EPJ Web of Conferences **245**, 02010 (2020) https://doi.org/10.1051/epjconf/202024502010
CHEP 2019

Generation of Belle II Pixel Detector Background Data with a GAN

Matej Srebre[1], *Pascal* Schmolz[1], *Hosein* Hashemi[1], *Martin* Ritter[1], and *Thomas* Kuhr[1*]
for the Belle II Software Group

[1]Ludwig-Maximians-Universität München

Abstract. To match the statistical precision due to the large dataset that Belle II is expected to collect, simulations that accurately describe real data are required. The effect of beam background must be considered and can be taken into account with overlaying random trigger data, but its large size is a technical challenge. This problem can be mitigated by generating beam background data with generative adversarial networks. A proof of principle is shown for the background data recorded by the pixel vertex detector.

1 Introduction

Belle II [1] is a B-factory experiment at the SuperKEKB accelerator [2] in Tsukuba, Japan. Its design luminosity is 8×10^{35} cm^{-2}s^{-1}, 40 times higher than what has been achieved at e^+e^- colliders so far. The accelerator is mainly operated at a center of mass energy of $\sqrt{s} \approx 10.6$ GeV to produce $Y(4S)$ resonances that decay almost exclusively to entangled pairs of either charged or neutral B mesons. Other processes of interest are continuum events ($e^+e^- \to q\bar{q}$ with $q = u, d, s, c$) and tau pair production. The Belle II physics program [3] is broad and covers in particular indirect searches for new physics in rare decays and CP asymmetries as well as studies of (exotic) hadron states.

To significantly increase the sensitivity of searches and measurements it is planned to collect a 50 times larger dataset than its predecessor Belle. To profit from the much larger recorded dataset systematic uncertainties of simulated datasets must be well under control. This requires an accurate incorporation of beam background effects in the detector simulation.

2 Beam background simulation

Various background processes lead to signals in the detector that do not originate from the physics processes of interest. These are Touschek scattering of electrons or positrons inside a bunch, beam-gas scattering, synchrotron radiation, (radiative) Bhabha events ($e^+e^- \to e^+e^-(\gamma)$), and two-photon events ($e^+e^- \to e^+e^-e^+e^-$). The last has an event rate of about one order of magnitude higher than the physics events and contributes in particular to the beam background in the innermost detector. A further source of background are noisy bunches from injections.

*e-mail: Thomas.Kuhr@lmu.de

One method for assessing the beam background effects is a simulation of the particles produced in the background processes and their interaction with the detector using Geant4 [4]. This has the advantage that the subsequent simulation of the digitization correctly takes into account cases where energy deposits from background and signal particles contribute to the same channel. On the other hand the comparison of background simulations using this so-called mixing technique with measurements show sizable discrepancies.

An alternative approach is to take data collected with a random trigger and combine it with the simulated signal event. This technique, called background overlay, works on digit level and thus needs a detector-specific treatment of cases where background and signal contribute to the same channel. Nevertheless it is expected to yield a more realistic simulation of beam background effects. In particular, it provides correct levels of beam background in a run-dependent simulation.

A drawback of the background overlay method is that resources are required for the readout, storage, and distribution of the random trigger events. A single event in the format to be used for the background overlay has a size of about 200 kB. This is roughly a factor 20 more than the size of events in the format used for analyses. Furthermore the beam background files have to be distributed to all sites where run-dependent Monte-Carlo samples are produced. This multiplies the storage demands by the number of sites. In particular smaller sites do not have the storage resources with the required size and bandwidth to host full beam background datasets.

3 Pixel Vertex Detector

The data of the innermost detector, the Pixel Vertex Detector (PXD) [5], account for about half of the size of beam background events. The PXD is a silicon detector based on the DEPFET technology [6] with two layers as shown in Fig. 1. The layer at a radius of 14 mm consists of 8 layers and the layer at a radius of 22 mm of 12 layers. Each ladder has two sensors with 250×768 pixels each. This results in approximately 8 million channels.

