Selective background Monte Carlo simulation at Belle II

James Kahn[12], **Thomas Kuhr**[1], **and Martin Ritter**[1]

[1] Ludwig-Maximilians University Munich, Faculty of Physics, Excellence Cluster Universe, Boltzmannstr. 2, 85748 Garching, Germany

E-mail: `james.kahn@kit.edu`

Abstract. The Belle II experiment, beginning data taking with the full detector in early 2019, is expected to produce a volume of data fifty times that of its predecessor. This dramatic increase in data comes the opportunity for studies of rare previously inaccessible processes. The investigation of such rare processes in a high data-volume environment requires a correspondingly high volume of Monte Carlo simulations to prepare analyses and gain a deep understanding of the contributing physics processes to each individual study. This presents a significant challenge in terms of computing resource requirements and calls for more intelligent methods of simulation, in particular background processes with very high rejection rates. This work presents a method of predicting in the early stages of the simulation process the likelihood of relevancy of an individual event to the target study using convolutional neural networks. The results show a robust training that is integrated natively into the existing Belle II analysis software framework.

1. Introduction

The Belle II experiment will push the precision frontier by obtaining an integrated luminosity of $50\,\mathrm{ab}^{-1}$ worth of data, or roughly fifty times that of the Belle experiment. Performing measurements on this volume of 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 $10 \times 0.711\,\mathrm{ab}^{-1}$. The same approach at Belle II would then require simulations of an integrated luminosity of $500\,\mathrm{ab}^{-1}$ which is infeasible given the current simulation time and computing power 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 that particular study. Therefore more intelligent methods of simulation are required to produce large volumes of relevant simulated data.

[2] Present address: Karlsruhe Institute of Technology, Institut für Experimentelle Teilchenphysik, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany

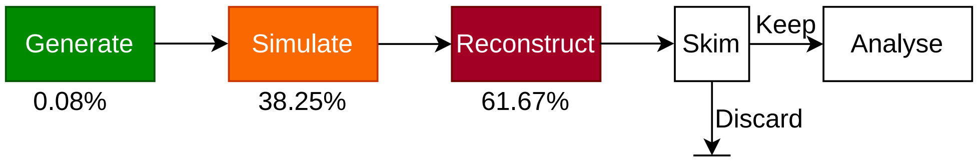

Figure 1: Stages in the Monte Carlo event simulation process. The figures shown under the first three stages indicate the percent of simulation time required by the given stage before the skim.

2. Monte Carlo simulation at Belle II

Figure 1 shows an outline of the current Monte Carlo simulation procedure for Belle II, where the stages are as follows: Generate (green) involves the generation of the initial collision event using the EvtGen package [1] in conjunction with Pythia [2]; Simulate (orange) utilises the Geant4 toolkit [3] to perform the simulation of the generated collision products with the Belle II detector. The output format of this stage is identical to that of the real experiment output. Reconstruct (red) performs the reconstruction of detector information into particle candidates; Skim involves the selection of events which are of easily identifiable interest to particular working-groups within the Belle II collaboration; and Analyse represents the specific physics analyses performed on the data (real or simulated). The relative execution times of each stage in the simulation procedure are also shown as a percentage of the total time.

The Belle II Analysis Software Framework (basf2) [4] is the core framework used to manage each stage in the data flow shown in figure 1. The framework is written predominantly in C++ but supports user-developed modules written in Python. This allows the integration of current machine learning libraries, most of which are developed almost exclusively for high-level languages such as Python, directly into the simulation processing flow. The machine learning framework used in this study is Tensorflow [5], with the Keras libraries utilised to simplify the construction of the neural networks. All trainings described in this study were performed using a single NVIDIA GTX 1080Ti GPU.

3. Data set

A key part of missing energy studies at Belle II, those containing neutrinos in the final state, is the ability to fully reconstruct the initial $\Upsilon(4S)$ produced within the detector while also having knowledge of the initial state. The Full Event Interpretation (FEI) [6] software is the tool used within the Belle II software to perform this full reconstruction of the $\Upsilon(4S)$ by automating the reconstruction of the B meson not involved in the decay being searched for by the particular analysis. The data set used in this study contains simulated events which have had the FEI algorithm applied. Loose selections were applied to the resulting events as the skim stage of processing. The resulting combined event-level retention rate of the FEI and selections are shown in table 1, where hadronic represents the reconstruction of B meson decays involving no neutrinos in the final state[3]. The two categories highlighted in bold are those used for training in this study.

