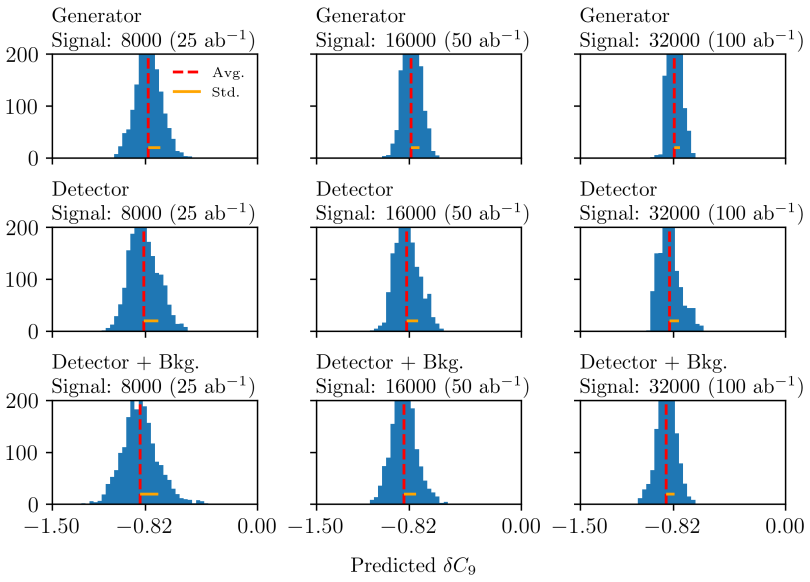

Figure 28: Event-by-event, $0 < q^2 < 20$ GeV^2 validation NP prediction plots.

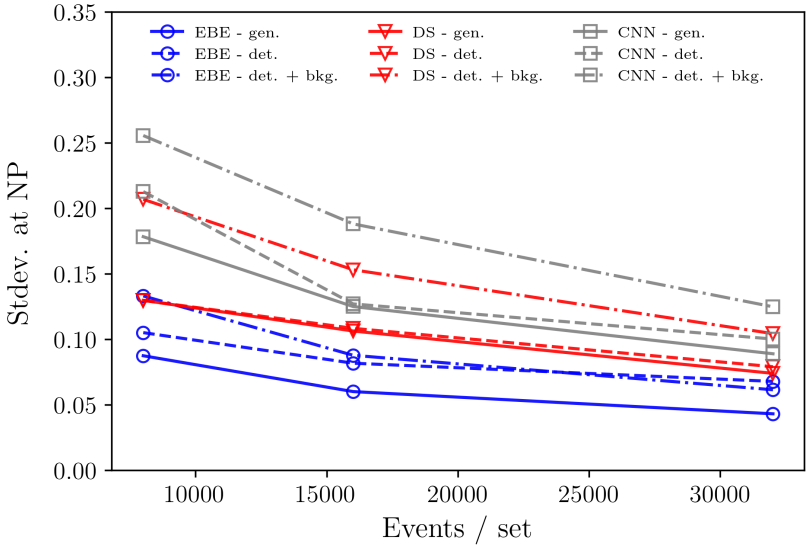

Figure 29: Sensitivity, $0 < q^2 < 20$ GeV^2 validation plot.

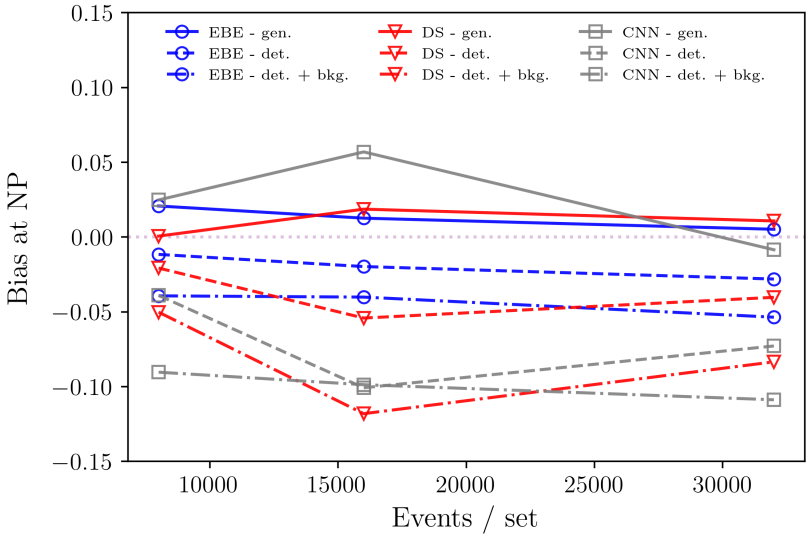

Figure 30: Bias, $0 < q^2 < 20$ GeV^2 validation plot.