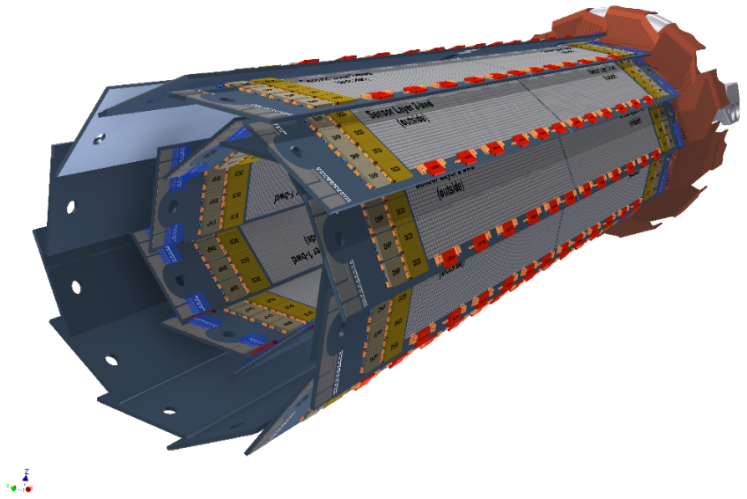

Figure 1: The Belle II Pixel Vertex Detector (PXD).

With a readout time of 20 μs the PXD integrates over about 5000 bunch crossings. Thus the majority of pixels with a charge above threshold stem from beam background. A possible way to reduce the PXD data rate is to read out only regions of interests defined by tracks reconstructed in the tracking devices at larger radii and extrapolated to the PXD. By default Belle II did not use this option so far, but it will likely be used when the luminosity and the PXD occupancy increase.

4 Background Generation

To mitigate the problem of resource demands for the beam background files we propose to generate the PXD beam background data on demand instead of storing it in the beam background files. Inspired by the successes of generative adversarial networks (GANs) [7] to produce realistic images we employ this type of machine learning to produce PXD sensor data.

The data of each of the 40 sensors corresponds to an 8-bit grey-scale image with 250×768 pixels. As training data set we use 900.000 simulated events. Ignoring the position dependence in a first iteration this gives 36 million sensor data samples.

Our GAN model is based on existing models that were used successfully to generate natural images of dimension 32×32 or 64×64 [8, 9]. To account for the larger 250×768 dimension of the PXD sensor data we added further blocks for dimensionality reduction or enlargement to the discriminator and the generator, respectively. The used model is shown in Tab. 1. The generator input are 96 normally distributed random numbers. As observed often with GANs we encountered convergence difficulties. We succeeded to train a model that produces reasonable generator output with a Wasserstein GAN [10] with gradient penalty [11]. The training was done in 180,000 steps where each step consists of five parameter updates of the discriminator and one parameter update of the generator. 40 randomly sampled images were used for each parameter update with the Adam optimizer [12]. The learning rate was set to 10^{-4} for the first 90,000 steps and then linearly decayed to reach 0 for the last step. The PyTorch framework [13] was used to implement and train the model.

$\mathbf{x} \in \mathbb{R}^{256\times768}$
Conv(32, 5, 2, 2), LeakyReLU
Conv(64, 5, 2, 2), LeakyReLU
Conv(128, 5, 2, 2), LeakyReLU
Conv(256, 5, 2, 2), LeakyReLUU
Conv(512, 5, 2, 2), LeakyReLU
Global Max Pooling
Dense $\rightarrow$ 1

(a) Discriminator $D(\mathbf{x})$

$\mathbf{z} \in \mathbb{R}^{96} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$
Dense $\rightarrow 512 \times 8 \times 24$
Conv(512, 5, 1, 2), BN, ReLU, Upsample
Conv(256, 5, 1, 2), BN, ReLU, Upsample
Conv(128, 5, 1, 2), BN, ReLU, Upsample
Conv(64, 5, 1, 2), BN, ReLU, Upsample
Conv(32, 5, 1, 2), BN, ReLU, Upsample
Conv(1, 5, 1, 2), Tanh

(b) Generator $G(\mathbf{z})$.

Table 1: Architecture of the discriminator and generator models of the GAN for the generation of PXD beam background data. Conv denotes convolutional layers and the values in parenthesis specify the numbers features and the sizes of kernel, stride, and padding. BN refers to BatchNorm layers.

The training was done on a NVIDIA Tesla V100, required 10 GB of GPU memory, and took on average 3 seconds per step. The generator model has 20 million parameters corresponding to a model size of 80 MB. The generation of a set of 40 PXD sensor images takes

17 seconds with one threads on an Intel Xeon E5-2660 CPU and 0.2 seconds on a NVIDIA Geforce GTX 1080 Ti. For comparison, the simulation of a $Y(4S)$ event in the Belle II detector takes about one second and is dominated by the Geant4 simulation.