The data set contains 8.5(4) million simulated events for the hadronically tagged $B^0\bar{B}^0(B^+B^-)$ sample, with approximately half of the events being those which survive the skim stage, and the other those discarded by it. Events are flagged by whether they survive the skim selections, and these flags are used as training labels for the neural networks. Ten percent of events are reserved for validation during training to check against overfitting.

We characterise the data set in two ways: decay strings, string representations of the

[3] Neutrinos may be present, however, in the decay of the B meson not reconstructed by the FEI.

Table 1: Full Event Interpretation skim retention rates showing the remaining fraction of simulated events after data cleaning selections (skim) have been applied. The figures in bold are those used in this study.

Channel	Hadronic B^+	Hadronic B^0
$\mathbf{B^0\bar{B}^0}$	5.62%	**4.25%**
$\mathbf{B^+B^-}$	**8.35%**	3.82%

Table 2: Features of each MC generated particle used as input to network training and their definition. For decay string inputs only the PDG code is used to identify each particle.

Feature	Definition
PDG code	Identifier of particle type and charge.
Mother PDG code	Particle parent PDG code.
Mass	Particle mass in GeV/c^2.
Charge	Electric charge of the particle.
Energy	Particle energy in GeV.
Momentum	Three momentum of the particle in GeV/c.
Production time	Production time in ns relative to $\Upsilon(4S)$ production.
Production vertex	Coordinates of particle production vertex.

simulated decay; and MCParticles, the array of physical properties associated with each individual simulated particle. Table 2 summarises the MCParitcles properties used as input in this study. All continuous properties are normalised to the range -1 to 1, and discrete properties are one-hot encoded (Charge) or tokenised for embedding within the network (PDG and mother PDG codes).

4. Training

The procedure for training the neural networks is as follows: we first explore a range of network architectures trained on the decay string and MCParticle inputs individually, we then select the optimally performing architectures from each and concatenate their outputs as input to a final series of fully-connected layers. The final combined network is trained again from scratch and re-optimised. A schematic of the final network structure is shown in figure 2. The structure is logically comprised of three individual sub-networks: two individual sub-networks processing the MCParticles and decay string inputs individually, and a final single sub-network to produce the classification output.

For each sub-network, combinations of fully connected and convolutional layers were investigated. ResNet [7], ResNeXt [8], and 1×1 convolutions [9] variants were also explored for the MCParticles sub-network. For the decay strings, recurrent LSTM [10] layers and so-called wide convolutions [11] were investigated.

The loss function used in training was binary cross-entropy, utilising the Adam [12] optimiser with and without AMSGrad [13]. LeakyRELU was used as the activation function for intermediate layers, with a sigmoid activation for the final output layer. To prevent overfitting batch normalisation and dropout were implemented as necessary. During training, the number of layers, layer sizes (kernel sizes, number of filters, etc.), learning rate, and batch size were varied

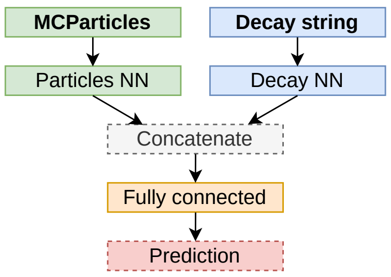

Figure 2: Outline of the overall neural network architecture. The two inputs are first analysed individually by their respective neural network (NN), their outputs concatenated, and passed as input to a final series of fully connected layers which output the resulting prediction.

Table 3: Results of the various architectures tested in this study. Each value indicates the maximal performance achieved by the given architecture after hyper-parameter optimisation.

Model	Validation loss (B^+/B^0)
Fully connected	0.291/0.346
Convolutional	0.276/0.299
ResNet	0.269/0.290
ResNeXt	0.278/0.299
1×1 CNN	0.269/0.288
Convolutional-LSTM	0.265/0.311

to maximise training data classification performance. Training ended when the classification performance on the validation data showed no improvement in three consecutive epochs.

Table 3 shows the results of training, expressed as validation loss for comparison. The fully connected model serves as a baseline, with fully connected layers comprising both MCParticle and decay string sub-networks. The convolutional-LSTM model corresponds to a vanilla convolutional network for MCParticle inputs, with kernel size greater than one and no skip-connections, and recurrent LSTM layers for decay strings. The remainder of the models use wide convolutions for decay string inputs, and the convolutional variant shown in the model name for MCParticle inputs. The convolutional (MCParitlces) and LSTM (decay strings) combination showed the highest precision on validation data. The presence of recurrent layers, however, greatly slows the relative inference and training times due to the lack of parallelisation afforded by fully connected or convolutional layers. Therefore we identify the fully convolutional architectures utilising the wide convolutions on decay string inputs as providing the optimal compromise between speed and accuracy.

5. Evaluation

In order to evaluate the performance of the trained network during simulation an additional 10^5 events of the charged sample were simulated. The trained neural network was integrated into the simulation procedure, as shown in figure 3. The goal here is two-fold: firstly to check that the trainings do indeed provide a speedup in the simulation of backgrounds, and secondly to attempt to identify any biases introduced by the network removing select events.

To apply the trained network we inserted a custom basf2 module into the simulation data flow. The module receives the output of the generate stage, performs the necessary preprocessing prescribed in section 3, applies the trained network, and returns a predicted likelihood that the event will pass the skim stage. For the purposes of evaluation we retained all events, regardless of their predicted pass likelihood.

To measure the speedup provided by the vetoing of events by the neural network, we calculated the total simulation time required per event. To do so we summed the time contributions of each event case: true positive, false positive, true negative, and false negative. True and false positive cases, those the network predicts to pass, both contribute the processing times of the entire simulation process. True and false negatives only contribute the processing times of the generation and neural network inference. False negatives, however, require additional simulations to compensate for those which would have passed the skim stage being

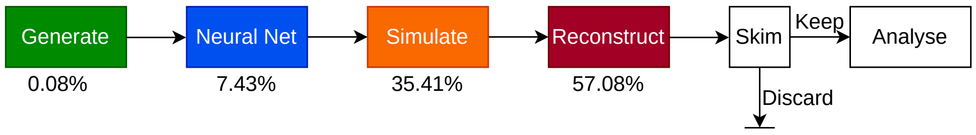

Figure 3: The updated Monte Carlo event simulation process from figure 1 with the trained neural network inserted. The numbers indicate the relative percent of processing time required by each stage.

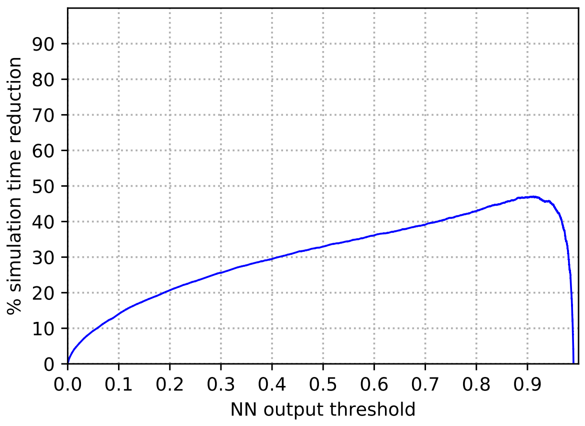

Figure 4: Simulation time speedup per event reaching the analyse stage of processing in figure 3 for the charged (B^+) FEI sample.

incorrectly discarded. Figure 4 shows the resulting speedup factor for the evaluation sample as a function of the threshold applied to the neural network output probability. At a high threshold of around 0.9 we observed a speedup of just under 50%.

In order to check against false-negative event-rejection introducing biases into the final simulated data set, we examined a selection of kinematic variables from the sample. We fixed the neural network threshold at 0.85 to allow a high statistics comparison. Figure 5 shows an example of two kinematic variables used: the beam-constrained mass of the reconstructed B meson, defined as $M_{\mathrm{bc}} = \sqrt{E_{\mathrm{beam}}^2 - p_B^2}$, where E_{beam} is half of the initial beam energy and p_B is the reconstructed momentum of the B mesons; and the total number of charged tracks in the event. The kinematic distributions obtained from no threshold applied (blue) were scaled by the true-positive rate of the network and compared to the observed distribution when the threshold was applied (orange) via a binomial test. Below each distribution is the resulting p-value, with the orange line indicating a value of 0.05 for comparison. While M_{bc} shows good agreement, the number of charged tracks indicates a significant deviation from the unbiased distribution in the six and eleven track events. This motivates a bias mitigation procedure to be investigated in detail, either by means of re-weighting of events or as an adversarial penalty applied to counteract bias during training (as discussed in [14]).

