PXD background simulation using GANs at Belle II

M Srebre, T Kuhr, M Ritter, P Schmolz

Ludwig-Maximilians-Universität München
Exzellenzcluster “Origin and Structure of the Universe”

October 1, 2019

Introduction

The Belle II experiment

- $e^+ e^-$ at SuperKEKB collider, KEK (Japan)
- Belle II detector: general-purpose detector composed of dedicated sub-detector systems
- high luminosity experiment for precision measurements
- requires good understanding of backgrounds
  → **this talk will address one of them: PXD background**

Pixel Vertex Detector (PXD)

Configuration

- innermost sub-detector, cylindrically arranged surrounding the IP
- composed of two layers with 16 and 24 sensors → 40 modules
- radii: 14mm and 22mm respectively
- fine-grained layout of 250×768 pixels

Purpose:

- sensitive to charged particles propagating through
- reconstruct trajectories (**tracking**) and decay vertices (**vertexing**)

Belle2 experiment and PXD

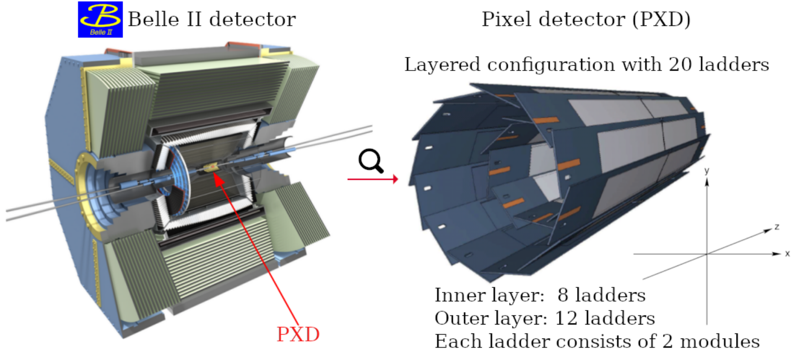


Figure: The Belle II detector on the right, zoomed in on the PXD on the right.

Main idea

Principles of background simulation

- need background for each signal event
- basically two ways: simulate it, or measure it

Approach 1: Measure background

- complete PXD output is extremely large, due to
  1. fine-grained pixels
  2. long readout time for complete PXD image: many bunch crossings are recorded

Approach 2: Simulate background

- very difficult
- overlay of many events
- many different types of physics background processes, hard to consider all of them

Approach 3: our proposal

- measure real background
- use this sample to train a GAN that generates new PXD background
- "on-the-fly", no need to store this

Simulation of PXD background

How we simulate events

- simulate event
  - signal decay with Monte Carlo
  - add background processes, either from **data** or **simulation**
- simulate detector response → tracks and clusters
- reconstruct the event

Current setup

- PXD background is simulated with Monte Carlo methods, stored as hitmaps (PXDDigits)
- background for 100 000 events, files are about 18GB, ~ 200 kB/evt
- almost half of the complete background storage volume is for PXD

Problems

- large files on grid, need to be imported for every event simulation → costs storage and bandwidth, is slow
- limited size of background samples → using same background multiple times
- simulation and measurement do not match (ratio: up to 2 orders of magnitude)
  background processes not well understood → no realistic background simulation available

Generating background

Proposal

Explore Generative Adversarial Network (GAN) to generate PXD background hitmaps (**images**)

GAN

- two neural networks, competing against each other
  - generator: creates new samples from random input vector
  - discriminator: predicts whether its input is **fake** (from generator) or **real** (from data set)
- generator tries to fool the discriminator into classifying generated images as real images
- generator reweights according to discriminator's decision
- aim: both networks improve their skills until discriminator cannot distinguish real from fakes
  $\rightarrow$ generated images "as good as" real images

Generating background

Proof-of-concept study

This work just shows ability to generate PXD-background-like images from a given training sample

- for now: use MC simulated PXD background samples as training data set
- later: use measured PXD outputs as training set
  will be a realistic replica of the background

Desired concept

Have experiment run-time dependent generated PXD background samples

- measure with random-trigger: PXD output shows all (physics + non-physics) background
  → real data training set
- may change over experiment time, but no need to really "understand" what happens
- record a small sample of PXD background every now and then → generate individual, statistically independent PXD background for every single simulated event
- fast, low-costs of resources, on-the-fly generation
- generator embedded in simulation software