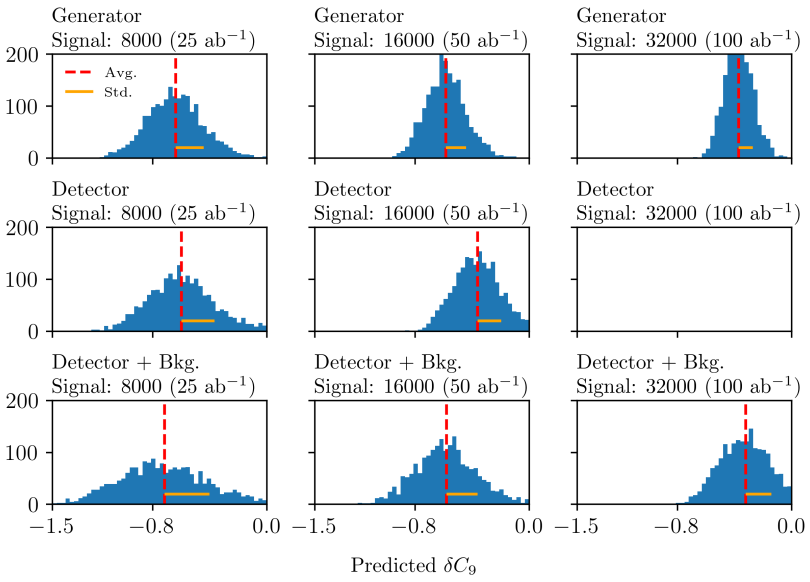

Figure 31: CNN, $1 < q^2 < 19$ GeV^2 test NP prediction plots. The missing distribution is outside of the plotting interval. We observe large biases here, again due to the systematic shift in the distribution of the test dataset.

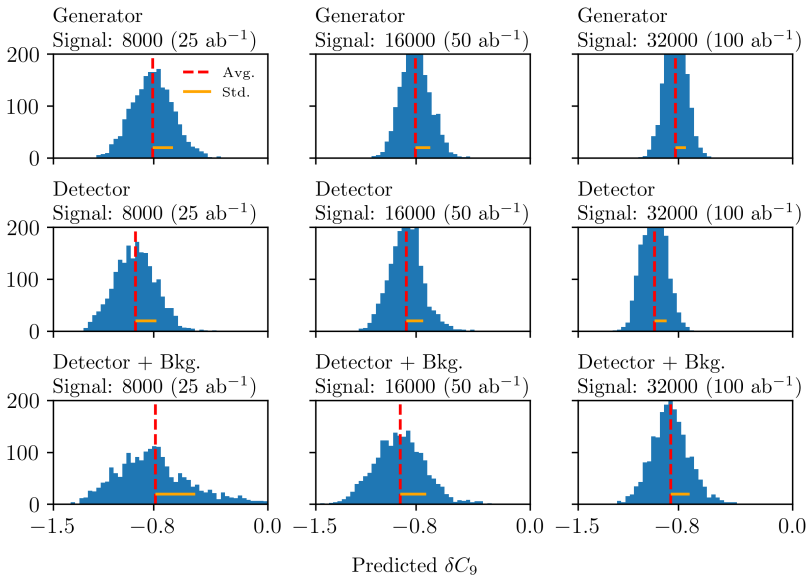

Figure 32: Deep Sets, $1 < q^2 < 19$ GeV^2 test NP prediction plots.

