A hybrid deep learning approach to vertexing

Rui Fang\textsuperscript{1}  Henry Schreiner\textsuperscript{1, 2}  Mike Sokoloff\textsuperscript{1}  Constantin Weisser\textsuperscript{3}  Mike Williams\textsuperscript{3}

April 3, 2019

\textsuperscript{1}The University of Cincinnati
\textsuperscript{2}Princeton University
\textsuperscript{3}Massachusetts Institute of Technology

Supported by:

CtD/WIT 2019
Fang, Schreiner, Sokoloff, Weisser, Williams

A hybrid deep learning approach to vertexing

April 3, 2019
The changes

- 30 MHz software trigger
- 7.6 PVs per event (Poisson distribution)
- Roughly 5.5 visible PVs per event

The problem

- Much higher pileup
- Very little time to do the tracking
- Current algorithms too slow

We need to rethink our algorithms from the ground up...
Vertices and tracks

**Introduction**

**Vertices**

- Events contain \(\approx 7\) Primary Vertices (\(\approx 5\) visible PVs)
  - A PV should contain 5+ long tracks
- Multiple Secondary Vertices (SVs) per event as well
  - A SV should contain 2+ tracks

**Adapt to machine learning?**

- Sparse 3D data (41M pixels) \(\rightarrow\) rich 1D data
- 1D convolutional neural nets
- Highly parallelizable, GPU friendly
- Opportunities to visualize learning process
A hybrid ML approach

Machine learning features (so far)

- Prototracking converts sparse 3D dataset to feature-rich 1D dataset
- Easy and effective visualization due to 1D nature
- Even simple networks can provide interesting results

Introduction
Kernel generation

Design

Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown

- Make a 3D grid of voxels (2D shown)
  - Note: only $z$ will be fully calculated and stored

- Tracking (full or partial)
  - Fill in each voxel center with Gaussian PDF
  - PDF for each (proto)track is combined
  - Fill $z$ "histogram" with maximum KDE value in $xy$ axis (along the beam)
**Tracking procedure**

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only \( z \) will be fully calculated and stored
Kernel generation

Design

Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only \( z \) will be fully calculated and stored
- Tracking (full or partial)
**Tracking procedure**

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only $z$ will be fully calculated and stored
- Tracking (full or partial)
- Fill in each voxel center with Gaussian PDF
Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only $z$ will be fully calculated and stored
- Tracking (full or partial)
- Fill in each voxel center with Gaussian PDF
- PDF for each (proto)track is combined
Kernel generation

Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only $z$ will be fully calculated and stored
- Tracking (full or partial)
- Fill in each voxel center with Gaussian PDF
- PDF for each (proto)track is combined
- Fill $z$ “histogram” with maximum KDE value in $xy$
### Human learning
- Peaks generally correspond to PVs and SVs

### Challenges
- Vertex may be offset from peak
- Vertices interact

---

**Note:** All events from toy detector simulation

**Example of z KDE histogram**

<table>
<thead>
<tr>
<th>z values [mm]</th>
<th>Density of Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>150</td>
<td>50</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>250</td>
<td>150</td>
</tr>
</tbody>
</table>

**Design**

- Kernel
- LHCb PVs
- Other PVs
- LHCb SVs
- Other SVs

---

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019
Target distribution

Build target distribution

• True PV position as the mean of Gaussian
• $\sigma$ (standard deviation) is 100 $\mu$m (simplification)
• Fill bins with integrated PDF within $\pm 3$ bins ($\pm 300 \mu$m)
Neural network architecture

Design

Inputs
- 1
- 2
- 3
- ...
- 25
- 26
- ...
- 3998
- 3999
- 4000

25 Channels
- 1
- 2
- 3
- ...
- 15
- 16
- ...
- 3998
- 3999
- 4000

Convolution
- Width: 25
- Channels: 1 → 25

25 Channels
- 1
- 2
- 3
- ...
- 15
- 16
- ...
- 3998
- 3999
- 4000

Convolution
- Width: 15
- Channels: 25 → 25

25 Channels
- 1
- 2
- 3
- ...
- 15
- 16
- ...
- 3998
- 3999
- 4000

Convolution
- Width: 15
- Channels: 25 → 25

25 Channels
- 1
- 2
- 3
- ...
- 4
- 5
- ...
- 91
- 92
- ...
- 3998
- 3999
- 4000

Convolution
- Width: 5
- Channels: 25 → 1

1 Channel
- 1
- 2
- 3
- ...
- 5
- ...

Output
- 1
- 2
- 3
- ...

Leaky relu

Leaky relu

Leaky relu

Leaky relu

Softplus

A hybrid deep learning approach to vertexing

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019
**Approach**

- Symmetric cost function: low FP but low efficiency
- Adding asymmetry term controls trade-off for FP vs. efficiency
False Positive and efficiency rates

### Results

**Search for PVs (handwritten, maybe not optimal):**
- Search $\pm 5$ bins ($\pm 500\mu m$) around a true PV
- At least 3 bins with predicted probability $> 1\%$ and integrated probability $> 20\%$.

