Selective background Monte Carlo simulation at Belle II

James Kahn[1,*], *Emilio* Dorigatti[2,**], *Kilian* Lieret[2,3,***], *Andreas* Lindner[2,3,****], and *Thomas* Kuhr[2,3,†]

[1]Karlsruher Institut für Technologie
[2]Ludwig-Maximilians-Universität München
[3]Excellence Cluster Origins

Abstract. The large volume of data expected to be produced by the Belle II experiment presents the opportunity for studies of rare, previously inaccessible processes. Investigating such rare processes in a high data volume environment necessitates 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 resulting challenge, in terms of computing resource requirements, calls for more intelligent methods of simulation, in particular for processes with very high background 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 graph 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 has begun data taking in 2019 [1]. Over its lifetime it is expected to record an integrated luminosity of $50\,\mathrm{ab}^{-1}$, roughly 50 times that of its predecessor, the Belle experiment [2]. Analysing this volume of data requires a correspondingly large volume of simulated data, in particular if the focus is on studies of rare processes (branching ratio $< \mathcal{O}(10^{-6})$) [3], as is the case at Belle II. This simulated data is used to understand the relevant regions of phase space for a particular study. More importantly, however, it is used to understand which regions can be excluded to remove background physics processes, those not of interest to that study. The approach at the Belle experiment was to simulate ten times the volume of data. This becomes computationally infeasible at Belle II without significant improvements to the simulation time. Further, for any given physics working group within Belle II, a large fraction of the simulated data is discarded trivially before reaching individual analyses, typically of the order $\mathcal{O}(0.1$–$10\%)$. In this work we present an alternative approach to simulating large data volumes. The key shift is from simulating everything and discarding uninteresting events later to simulating only those events likely to be of interest to a particular physics working group. This work builds on that of [4], with the

[*]e-mail: james.kahn@kit.edu
[**]e-mail: emilio.dorigatti@stat.uni-muenchen.de
[***]e-mail: kilian.lieret@lmu.de
[****]e-mail: and.lindner@physik.uni-muenchen.de
[†]e-mail: thomas.kuhr@lmu.de

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

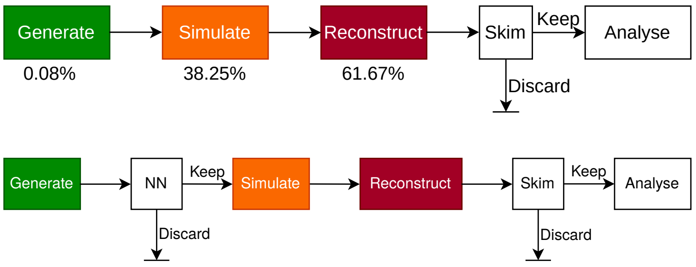


Figure 1: Monte Carlo event simulation workflow at Belle II with (bottom) and without (top) a neural network inserted to predict the likelihood of simulated events being kept by the skim stage. Relative processing times for each stage are shown under the top diagram.

main contribution being the introduction of graph neural networks as a more natural means of processing particle decays.

2 Monte Carlo simulation at Belle II

The simulation of collision events at Belle II is performed in three main stages, followed by a data reduction stage to prepare datasets for specific physics working groups within the collaboration. Figure 1 (top) shows the stages with their relative time contributions.These are: *Generate* (green) which is the generation of the initial collision event and subsequent decay using the EvtGen [5] and Pythia [6] packages. *Simulate* (orange) performs the simulation of the generated decay interaction with the Belle II detector using the Geant4 toolkit [7], the output of which models that of the real experiment readout. *Reconstruct* (red) involves the reconstruction of detector information into particle candidates. *Skim* is the filtering of events which are easily identified as not relevant to particular physics working groups within the Belle II collaboration. The volume of events remaining after the skim is typically $\mathcal{O}$(0.1–10%). *Analyse* represents the specific physics analysis performed on the data (real or simulated).

The entire simulation and analysis workflow is performed using the Belle II Analysis Software Framework (basf2)[8]. The framework is written in C++, with the user API being Python. This allows user-developed modules written in Python to be integrated into the processing workflow.

As the machine learning libraries used in this study are all Python, we are able to directly insert in a simple way our neural network solution into the simulation processing flow by making use of the existing basf2 API. Figure 1 (bottom) shows the updated workflow with the neural network (NN) from this study inserted after the generate stage of simulation. The machine learning framework used in this study is Tensorflow [9], with the Keras API utilised to simplify construction of the NNs.

