Lightweight Jet Reconstruction and Identification as an Object Detection Task

Adrian Alan Pol[1] Thea Aarrestad[1] Ekaterina Govorkova[1] Roi Halily[2] Anat Klempner[2] Tal Kopetz[2] Vladimir Loncar[1,4] Jennifer Ngadiuba[3] Maurizio Pierini[1] Olya Sirkin[2] Sioni Summers[1]

[1]European Organization for Nuclear Research (CERN) [2]CEVA Inc. [3]Fermi National Accelerator Laboratory [4]Institute of Physics Belgrade

Abstract

We apply deep learning object detection techniques based on convolutional blocks to end-to-end jet identification and reconstruction problem encountered at the CERN LHC. Collision events produced at the LHC and represented as an image composed of calorimeter and tracker cells are given as an input to a Single Shot Detection network. The algorithm, named PF-Jet-SSD performs simultaneous localization, classification and regression tasks to measure jet kinematic features. All in a single feed-forward pass. Besides this parallelization, we gain additional acceleration from network slimming, homogeneous quantization, and optimized runtime for meeting memory and latency constraints. We experiment with two quantization schemes. The first, using the Ternary Weight Network to represent the weights in ternary representation where each input and output feature has its own scale factor. The second, 8-bit quantized weights and activations. We compare these two quantization schemes with a single precision floating point in terms of accuracy and latency. We show that the Ternary Weight Network closely matches the performance of its full-precision equivalent and outperforms the state-of-the-art legacy algorithm. Finally, we report the inference latency on different hardware platforms and discuss future applications.

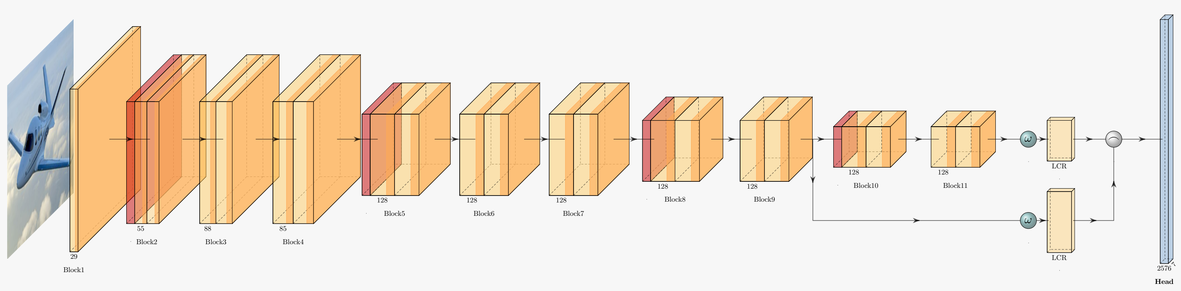
Contributions

- We introduce the **PF-Jet-SSD** algorithm to perform localization, classification and additional regression tasks to measure jet features in a single feed-forward pass (concurrently, or *single-shot*). We combine ideas from different fields of deep learning, i.e. object detection, attention mechanisms, pruning and quantization.
- We report acceleration from optimized runtime on different computing architectures in the context of memory and latency constraints.
- We generate and publicly share a dataset of simulated LHC collisions, pre-processed to be suited for computing vision applications similar to those discussed in this work, as well as for point-cloud end-to-end reconstruction. The dataset is available on Zenodo and it is accompanied by annotated jet labels, to be used as ground-truth during training.

Techniques

- **Jet images**. Project the lower level detector measurements onto a cylindrical detector and then unwrapped the inner surface of the calorimeter on a rectangle.
- **Single Shot Detection (SSD)**. A simple one-stage, anchor-based object detector (classification and localization of objects). At inference each anchor is refined by four coordinates (width, height, x and y) and predicts the category.
- **Efficient Channel Attention (ECA)**. Defined by $\boldsymbol{\omega = \sigma(\mathbf{W} \odot g(y))}$, where $\boldsymbol{y \in \mathbb{R}^C}$ is the feature map activation with channels $\boldsymbol{C}$, $\boldsymbol{g}$ is channel-wise global average pooling, $\boldsymbol{\sigma}$ is the Sigmoid function and $\mathbf{W}$ is a weight of a $\boldsymbol{1D}$ convolution.
- **Ternary Weight Network (TWN)**. Very aggressive strategy which reduces weight precision to $\{-1, 0, 1\}$. The approximated solution for threshold Δ based ternary function is $\Delta^* \approx 0.7 \cdot \boldsymbol{E}(|\mathbf{W}|)$. Scaling factor α minimize the Euclidian distance between full precision and ternary weights.