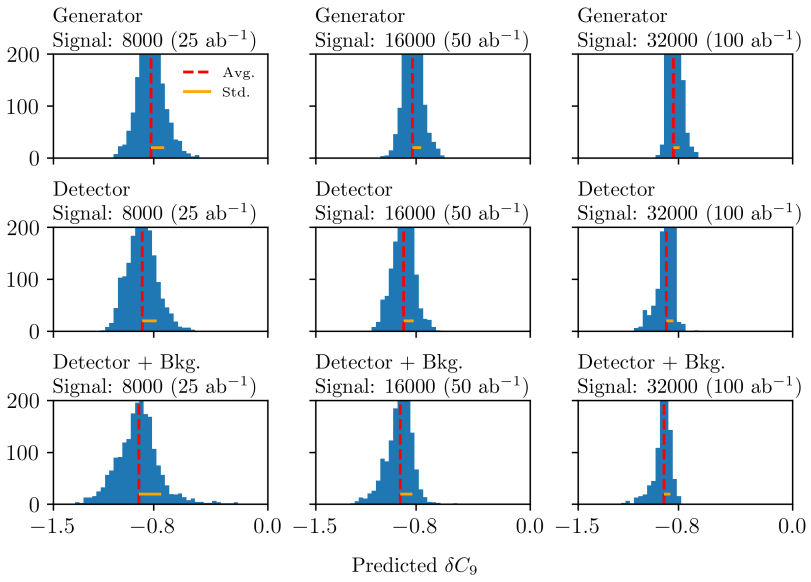

Figure 33: Event-by-event, $1 < q^2 < 19$ GeV^2 test NP prediction plots.

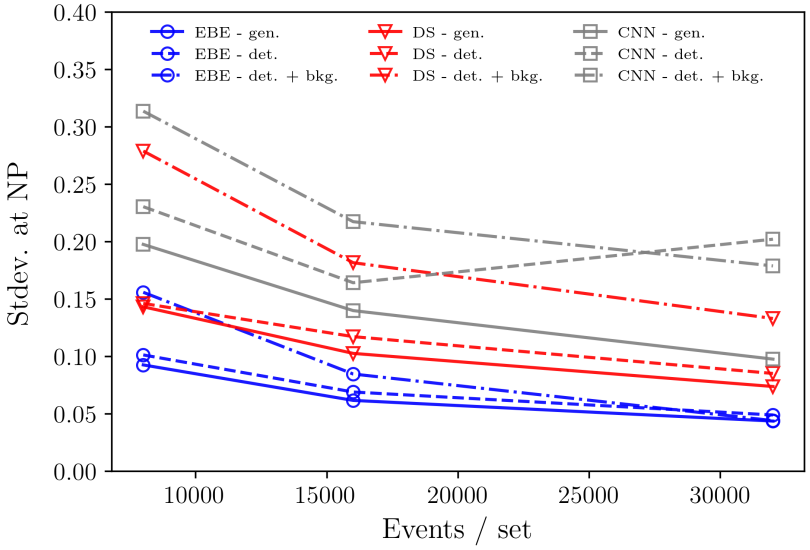

Figure 34: Sensitivity, $1 < q^2 < 19$ GeV^2 test plot.

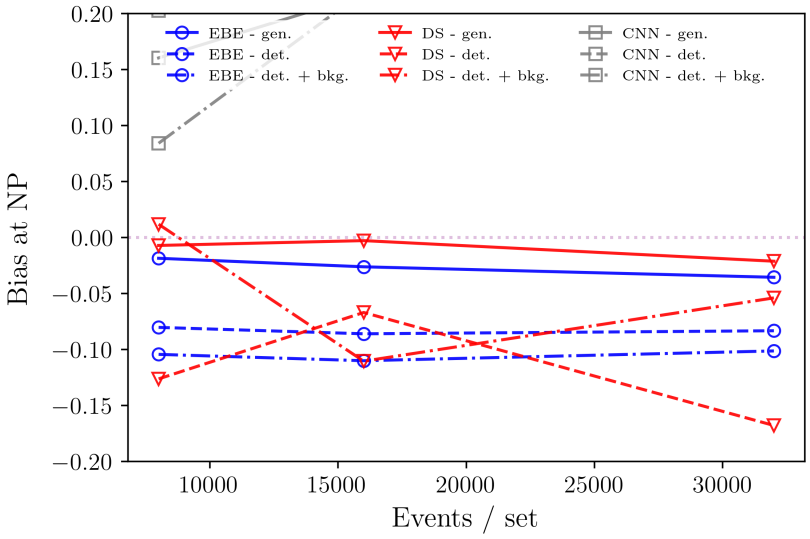

Figure 35: Bias, $1 < q^2 < 19$ GeV^2 test plot.