Generating background

Training data set

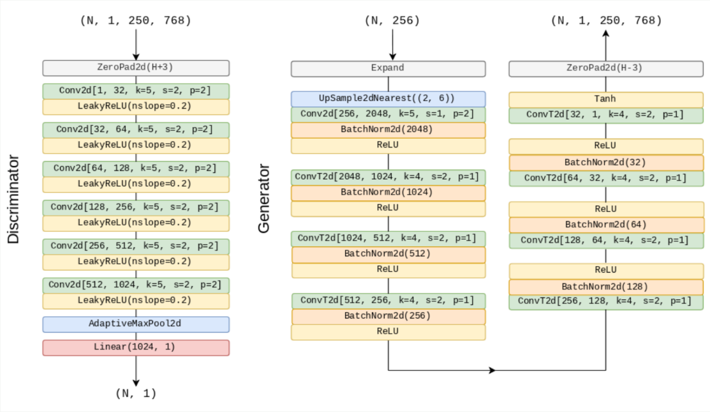
- 400 000 PXD sensor images: 250×768 pixels
- 8-bit readout: convert to 2D gray-scale
- no spatial dependency of modules, i.e. no difference made for L1 and L2 outputs

Conducted studies

Explored several state-of-the-art neural network architectures for natural image generation

1. **Deep Convolutional GAN** (DCGAN)
   extended low-resolution DCGAN examples to accept high-resolution images
   → GAN framework often experienced convergence difficulties
2. **Wasserstein GAN (WGAN)**
   WGAN provides new loss function + some empirical guidelines (weight-clipping)
   → did not meet expecations as well
3. **Improved Training of Wasserstein GAN (WGAN-GP)**
   final implementaion: WGAN with gradient penalty (enforcing the 1-Lipschitz constraint to discriminator)
   gradient penalty replaces weight-clipping and produces higher-quality images

WGAN-GP architecture



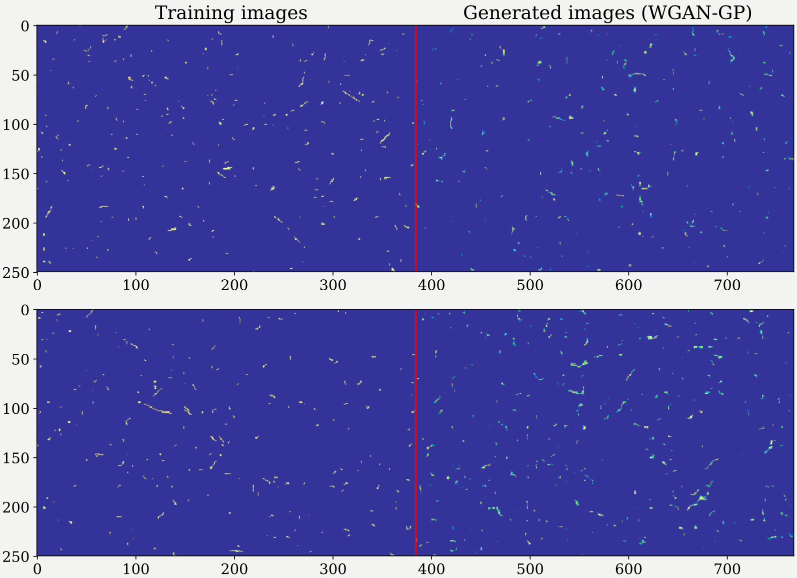
Discriminator

- depth of network: 6 layers
- activation function: LeakyReLU
- kernel: 5×5
- stride: 2
- number of filters: 32

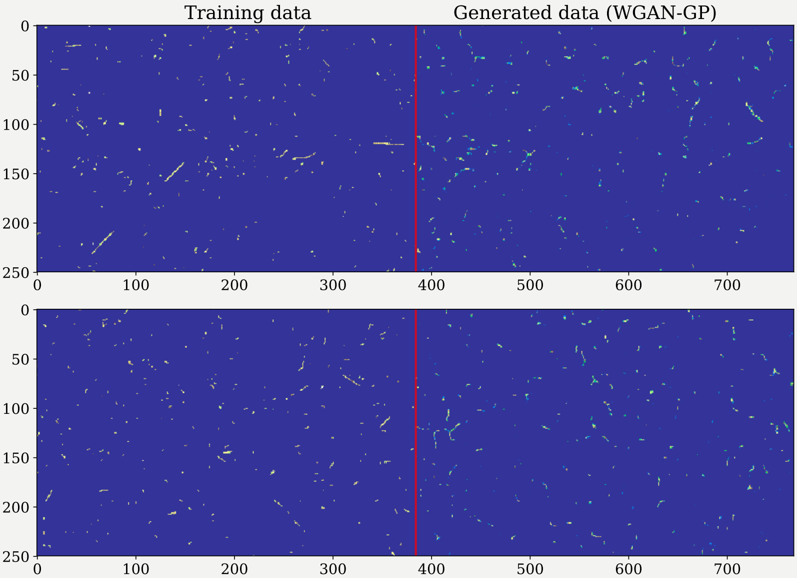
Generator

- depth of network: 8 layers
- activation functions: ReLU + Tanh
- kernel: 4×4
- in: 256-dim normally distributed vector
- out: single module output