5 Evaluation

Fig. 2 shows a comparison of generated sensor images with examples of input images. The distributions and shapes of input and generated clusters look similar.

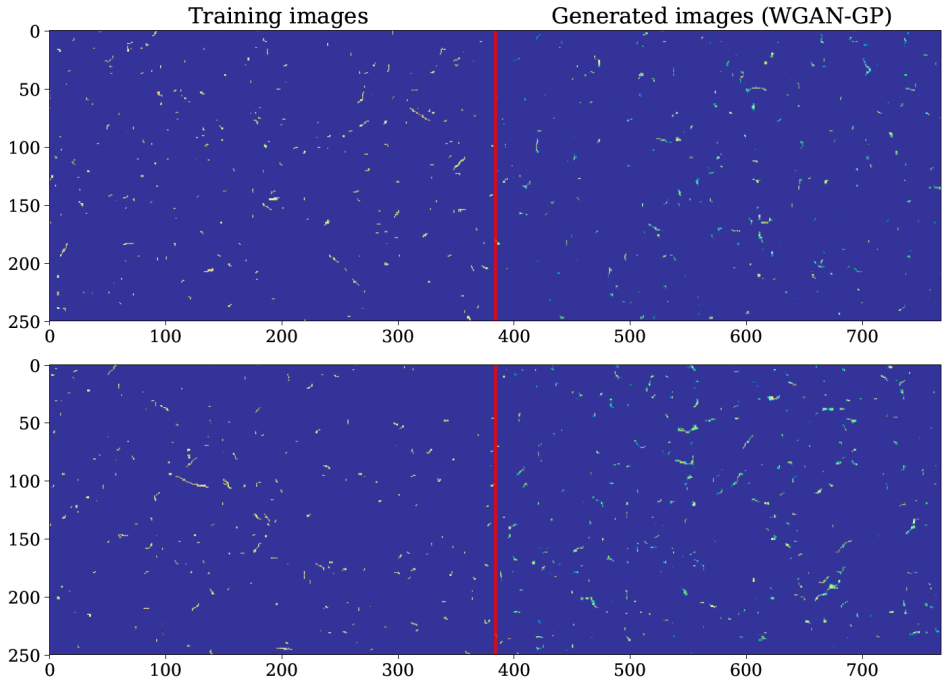

Figure 2: Comparison of examples of input sensor images (left) and generator output images (right).

To quantify the agreement of input and generated data we study the impact of the PXD background on the track reconstruction. We simulate $Y(4S)$ signal events and consider three background overlay scenarios:

(a) No background in the PXD

(b) Fully simulated PXD background as in the input data

(c) PXD background generated with the GAN.

The main effect of background in the PXD is that wrong hits are attached to tracks. This leads to a degradation of the impact parameter resolution.

In Fig. 3 the pulls of the impact parameter in the transverse plane, d_0, and in beam direction, z_0, are compared for the three scenarios. If background is added in the PXD the distributions get broader in both cases, (b) and (c), for low momentum tracks as expected.