**Tunable efficiency vs. FP:**
- The asymmetry parameter controls FP vs. efficiency

---

A hybrid deep learning approach to vertexing

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019  10/16
Compare predictions with targets: Examples

Event 5 @ 197.4 mm: PV found

True: 197.461 mm
Pred: 197.396 mm
Δ: -65 µm

Event 6 @ 36.1 mm: PV found

True: 36.068 mm
Pred: 36.400 mm
Δ: 332 µm

PV found example

A hybrid deep learning approach to vertexing
Fang, Schreiner, Sokoloff, Weisser, Williams
April 3, 2019
Compare predictions with targets: When it works

**PV found example**

- Event 0 @ 48.9 mm: PV found
- Target: True = 48.904 mm, Pred = 48.954 mm
- Masked: Δ = 50 µm

**Masked (<5 tracks) example**

- Event 0 @ 1.0 mm: Masked
- Target: Pred = 0.976 mm
- Masked (<5 tracks) example

A hybrid deep learning approach to vertexing

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019
Compare predictions with targets: When it fails

False Positive example

PV not found example

A hybrid deep learning approach to vertexing

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019  13/16
Future addition: xy information

Adding xy information
- Point of maximum z in xy available
- Extra information: sharp discontinuities between PVs
- Need iterative approach or “reduced importance”

What about a full 2D kernel?
- Not needed for LHCb currently (large xy, “low” z overlap)
- Might be useful for other detectors!
• Proof-of-Principle established: a hybrid ML algorithm using a 1-dimensional KDE processed by a 5-layer CNN finds primary vertices with efficiencies and false positive rates similar to traditional algorithms.

• Efficiency is tunable; increasing the efficiency also increases the false positive rate.

• Adding information should improve performance.
  • can add KDE (x,y) information to algorithm
  • can associate tracks to PV candidates, then iterate.

• Next steps: train with full LHCb MC and deploy inference engine in LHCb Hlt1 framework.

• Beyond LHCb
  • approach might work for ATLAS and CMS (in 2D?);
  • algorithm is an interesting ML laboratory.
Final words

Future plans

Source code:
- [https://gitlab.cern.ch/LHCb-Reco-Dev/pv-finder](https://gitlab.cern.ch/LHCb-Reco-Dev/pv-finder)
- Runnable with Conda on macOS and Linux
  - Run: `conda env create -f environment-gpu.yml`
  - Python 3.6+ and PyTorch used for machine learning code
  - Generation now available too using the new Conda-Forge ROOT and Pythia8 packages

Supported by:
- NSF OAC-1836650: IRIS-HEP
- NSF OAC-1740102: SI2:SSE
- NSF OAC-1739772: SI2:SSE
Questions?

Source code:
- [https://gitlab.cern.ch/LHCb-Reco-Dev/pv-finder](https://gitlab.cern.ch/LHCb-Reco-Dev/pv-finder)
- Runnable with Conda on macOS and Linux
  - Run: `conda env create -f environment-gpu.yml`
  - Python 3.6+ and PyTorch used for machine learning code
  - Generation now available too using the new Conda-Forge ROOT and Pythia8 packages

Supported by:
- NSF OAC-1836650: IRIS-HEP
- NSF OAC-1740102: SI2:SSE
- NSF OAC-1739772: SI2:SSE
More predictions with targets (1)

Event 5 @ 221.5 mm: PV found

- True: 221.595 mm
- Pred: 221.546 mm
- $\Delta$: -49 $\mu$m

Event 2 @ 114.6 mm: PV found

- True: 114.622 mm
- Pred: 114.597 mm
- $\Delta$: -26 $\mu$m

A hybrid deep learning approach to vertexing

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019 17/16
More predictions with targets (2)

Event 6 @ 129.3 mm: PV found
- True: 129.336 mm
- Pred: 129.337 mm
- Δ: 1 µm

Kernel Density

Target: 129.336 mm
Predicted: 129.337 mm
Masked: 1 µm

Event 6 @ 143.2 mm: PV found
- True: 143.224 mm
- Pred: 143.199 mm
- Δ: -25 µm

Kernel Density

Target: 143.224 mm
Predicted: 143.199 mm
Masked: -25 µm

A hybrid deep learning approach to vertexing

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019
A hybrid deep learning approach to vertexing

Fang, Schreiner, Sokoloff, Weisser, Williams

April 3, 2019  19/16
The VELO

Tracks
- Originate from vertices (not shown)
- Hits originate from tracks
- We only know the true track in simulation
- Nearly straight, but tracks may scatter in material

The VELO
- A set of 26 planes that detect tracks
- Tracks should hit one or more pixels per plane
- Sparse 3D dataset (41M pixels)
Questions for other experiments

- Beam width \((x, y)\): 40 \(\mu\text{m}\) for LHCb, what is yours?
- Transverse resolution: 5–15 \(\mu\text{m}\) for LHCb depending on number of tracks, what is yours?
- Longitudinal resolution: 40–100 \(\mu\text{m}\) for LHCb depending on number of tracks, what is yours?
- Cleaning up prototracks based on IP could simplify kernel
- Can prototracking be done in the triggers?