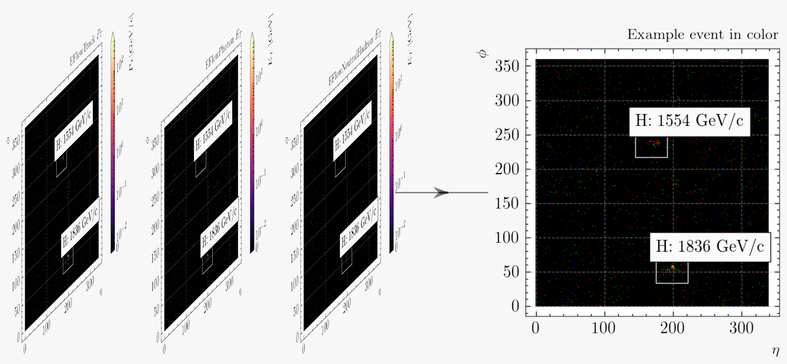
PF-Jet-SSD: Architecture



The convolution block (3×3 convolution followed by batch normalization and PReLU activation) is in yellow, average pooling (2×2 kernel) is in red, the detection head which is the output layer is in blue. ω is the attention module and $\frown$ is the concatenation. The numbers indicate the number of output channels in each block. `Block10` and `Block11` are removed at inference.

Dataset

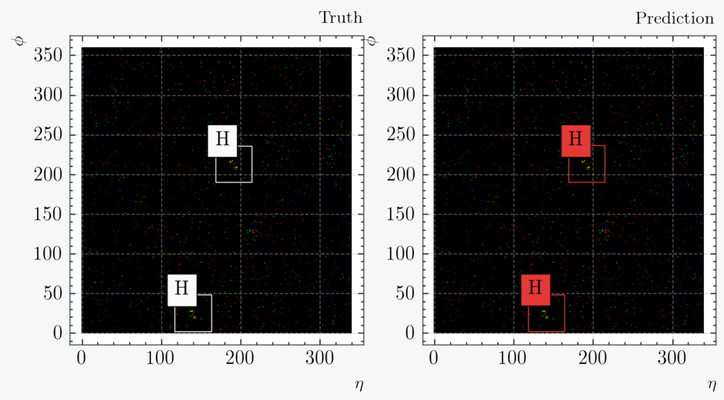
- This study considers 13 TeV proton-proton collision events (with *pileup*), in which Randall-Sundrum gravitons decay to $\boldsymbol{b\bar{b}}$, gg, qq, HH, WW, ZZ, or $\boldsymbol{t\bar{t}}$ final states.
- Events are generated with Pythia and the Compact Muon Solenoid (CMS) detector effects are emulated using the Delphes library.
- The labels for jets are obtained using generator-level *particle status* information.



Tracker information and energy deposits in CMS ECAL and HCAL translated to two-dimensional image (340×360 pixels). The bounding boxes correspond to ground truth with target label and momentum.

Training Procedure

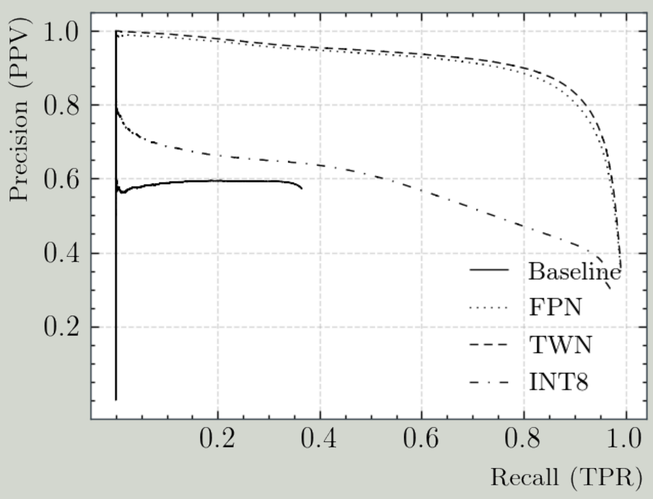
- **PF-Jet-SSD** network is implemented on a NVidia Tesla GPUs using PyTorch.
- We use 90 k and 36 k samples for training and validation respectively.
- Training in mixed-precision distributed across 3 GPUs.
- We minimize $\boldsymbol{\mathcal{L}_{SSD} = \mathcal{L}_c + \mathcal{L}_l + \mathcal{L}_r}$, where $\boldsymbol{\mathcal{L}_c}$ is the cross-entropy classification loss with smooth labels ($\alpha = 0.1$), $\boldsymbol{\mathcal{L}_l}$ is the localization loss and $\boldsymbol{\mathcal{L}_r}$ is the regression loss, both Huber loss ($\delta = 1$).
- Five steps of iterative pruning, each with 20 epochs of retraining.



An example of the PF-Jet-SSD at inference for one event with the input image and highlighted true labels (left) and predicted bounding boxes and classes (right).

Results: Detection

- Evaluation: precision (positive predictive value, PPV) and recall (true positive rate, TPR) curve, and an average precision metric (AP) on 90 k samples.
- The TWN network results are closely matching the results of the FPN.
- Poor results of 8-bit fixed-precision (INT8) network suggest that activation quantization is not straightforward and may benefit from mixed-precision setup.



Results: Latency and throughput

Results of the proposed algorithm on different architectures:

- Baseline: running native PyTorch inference on the Intel Xeon Silver 4114 CPU
- ONNX accelerated version on the same CPU.
- TensorRT optimized version on Nvidia Tesla V100.