Results


Training images
Generated images (WGAN-GP)
0
50
100
150
200
250
0
100
200
300
400
500
600
700
0
50
100
150
200
250
0
100
200
300
400
500
600
700

Results



Training data
Generated data (WGAN-GP)
0
50
100
150
200
250
0
100
200
300
400
500
600
700

Validation of generated PXD background images

Problem

- How to evaluate quality of generated PXD background images?
  → general problem in ML, missing a metric to say how good generated images are
- usual way of visually inspecting images (does this cat look like a cat?) does not apply here

Solution

Perform track reconstruction on:

1. signal decay + no PXD bkg
2. signal decay + MC simulated PXD bkg
3. signal decay + generated PXD bkg

and, as a metric, compare tracking parameters:

- 5 helix parameters of a trajectory: d0, z0, phi0, omega, tan lambda
- track reconstruction efficiency

Validation: d0

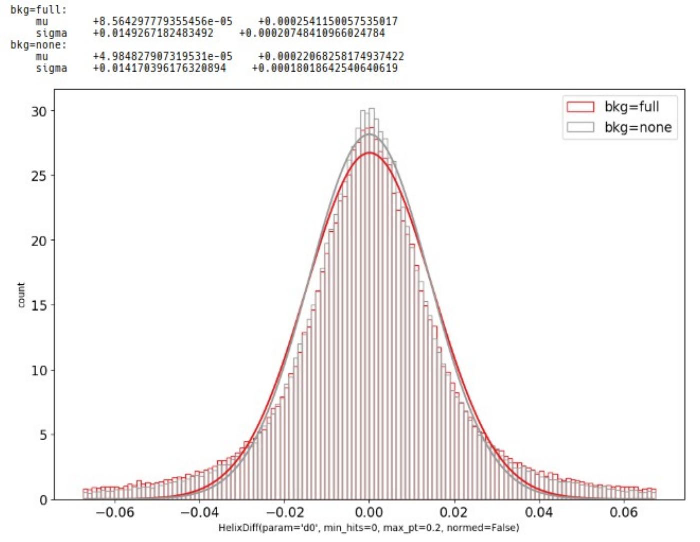


Figure: Preliminary validation plot, comparing tracking parameter d0 with and without PXD background.

Summary

MC simulation

- impractical for PXD background
- simulation is slow, files are large, only limited amount of samples
- and not reliable/realistic: ratio MC/data $\sim$ 2 orders of magnitude

PXD bkg generation

- concept-of-proof study to generate PXD background module output (images)
- dynamical training sets via measured samples

Outlook

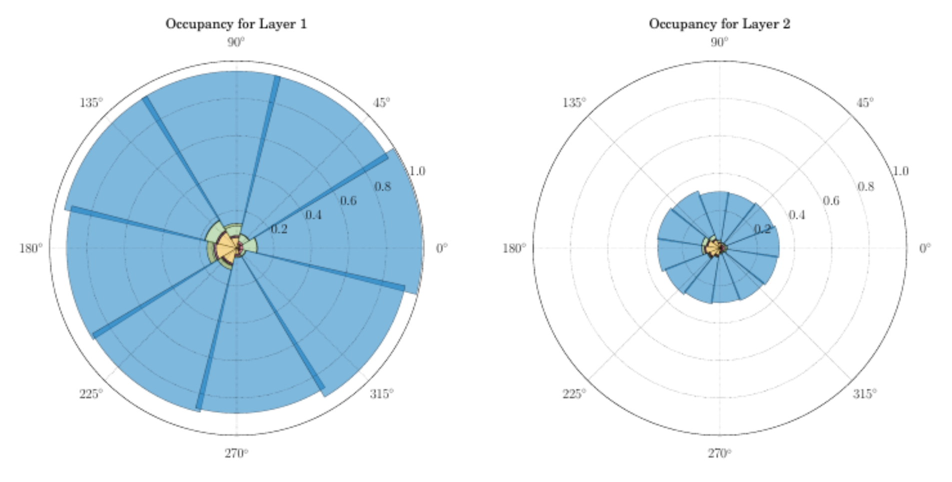
- embed generator in software for event simulation
- desirable: automate process
  1. sample of PXD background measurement
  2. convert to 2D gray-scale images
  3. train GAN
  4. export weights to embedded generator

  $\rightarrow$ no storage of large PXD background files needed
- maybe: two separate networks for layer1 and layer2
- maybe: GAN for ROI (small frame around particle hit)

BACKUP

PXD occupancy

[noframenumbering]



PXD

[noframenumbering]