3 Graph Neural networks

A graph is a mathematical object composed of nodes connected by edges. Because of its generality, graph theory has been applied to a variety of problems in a multitude of fields, and a number of NN architectures have been devised to handle this specific data modality[10]. Most architectures are based on a message-passing framework where the features associated with every node are combined with the features of its neighbors, thus propagating information one hop at a time. A general way to formalize this message passing operation is as a graph convolution in the spectral domain, using concepts from graph signal processing[11]. Let $H^{(l)}$ denote the activations in layer l, then one possible formulation is [12]:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right) = \sigma\left(LH^{(l)}W^{(l)}\right) \tag{1}$$

with $\sigma(\cdot)$ an activation function, $\tilde{A} = I_N + A$ the sum of the identity matrix and the adjacency matrix A, $A_{ij} = 1$ if there is an edge between nodes i and j, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix counting the number of neighbors of every node plus one, and $W^{(l)}$ a learnable weights matrix. The Laplacian L is a central object in graph signal processing, as its eigenvalues reveal several properties of the respective graph.

Knowing that every particle decay represents a multifurcating tree, we take advantage of this by extending the variant of eq. (1) prescribed in [13]. The subsequent node update logic is:

$$N^{(l+1)} = \sigma\left(W_p^{(l)}N_p^{(l)} + W^{(l)}N^{(l)} + W_d^{(l)}\sum_{\text{daughters}} N_d^{(l)}\right), \tag{2}$$

where N is the node being updated at layer l, and $N_p^{(l)}$ and $N_d^{(l)}$ are the node's parent and daughter nodes respectively. Corresponding trainable weights are assigned to each of the three: $W_p^{(l)}$, $W^{(l)}$, and $W_d^{(l)}$. In practice this is implemented by splitting the Laplacian in eq. (1) into a lower triangular, diagonal, and upper triangular components to isolate the parent, current node, and daughters, respectively.

For the scope of this project, each decay is interpreted as a graph in which nodes correspond to particles and edges model the parent-daughter particle relationship. Graph neural networks make it possible to take into account the structure of the decay and the relationship between particles, as well as properties of the particles themselves, increasing predictive power compared to models that only use particles' features.

4 Dataset

Our strategy is applied to two skims of Belle II MC simulations. One skim is produced for the study of time dependent CP violation (TDCPV skim, ~0.2% retention rate), the other contains hadronic $B^{\pm}$ decays (*FEI skim*, ~5% retention rate). In both cases our dataset consists of $\sim$300,000 generated events containing a binary classification label signalling whether at least one reconstructed candidate is retained by the skim.

The TDCPV skim selects candidates for $\sim$20 exclusive B decays featuring $b \to qqs$, $q = u, d, s$ (e.g. $B^0 \to K_S^0\pi\gamma$) and $b \to ccs$ transitions. Besides detector acceptance cuts and event level continuum suppression cuts, only $M_{\text{bc}} \in (5.2, 5.29)$, GeV and $|\Delta E| < 0.5$ GeV are required. Here the beam-constrained mass M_{bc} and the energy difference ΔE are two complementary variables comparing the B kinematics to the energy of the beam [14].

The FEI skim selects candidates for hadronic $B^{\pm}$ decays using the Belle II Full Event Interpretation (FEI), a reconstruction algorithm relying heavily on machine learning techniques [15]. Detector acceptance and event level continuum suppression cuts similar to the TDCPV skim are applied before the FEI returns a list of BB candidates, of which we require a signal probability (as reported by the FEI) of larger than 0.1%, $M_{\text{bc}} > 5.24$ GeV and $|\Delta E| < 0.2$ GeV.

The following generator level variables (available for each particle in the event) are used as inputs to the network:

- PDG code (integer corresponding to particle type and charge, tokenized).
- Mother PDG code (PDG code of the ancestor particle): Used together with the PDG code to build up the adjacency matrix as described in section 3.

- Energy of the particle (log scaled).
- Particle coordinates x, y, z at time of creation. All coordinates sharply peak around zero but also contain a significant amount of larger values, making a meaningful normalization challenging. Therefore, each coordinate $c = x, y, z$ is split into three features: the double log of the coordinate ($\log\log c$), as well as two boolean features (represented as zero or unit value) corresponding to $|c| < 0.01\,\mathrm{m}$ and $|c| \geq 0.01\,\mathrm{m}$. Particles with $|c| > 10\,\mathrm{m}$ are removed, as they leave no trace in the detector (and therefore add no information that could be picked up by the reconstruction/skimming algorithms).
- Particle momentum projections p_x, p_y, p_z (normalized to vanishing mean and unit variance).
- Production time t (time between initial e^+e^- collision and birth of the particle) as $t^{1/10}$.