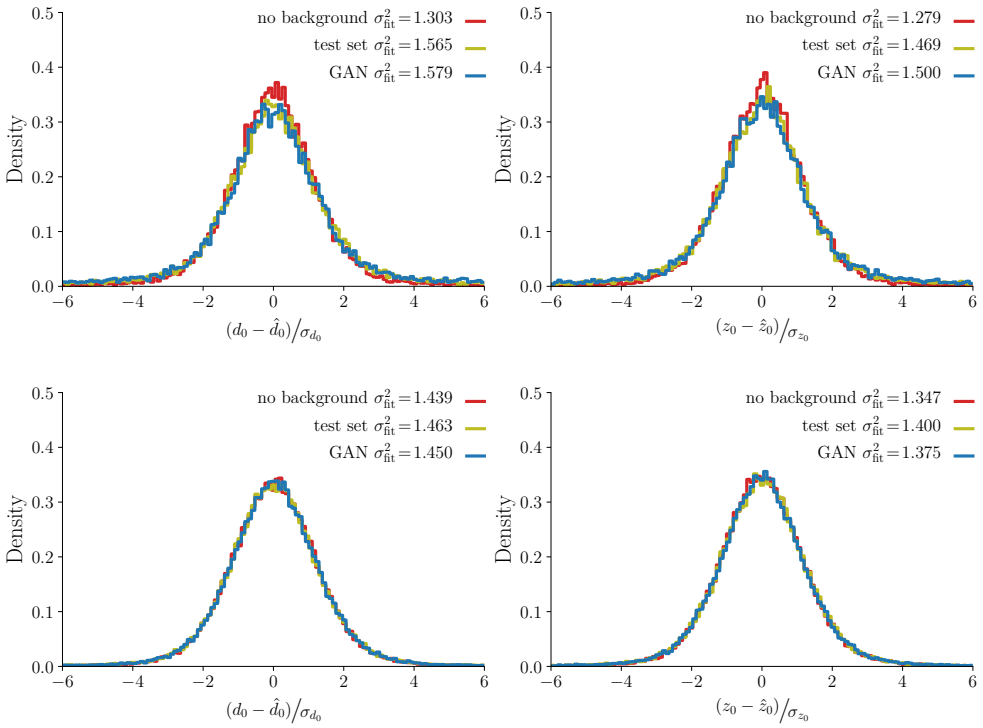

Figure 3: Comparison of pulls of impact parameter in the transverse plane (left) and in beam direction (right) for the transverse momentum ranges $p_T < 0.2$ GeV (top) and $0.2 < p_T < 1$ GeV for the cases of (a) no background in the PXD, (b) fully simulated PXD background as in the input data (test set), and (c) PXD background generated with the GAN. The values in the top right are the widths of a Gaussian fitted to the pull distributions.

Similar increases of the pull widths are observed for the other track parameters. So the background images produced with the GAN show the desired effect. Higher momentum tracks suffer less from multiple scattering and thus their extrapolation to the PXD is more precise so that the addition of background at the considered level does not significantly increase the chance of assigning a wrong PXD hit.

6 Summary and outlook

A realistic simulation requires that beam background effects are taken into account. This can be achieved by overlaying random trigger data, but the large size of background events makes this impractical. We have shown a proof of principle that generative adversarial networks can be used to generate PXD background data. This would allow to reduce the storage demands for background files by a factor two at the cost of additional CPU resources needed for the generation. The remaining half of the storage is needed for the background data of the other detectors and may be reduced by applying the same approach.

While simulated data was used as input in this study, real random trigger data can also be used to accurately model spacial and temporal variations. Besides a track level evaluation cluster properties could be studied as a further measure to quantify the quality of the generated images. A restriction to regions of interest could simplify the training of models. The

background generation presented here ignores correlations among detectors. As the integration time of the PXD is typically more than an order of magnitude longer than that of the other detectors this seems justified, but should be studied. The correlation is in particular to be considered if the approach is extended to other detectors. Another aspect to be investigated is the variability of the generated images. Finally the PXD background generation should be integrated in the Belle II software [14] and the tradeoff between storage and CPU resources assessed. While GANs are already used in particle physics for the simulation of the detector response to signal particles, the presented approach of background data generation may be applied at other experiments as well.

Acknowledgements

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

References

[1] T. Abe (Belle II Collaboration) (2010), 1011.0352
[2] K. Akai, K. Furukawa, H. Koiso (SuperKEKB), Nucl. Instrum. Meth. **A907**, 188 (2018), 1809.01958
[3] W. Altmannshofer et al. (Belle-II) (2018), 1808.10567
[4] S. Agostinelli et al. (GEANT4), Nucl.Instrum.Meth. **A506**, 250 (2003)
[5] F. Mueller (DEPFET), JINST **9**, C10007 (2014)
[6] J. Kemmer, G. Lutz, Nucl. Instrum. Meth. **A253**, 365 (1987)
[7] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *Generative adversarial networks* (2014), 1406.2661
[8] A. Radford, L. Metz, S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks* (2015), 1511.06434
[9] A. Odena, V. Dumoulin, C. Olah, Distill (2016)
[10] M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein gan* (2017), 1701.07875
[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, *Improved training of wasserstein gans* (2017), 1704.00028
[12] D.P. Kingma, J. Ba, *Adam: A method for stochastic optimization* (2014), 1412.6980
[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., *Pytorch: An imperative style, high-performance deep learning library* (2019), 1912.01703
[14] T. Kuhr, C. Pulvermacher, M. Ritter, T. Hauth, N. Braun (Belle-II Framework Software Group), Comput. Softw. Big Sci. **3**, 1 (2019), 1809.04299