6. Conclusion
In this work we have explored a method of selective Monte Carlo simulation as a means of reducing computational requirements. With the use of neural networks we have demonstrated that it is possible to predict early in the simulation process how likely a simulated event is to be useful in a given analysis. We developed the procedure for extracting information from Monte

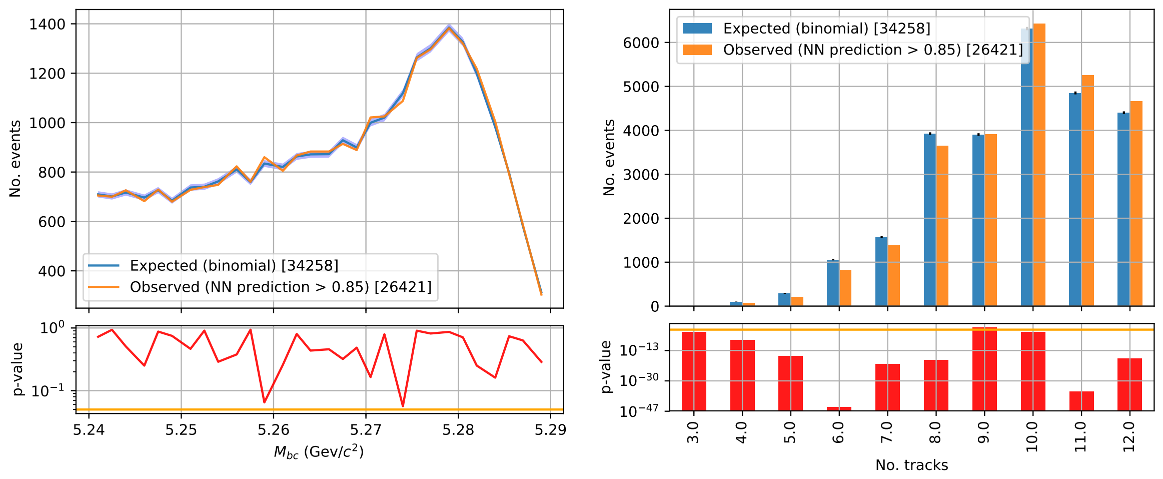

Figure 5: Examples of comparison of expected (blue) and observed (orange) kinematics of simulated events in the charged FEI evaluation sample for beam-constrained mass (left) and number of charged tracks (right). The figures under each comparison show the corresponding p-value of the binomial test, with the orange line marking 0.05 for visual comparison.

Carlo data in Belle II and performing the necessary preprocessing steps for input to a neural network. We then showed, through the simulation of an independent sample, that the trained network was able to significantly reduce the simulation time requirements. Using the output of this independent sample we outlined a method of quantifying kinematic biases introduced by the network. The biases found were non-negligible and motivate further investigation into their mitigation. 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 controlled then this represents a powerful technique that will enable significant increases in simulation volumes for Belle II.

References

[1] Lange D J 2001 The EvtGen particle decay simulation package *Nucl. Instrum. Meth. A* **462** 152
[2] Sjöstrand T *et al.* 2015 An introduction to PYTHIA 8.2 *Comput. Phys. Commun.* **191** 159
[3] Agostinelli S *et al.* 2003 GEANT4: A simulation toolkit *Nucl. Instrum. Meth. A* **506** 250
[4] Kuhr T *et al.* 2019 The Belle II core software *Comput. Softw. Big Sci. no. 1* **3** 1
[5] Abadi M *et al.* 2016 TensorFlow: Large-scale machine learning on heterogeneous distributed systems *arXiv e-prints*
[6] Keck T *et al.* 2019 The Full Event Interpretation: An exclusive tagging algorithm for the Belle II experiment *Comput. Softw. Big Sci. 1* **3** 6
[7] Kaiming H, Xiangyu Z, Shaoqing R and Jian S 2015 Deep residual learning for image recognition *arXiv e-prints*
[8] Saining X, Ross G, Piotr D, Zhuowen T and Kaiming H 2016 Aggregated residual transformations for deep Neural Networks *arXiv e-prints*
[9] Szegedy C *et al.* 2014 Going deeper with convolutions *arXiv e-prints*
[10] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput. no. 8* **9** 1735
[11] Zhang Y and Wallace B 2015 A sensitivity analysis of (and practitioners' guide to) Convolutional Neural Networks for sentence slassification *arXiv e-prints*
[12] Kingma D P and Ba J 2014 Adam: A method for stochastic optimization *arXiv e-prints*
[13] Reddi S J, Kale S and Kumar S 2019 On the convergence of adam and beyond *CoRR*
[14] Shimmin C, Sadowski P, Baldi P, Weik E, Whiteson D, Goul E and Søgaard A 2017 Decorrelated jet substructure tagging using Adversarial Neural Networks *Phys. Rev. D no. 7* **96** 074034