In addition, the following event level variables are written out for bias checks (section 6.1):

- Event level reduced Fox-Wolfram moment R_2 (ratio of second to zeroth Fox-Wolfram moment as introduced in [16]), a variable describing the distribution of momentum and energy in the event (typically used for suppression of continuum events).
- Number of charged tracks in the event.
- Δ decay time: The generator level difference of the decay times of both B mesons in the event.

As all input features have to be supplied as fixed length arrays to the training, a decision had to be made for the maximum number of particle features to be included. While up to 100 particles are produced in certain events, cropping the arrays to 40 particles (ordered by ascending production time) was found to be sufficient to reduce the data size without sacrificing performance. For events with less than 40 particles, the arrays are zero-padded.

5 Model study

We constructed individual classification NNs for both of the FEI and TDCPV datasets. The initial layers are extended graph convolutions described in section 3 which produce representations of individual nodes. These are pooled to produce a fixed sized representation of the entire graph, followed by fully connected layers to output the final classification prediction. To find the optimal network architecture, we investigated variations of layer components, with individual layer hyperparameters varied over reasonable ranges.

Trainings were performed on 250×10^3 of the events from each of the FEI and TDCPV datasets, and ten percent of this sample is used for validation during training to monitor for overfitting to the training data. The remaining roughly 50×10^3 events in each dataset were set aside for performance testing after training and model optimisation were complete.

For all trainings we use the Adam optimiser with an initial learning rate of 10×10^{-3}, which is reduced by a factor of 2 when training loss no longer decreases after several epochs. We utilised batch normalisation [17] to expedite training, and dropout [18] on the final dense layers to help mitigate overfitting. We used a binary cross-entropy loss function, with the loss label terms weighted by their corresponding class weights to compensate for any imbalances in the training dataset. Training was stopped when validation loss no longer improved after two learning rate reductions. The final model saved for inference was that with the best performance on validation data.

Figure 2 shows the components of the final, best-performing architectures. Two graph convolution layers were found to produce optimal performance. As the PDG codes described in section 4 are discrete identifiers with no relational ordering, we projected them first into

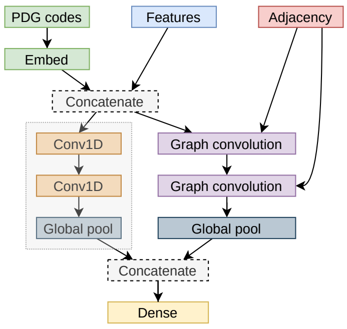
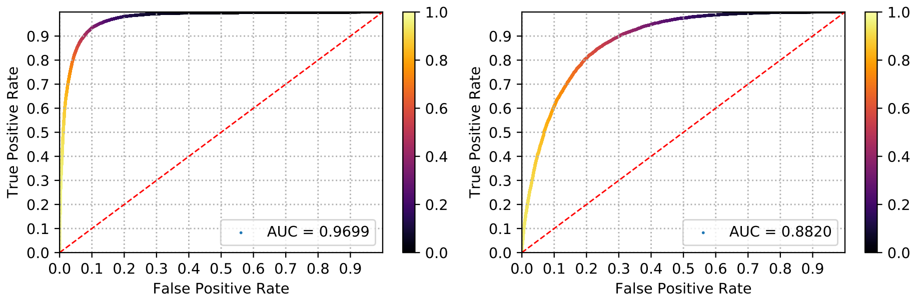


Figure 2: Neural network components. The convolutional layers (conv1D) in the grey box were used only on the hadronic B^+ meson reconstruction (FEI) data.



(a) TDCPV (b) FEI

Figure 3: Receiver operating characteristic curves for the TDCPV and FEI trained networks on independent test data. The colours indicate the threshold applied to the network outputs at each point on the curve, and the annotation indicates the area under the curve (AUC).

a learned, continuous, sixteen-dimensional representation [19]. The output of this was concatenated with the remaining particle features as input to the following graph convolution layers. As prescribed by [13], we utilised a readout function to produce a whole graph embedding from the individual node representations output by the GIN layers. To prevent overrepresentation of larger graphs containing more nodes, we opted for global average pooling. An additional, parallel series of one-dimensional convolution layers were found to boost the performance of training on the FEI dataset. The output of this was concatenated with the pooled graph convolution outputs. We used a final series of three fully connected (dense) layers with 256, 128, and 32 nodes. All layers used a Leaky-ReLU activation except the final output, which used a sigmoid activation function.