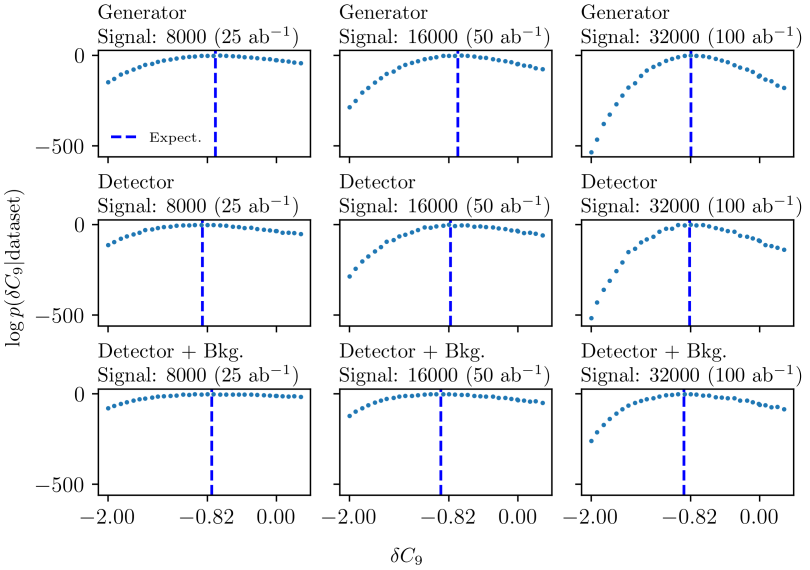

Figure 36: Predicted log probabilities, $0 < q^2 < 20$ GeV^2 validation plot.

5.4 Event-by-event Outputs

In order to compare the event-by-event model to the CNN and Deep Sets models, previously shown linearity, error and sensitivity event-by-event model results were calculated using expectation values of predicted probability distributions. Here we show examples of the predicted probability distributions themselves.

5.4.1 $0 < q^2 < 20$ GeV^2 Validation Results

Figure 36 shows example event-by-event model predictions given validation datasets simulated with $\delta C_9 = -0.82$.

5.4.2 $1 < q^2 < 19$ GeV^2 Test Results

Figure 37 shows example event-by-event model predictions given test datasets simulated with $\delta C_9 = -0.8$.

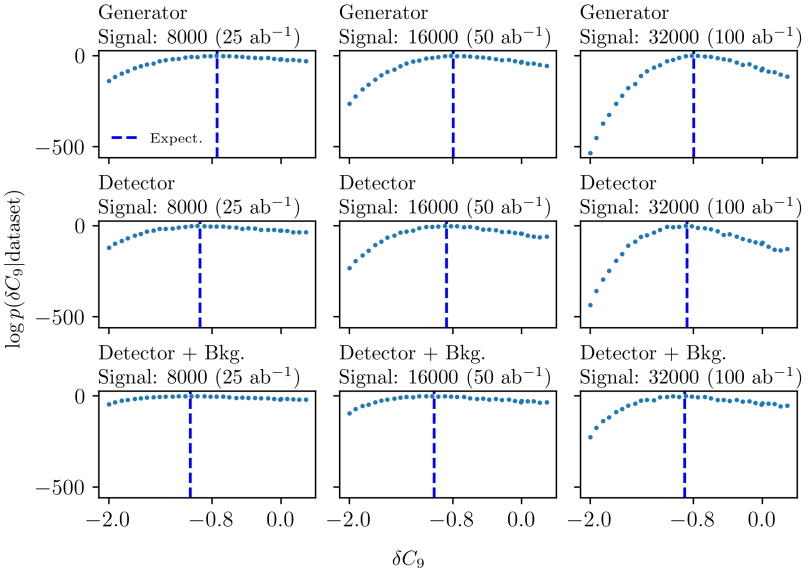

Figure 37: Predicted log probabilities, $1 < q^2 < 19$ GeV^2 test plot.

6 Discussion

6.1 Quantitative Comparison

As seen in section 5, for the $0 < q^2 < 20$ GeV^2 validation dataset, the event-by-event model has the lowest errors and highest sensitivity at all simulation levels and dataset sizes. The Deep Sets model comes in second and the CNN comes in third. These rankings hold for the $1 < q^2 < 19$ GeV^2 test dataset as well.