6 Evaluation

The ultimate goal of this work is integration into the Monte Carlo simulation workflow described in section 2, the key requirement being an overall improvement in the speed of simulating events that reach physics analyses. Simultaneously, we want to identify any bias introduced in the output. The evaluation of the trained network we performed was twofold:

a network accuracy check on the test dataset of 50×10^3 events and an inference time check on a newly simulated set of events. The accuracy check was used to determine the true and false positive rates of the network. The time check determined the relative time contribution of the module to the overall simulation process.

Basf2 allows custom Python modules to be inserted into the processing flow. We used this to construct an inference module that applies a trained network to simulated events directly after the generation stage. The module repeats the steps taken during data production (extracting the decay graph), preprocessing of attributes, application of the network, and finally returns the network prediction. The returned prediction can then be used in basf2's control flow to discard events below a specified threshold.

For the accuracy tests, inference was performed on the test dataset and the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR) recorded for a range of thresholds.

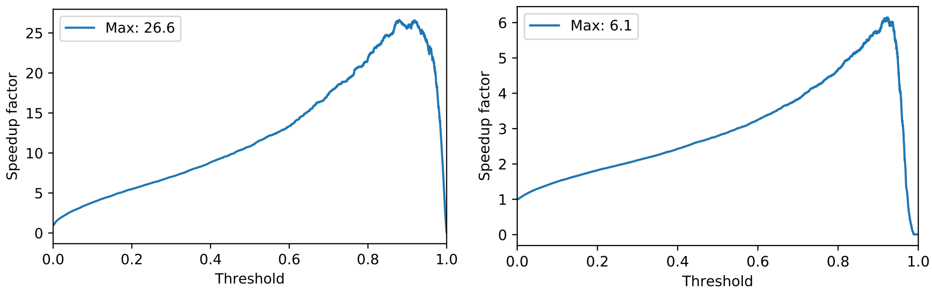
For the time check, the inference was performed on a per-event basis, using an identical NumPy implementation of preprocessing to that described in section 4. Multiple simulations were performed of batches of 10,000 events. The relative average execution time was recorded for input to the speedup calculation (recall that we are only interested in relative simulation improvements).

We are concerned only with speeding up the simulation of events used for analysis, which are those that pass the skim stage. Therefore, we present the result as the ratio of the time taken to simulate one skim-passing event without the NN applied to the time taken with the NN applied. There are two main cases to consider when calculating this: events which do not pass the specified threshold and events which do pass the NN threshold, regardless of whether they pass the skim. The former only contributes the times of the generate stage (t_{gen}) and NN inference (t_{NN}) to the total time calculation. The latter additionally contributes the time of the simulate and reconstruct stages (t_{SR}) to t_{gen} and t_{NN} in the calculation. As the time contribution of the skim depends highly on the specific skim process and at which point during the skim an event is rejected, we neglect it from our calculations. We expect that including the time contribution of the skim into the speedup calculation would increase the final speedup factor as by design we are reducing the number of skim rejected events reaching the skim stage.

Figure 4 shows the speedup factors for both TDCPV and FEI. At a high (~ 0.9) TDCPV threshold the speedup factor is greater than 25. For FEI it is just over 6. Note that in practice, multiple trained networks (or one large multi-skim network) may be used to provide classification predictions for multiple skims, saving the need for repeated simulation of events which pass more than one skim. As t_{NN} is on average two orders of magnitude less than t_{SR} we don't expect this to significantly affect the speedup.