The distribution shift introduced in the $1 < q^2 < 19$ GeV^2 test dataset affects the CNN the most, causing its predictions to be biased. The Deep Sets and event-by-event models are not affected as much. However, overall results on the $1 < q^2 < 19$ GeV^2 test dataset are worse than results on the $0 < q^2 < 20$ GeV^2 dataset. This might be due to the loss of training data induced by the tighter q^2 cut. One possible reason for the CNN's large bias is that its predictions rely heavily on the q^2 features of a dataset. The distribution shift might primarily consist of a shift in the q^2 distribution, leading to a large bias in the CNN's predictions.

In comparison to fitting methods explored in previous research, the event-by-event method has similar or better sensitivity. A. Sibidanov et al. [16] fit for δC_9 using a 4-dimensional unbinned maximum likelihood fit using the same input variables that we use, and they obtain a prediction distribution with a standard deviation of about 0.11 when predicting $\delta C_9 = -0.87$ given 50 ab^{-1} datasets at the generator level. This is comparable to the sensitivity (see figure 29) of the event-by-event method when evaluated on 50 ab^{-1} (16,000 events)

datasets at a similar $\delta C_9 = -0.82$, also at the generator level.

Using the CNN method, S. Dubey et al. [10] report a prediction standard deviation of 0.044 to 0.051 at $\delta C_9 = -2.0$ at the generator level with approximately 24,000 events per image. The event-by-event method receives a standard deviation of 0.026 under these same circumstances, but with 16,000 events per dataset.

6.2 Qualitative Comparison

Each model also has different qualitative benefits and drawbacks. One important factor is ease of use, which includes ease of training as well as ease of inference. First we discuss ease of training. In our experience, it is easier to train set-based models. We think that this is due to two underlying effects. One, sets of events contain more information than single events; each training example has thousands of times more information than the examples used to train an event-based model, leading to easier fit convergence. Although we can increase the amount of information available to an event-based model by using a large batch size, it is difficult to reach a high concentration of events per label without running out of memory. Another reason set-based models are easier to train is that event-based approaches are sensitive to small biases. In order to obtain a prediction, the same event-based model is applied over all events in a dataset. Any bias is aggregated over many events, leading to a large error. Evidence for this is the careful training required for the event-by-event network.

Even though it is more difficult to train, the event-by-event model seems to require orders of magnitude fewer parameters than the CNN and Deep Sets models. This might be due to how the independence of events (given a value of δC_9) is hard-wired into the model. The set-based models do not include this information. The small size of the event-based model is a positive feature in that it requires less memory to train, requires less space to store, and requires less computation to make predictions.

In terms of inference, event-based models are much easier to use. If training is successful, an event-based network can be used to make predictions on a dataset of any size because it only sees one event at a time. In comparison, set-based networks must be retrained for inference on different dataset sizes because dataset size could be learned by the model during training.

Event-based models are also more interpretable than set-based models; we can visualize the prediction of each event. It is more difficult to see how each event affects a prediction from a set-based model. This interpretability is not only great for debugging, but also important for building a user's trust in a model, which is essential for adoption by other researchers.

6.3 Future Directions

Interesting avenues for future work include exploring how these SBI techniques apply to fitting for δC_9 given $B^0 \to K^{*0} e^+ e^-$ decays as well as simultaneous fits for multiple Wilson Coefficients. The simulator we use can already generate

events given new physics values for multiple Wilson Coefficients, so testing how these approaches generalize should be straightforward.

Another possible future topic could be changing the way the event-by-event model handles backgrounds. Currently, we try to train the model to predict a uniform distribution for background events by randomly labeling background events in the training dataset. Unfortunately, this requires that the model learn to both predict δC_9 and perform background suppression. An alternative approach could be to use separate models for δC_9 prediction and background suppression. One way to do this could be the following. If x_i is the ith event in a dataset and s_i is whether or not that event is a signal event (1 for signal and 0 for background), we see that

$$p(\delta C_9 \,|\, x_i) = p(\delta C_9,\ s_i = 1 \,|\, x_i) + p(\delta C_9,\ s_i = 0 \,|\, x_i) \tag{30}$$

Using the chain rule of probability, we obtain

$$p(\delta C_9,\ s_i = 1 \,|\, x_i) = p(\delta C_9 \,|\, s_i = 1,\ x_i)\ p(s_i = 1 \,|\, x_i) \tag{31}$$

and similarly,

$$p(\delta C_9,\ s_i = 0 \,|\, x_i) = p(\delta C_9 \,|\, s_i = 0,\ x_i)\ p(s_i = 0 \,|\, x_i) \tag{32}$$

Inserting equations 31 and 32 into equation 30 leaves us with

$$\begin{aligned} p(\delta C_9 \,|\, x_i) = \quad & p(\delta C_9 \,|\, s_i = 1,\ x_i)\ p(s_i = 1 \,|\, x_i) \\ & + p(\delta C_9 \,|\, s_i = 0,\ x_i)\ p(s_i = 0 \,|\, x_i) \end{aligned} \tag{33}$$

In equation 33 we can replace $p(\delta C_9 \,|\, s_i = 0,\ x_i)$ with the prior $p(\delta C_9)$ because a background event should not contain any information on δC_9. We can also replace $p(s_i = 0 \,|\, x_i)$ with $1 - p(s_i = 1 \,|\, x_i)$ because s_i can only be 0 or 1. This gives us

$$\begin{aligned} p(\delta C_9 \,|\, x_i) = \quad & p(\delta C_9 \,|\, s_i = 1,\ x_i)\ p(s_i = 1 \,|\, x_i) \\ & + p(\delta C_9)\ (1 - p(s_i = 1 \,|\, x_i)) \end{aligned} \tag{34}$$

After rearranging, we are left with

$$\begin{aligned} p(\delta C_9 \,|\, x_i) = \quad & p(s_i = 1 \,|\, x_i)\ p(\delta C_9 \,|\, s_i = 1,\ x_i) \\ & + (1 - p(s_i = 1 \,|\, x_i))\ p(\delta C_9) \end{aligned} \tag{35}$$

which is just a weighted average between the prediction given that an event is a signal event and the prediction given that an event is a background event (the prior), with each weight being the probability of the respective scenario. In this case, we could approximate $p(\delta C_9 \,|\, s_i = 1\ x_i)$ with the detector level event-by-event model, $p(s_i = 1 \,|\, x_i)$ with the background suppression model described previously, and $p(\delta C_9)$ with the uniform prior we also use previously. The alternative form of $p(\delta C_9 \,|\, x_i)$ given by equation 35 would be substituted

into equation 11. This method could allow us to forgo training a δC_9 prediction model on the detector and background dataset.

Another further future prospect could be obtaining detector level predictions from a model trained on generator level data. One way we could do this is by using the known reweighting properties of a binary classifier trained to distinguish between detector and generator level events. We obtain the reweighting equation as follows. Given that x_i denotes the ith event in a dataset, and d_i denotes whether or not x_i represents a detector ($d_i = 1$) or generator ($d_i = 0$) level event, applying Bayes' rule gives

$$p(\delta C_9 \,|\, x_i,\ d_i = 1) = \frac{p(d_i = 1 \,|\, x_i,\ \delta C_9)}{p(d_i = 1 \,|\, x_i)} p(\delta C_9 \,|\, x_i) \tag{36}$$

Similarly,

$$p(\delta C_9 \,|\, x_i,\ d_i = 0) = \frac{p(d_i = 0 \,|\, x_i,\ \delta C_9)}{p(d_i = 0 \,|\, x_i)} p(\delta C_9 \,|\, x_i) \tag{37}$$

Dividing equation 36 by 37, we get

$$\frac{p(\delta C_9 \,|\, x_i,\ d_i = 1)}{p(\delta C_9 \,|\, x_i,\ d_i = 0)} = \frac{p(d_i = 0 \,|\, x_i)}{p(d_i = 1 \,|\, x_i)} \frac{p(d_i = 1 \,|\, x_i,\ \delta C_9)}{p(d_i = 0 \,|\, x_i,\ \delta C_9)} \tag{38}$$

If we define

$$R_1(x_i) = \frac{p(d_i = 0 \,|\, x_i)}{p(d_i = 1 \,|\, x_i)} \tag{39}$$

and

$$R_2(x_i, \delta C_9) = \frac{p(d_i = 1 \,|\, x_i,\ \delta C_9)}{p(d_i = 0 \,|\, x_i,\ \delta C_9)} \tag{40}$$

rearranging gives us

$$p(\delta C_9 \,|\, x_i,\ d_i = 1) = R_1(x_i)\, R_2(x_i, \delta C_9)\, p(\delta C_9 \,|\, x_i,\ d_i = 0) \tag{41}$$

Assuming that $p(\delta C_9 \,|\, x_i,\ d_i = 1)$ is a binned distribution, we obtain a normalization condition

$$\sum_j p(\delta C_{9j} \,|\, x_i,\ d_i = 1) = 1 \tag{42}$$

from which we can solve for $R_1(x_i)$ to obtain

$$R_1(x_i) = (\sum_j R_2(x_i,\ \delta C_{9j})\, p(\delta C_{9j} \,|\, x_i,\ d_i = 0))^{-1} \tag{43}$$

Defining a reweighting function $R_3(x_i,\ \delta C_9)$, we have

$$p(\delta C_9 \,|\, x_i,\ d_i = 1) = R_3(x_i,\ \delta C_9)\ p(\delta C_9 \,|\, x_i,\ d_i = 0) \tag{44}$$

where

$$R_3(x_i,\ \delta C_9) = R_1(x_i)\ R_2(x_i,\ \delta C_9) \tag{45}$$

$R_1(x_i)$ is given in terms of $R_2(x_i,\, \delta C_9)$ above, and

$$R_2(x_i,\, \delta C_9) = \frac{1 - p(d_i = 0 \,|\, x_i,\, \delta C_9)}{p(d_i = 0 \,|\, x_i,\, \delta C_9)} \tag{46}$$

using the identity $p(d_i = 1 \,|\, x_i,\, \delta C_9) = 1 - p(d_i = 0 \,|\, x_i,\, \delta C_9)$. We can therefore convert generator level predictions into detector level predictions by approximating $p(d_i = 0 \,|\, x_i,\, \delta C_9)$ using a binary classifier.

7 Conclusion

In this work we investigated three different neural simulation-based inference approaches for predicting δC_9 given a dataset of $B^0 \to K^{*0}\mu^+\mu^-$ events. These methods include both set-based and event-based techniques. We find that there are practical benefits and drawbacks to each method. However, we find that event-based methods, in particular the event-by-event neural network, are most promising for measuring δC_9.

8 Resources

8.1 Software

We list the main software used for this project in table 7. Code for this project can be found at this GitHub repository:
`https://github.com/ethanlee20/btokstmumu-ml`.

8.2 Hardware

See table 8 for a list of hardware components used for model training. All training hardware is consumer grade. We also use the KEK computing cluster (KEKCC) and the Belle II distributed computing system (also known as the grid) to run Monte Carlo simulations to generate our datasets.

9 Acknowledgments

I would like to thank Professor Tom Browder for advising me on this project, Professor Sven Vahsen and the University of Hawai'i Belle II members for helpful feedback, Professor Peter Sadowski for suggesting the event-by-event approach, Dr. Alexei Sibidanov for creating the simulator used for this project, and Dr. Shawn Dubey for his work on the CNN method.

Name	Version	Note
Python	3.13.5 / 3.11.13	Programming language
basf2	06-01-08 / 9	Belle II software framework
NumPy	2.1.2	Linear algebra utilities
SciPy	1.15.3	Image generation
pandas	2.3.0	Data management
PyTorch	2.7.1+cu128	Neural network utilities
AutoGluon	1.4.0	Automated machine learning
Matplotlib	3.10.3	Visualization

Table 7: Main software used for this project. Note that we use separate Python versions for training neural networks (3.13.5) using PyTorch and training background suppression models (3.11.13) using AutoGluon. As discussed previously, we also use two different basf2 versions for generating the training (06-01-08) and test (9) signal datasets.

Component	Specification
CPU	AMD Ryzen 7 7800X3D 8-Core (4.20 GHz)
GPU	NVIDIA GeForce RTX 4090 (24 GB)
RAM	64 GB
Storage	1.8 TB

Table 8: Hardware used to train models. Note that this is not a list of requirements.

References

[1] M. Algueró, B. Capdevila, S. Descotes-Genon, J. Matias, and M. Novoa-Brunet. "b $\to$ s $\ell^+\ell^-$ global fits after R_{K_S} and $R_{K^{*+}}$". In: *The European Physical Journal C* 82.4 (Apr. 2022). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-022-10231-1. URL: http://dx.doi.org/10.1140/epjc/s10052-022-10231-1.

[2] W. Altmannshofer, P. Ball, A. Bharucha, A. J. Buras, D. M. Straub, and M. Wick. "Symmetries and asymmetries of $B \to K^*\mu + \mu$ decays in the Standard Model and beyond". In: *Journal of High Energy Physics* 2009.01 (Jan. 2009), pp. 019–019. ISSN: 1029-8479. DOI: 10.1088/1126-6708/2009/01/019. URL: http://dx.doi.org/10.1088/1126-6708/2009/01/019.

[3] W. Altmannshofer and P. Stangl. "New physics in rare B decays after Moriond 2021". In: *The European Physical Journal C* 81.10 (Oct. 2021). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-021-09725-1. URL: http://dx.doi.org/10.1140/epjc/s10052-021-09725-1.

[4] Belle II Collaboration et al. *Measurement of the branching fraction for the decay $B \to K^*(892)\ell^+\ell^-$ at Belle II*. 2022. arXiv: 2206.05946 [hep-ex]. URL: https://arxiv.org/abs/2206.05946.

[5] B. Capdevila. "Assessing lepton-flavour non-universality from B $\to$ K$^*\ell\ell$ angular analyses". In: *Journal of Physics: Conference Series* 873 (July 2017), p. 012039. ISSN: 1742-6596. DOI: 10.1088/1742-6596/873/1/012039. URL: http://dx.doi.org/10.1088/1742-6596/873/1/012039.

[6] M. Chrzaszcz, A. Mauri, N. Serra, R. S. Coutinho, and D. van Dyk. "Prospects for disentangling long- and short-distance effects in the decays B $\to$ K$^*\ell^+\ell^-$". In: *Journal of High Energy Physics* 2019.10 (Oct. 2019). ISSN: 1029-8479. DOI: 10.1007/jhep10(2019)236. URL: http://dx.doi.org/10.1007/JHEP10(2019)236.

[7] M. Ciuchini, M. Fedele, E. Franco, A. Paul, L. Silvestrini, and M. Valli. *Charming Penguins and Lepton Universality Violation in $b \to s\ell^+\ell^-$ decays*. 2022. arXiv: 2110.10126 [hep-ph]. URL: https://arxiv.org/abs/2110.10126.

[8] K. Cranmer, J. Brehmer, and G. Louppe. "The frontier of simulation-based inference". In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062. DOI: 10.1073/pnas.1912789117. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1912789117. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1912789117.

[9] M. Deistler, J. Boelts, P. Steinbach, G. Moss, T. Moreau, M. Gloeckler, P. L. C. Rodrigues, J. Linhart, J. K. Lappalainen, B. K. Miller, P. J. Gonçalves, J.-M. Lueckmann, C. Schröder, and J. H. Macke. *Simulation-Based Inference: A Practical Guide*. 2025. arXiv: 2508.12939 [stat.ML]. URL: https://arxiv.org/abs/2508.12939.

[10] S. Dubey, T. E. Browder, S. Kohani, R. Mandal, A. Sibidanov, and R. Sinha. *Training 3D ResNets to Extract BSM Physics Parameters from Simulated Data*. 2025. arXiv: 2311.13060 [hep-ex]. URL: https://arxiv.org/abs/2311.13060.

[11] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data". In: *arXiv preprint arXiv:2003.06505* (2020).

[12] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[13] T. Hurth, F. Mahmoudi, D. Martínez Santos, and S. Neshatpour. "More indications for lepton nonuniversality in $b \to s\ell^+\ell^-$". In: *Physics Letters B* 824 (Jan. 2022), p. 136838. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2021.136838. URL: http://dx.doi.org/10.1016/j.physletb.2021.136838.

[14] T. Kuhr, C. Pulvermacher, M. Ritter, T. Hauth, and N. Braun. "The Belle II Core Software". In: *Computing and Software for Big Science* 3.1 (2018), p. 1. DOI: 10.1007/s41781-018-0017-9. URL: https://doi.org/10.1007/s41781-018-0017-9.

[15] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. *Normalizing Flows for Probabilistic Modeling and Inference*. 2021. arXiv: 1912.02762 [stat.ML]. URL: https://arxiv.org/abs/1912.02762.

[16] A. Sibidanov, T. E. Browder, S. Dubey, S. Kohani, R. Mandal, S. Sandilya, R. Sinha, and S. E. Vahsen. "A new Monte Carlo generator for BSM physics in $B \to K^*\ell^+\ell^-$ decays with an application to lepton non-universality in angular distributions". In: *JHEP* 08 (2024), p. 151. DOI: 10.1007/JHEP08(2024)151. arXiv: 2203.06827 [hep-ph].

[17] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. *Deep Sets*. 2018. arXiv: 1703.06114 [cs.LG]. URL: https://arxiv.org/abs/1703.06114.