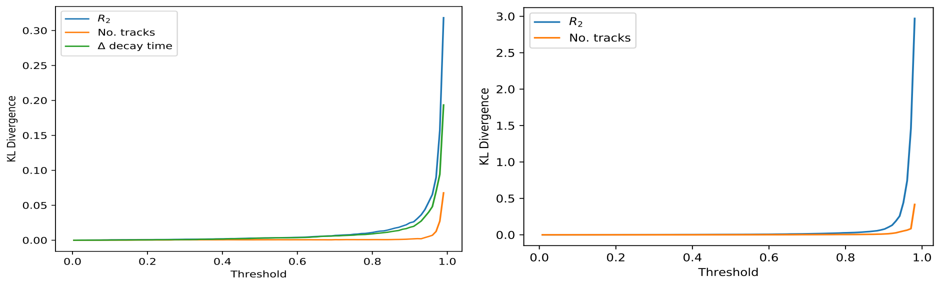
6.1 Bias checks

Biases may be introduced into the simulated dataset via different skim-passing event types (e.g. decay channels) being discarded via the network threshold selection differently. To check for this we inspect changes in selected event-level kinematic observables. We calculate the Kullback-Leibler divergence between the true distribution (when no NN is applied), and the resulting distribution from different thresholds when the network is applied. Figure 5 shows the resulting distributions. The sudden spike at high thresholds is due to the sharp drop in the number of events remaining after the threshold selection. This causes the statistical fluctuation in each observable to dominate the divergence calculation. Overcoming this, which is the original grounds for this project, demands the use of the trained NN in the sim-



(a) TDCPV (b) FEI

Figure 4: Simulation speedup factors at varying thresholds for the simulation of a single event passing the corresponding skim selection.



(a) TDCPV (b) FEI

Figure 5: Kullback-Leibler divergence between expected and resulting event-level kinematic distributions at varying thresholds.

ulation procedure. This involves the simulation of new set of simulated events specifically in the high threshold region, which is beyond the scope of this project.

7 Conclusion

Graph neural networks exploit the structural information contained in particle decays as well as the properties of the particles themselves. In this work we have presented a method of utilising graph convolutional networks to reduce the computational requirements of Monte Carlo simulations. Extending previous work on the topic, we have demonstrated their implementation in the context of the Belle II experiment. Our experiments showed a maximum speedup of over 6 and 25 times for fully reconstructed hadronic $B^{\pm}$ and time dependent CP violation datasets respectively. We also presented a potential means of identifying introduced biases and identified its shortcomings, motivating further work.

8 Acknowledgements

We thank the Max Planck Institute of Molecular Cell Biology and Genetics Dresden for organizing the deep learning hackathon during which the graph neural networks in this work were first applied for event selection at generator level.

Our work was also supported by the BMBF funded ErUM-Data pilot project Innovative Digital Technologies for Research on Universe and Matter.

References

[1] F. Abudinén et al. (Belle-II collaboration) (2019), `1910.05365`
[2] T. Abe et al. (Belle-II collaboration) (2010), `1011.0352`
[3] W. Altmannshofer et al. (Belle-II collaboration) (2018), `1808.10567`
[4] J.M.S. Kahn, Ph.D. thesis, Munich U. (2019), `https://edoc.ub.uni-muenchen.de/24013/`
[5] D.J. Lange, Nucl. Instrum. Meth. **A462**, 152 (2001)
[6] T. Sjöstrand, S. Ask, J.R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C.O. Rasmussen, P.Z. Skands, Comput. Phys. Commun. **191**, 159 (2015), `1410.3012`
[7] S. Agostinelli et al. (GEANT4 collaboration), Nucl. Instrum. Meth. **A506**, 250 (2003)
[8] T. Kuhr, C. Pulvermacher, M. Ritter, T. Hauth, N. Braun (Belle-II Framework Software Group), Comput. Softw. Big Sci. **3**, 1 (2019), `1809.04299`
[9] M. Abadi et al. (2016), `1603.04467`
[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. Yu, IEEE Transactions on Neural Networks and Learning Systems **PP**, 1 (2020)
[11] M. Defferrard, X. Bresson, P. Vandergheynst, *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*, in *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2016), NIPS'16, p. 3844–3852, ISBN 9781510838819
[12] T.N. Kipf, M. Welling, CoRR **abs/1609.02907** (2016), `1609.02907`
[13] K. Xu, W. Hu, J. Leskovec, S. Jegelka, CoRR **abs/1810.00826** (2018), `1810.00826`
[14] A.J. Bevan et al. (BaBar collaboration, Belle collaboration), Eur. Phys. J. **C74**, 3026 (2014), `1406.6311`
[15] T. Keck et al., Comput. Softw. Big Sci. **3**, 6 (2019), `1807.08680`
[16] G.C. Fox, S. Wolfram, Phys. Rev. Lett. **41**, 1581 (1978)
[17] S. Ioffe, C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (JMLR.org, 2015), Vol. **37** of *ICML'15*, p. 448–456
[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Journal of Machine Learning Research **15**, 1929 (2014)
[19] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, *Distributed Representations of Words and Phrases and their Compositionality*, in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*, edited by C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (2013), pp. 3111–3119, `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality`