

PAPER • OPEN ACCESS

Lightweight jet reconstruction and identification as an object detection task

To cite this article: Adrian Alan Pol *et al* 2022 *Mach. Learn.: Sci. Technol.* **3** 025016

View the [article online](#) for updates and enhancements.

You may also like

- [Peat fires and the unknown risk of legacy metal and metalloid pollution](#)
Colin P R McCarter, Gareth D Clay, Sophie L Wilkinson et al.
- [Envisioning a sustainable agricultural water future across spatial scales](#)
Tara J Troy, Laura C Bowling, Sadia A Jame et al.
- [Particle-based fast jet simulation at the LHC with variational autoencoders](#)
Mary Touranakou, Nadezda Chernyavskaya, Javier Duarte et al.



PAPER

OPEN ACCESS







Lightweight jet reconstruction and identification as an object detection task

RECEIVED
15 February 2022REVISED
14 June 2022ACCEPTED FOR PUBLICATION
17 June 2022PUBLISHED
4 July 2022

Original Content from this work may be used under the terms of the [Creative Commons Attribution 4.0 licence](https://creativecommons.org/licenses/by/4.0/).

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.



Adrian Alan Pol^{1,2,*} , Thea Aarrestad¹ , Ekaterina Govorkova¹ , Roi Halily³, Anat Klempner³, Tal Kopetz³, Vladimir Loncar^{1,4} , Jennifer Ngadiuba⁵ , Maurizio Pierini¹ , Olya Sirkin³ and Sioni Summers¹

¹ European Organization for Nuclear Research (CERN), Geneva, Switzerland

² Princeton University, Princeton, NJ, United States of America

³ CEVA Inc., Herzliya, Israel

⁴ Institute of Physics Belgrade, Belgrade, Serbia

⁵ Fermi National Accelerator Laboratory, Winfield Township, DuPage County, IL, United States of America

* Author to whom any correspondence should be addressed.

E-mail: adrianalan.pol@cern.ch

Keywords: high energy physics, jet tagging, jet reconstruction, jet images, object detection, attention mechanism, quantization aware training

Abstract

We apply object detection techniques based on deep convolutional blocks to end-to-end jet identification and reconstruction tasks encountered at the CERN large hadron collider (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 PFJet-SSD performs simultaneous localization, classification and regression tasks to cluster jets and reconstruct their features. This all-in-one single feed-forward pass gives advantages in terms of execution time and an improved accuracy w.r.t. traditional rule-based methods. A further gain is obtained from network slimming, homogeneous quantization, and optimized runtime for meeting memory and latency constraints of a typical real-time processing environment. We experiment with 8-bit and ternary quantization, benchmarking their accuracy and inference latency against a single-precision floating-point. We show that the ternary network closely matches the performance of its full-precision equivalent and outperforms the state-of-the-art rule-based algorithm. Finally, we report the inference latency on different hardware platforms and discuss future applications.

1. Introduction

The world's largest and most powerful particle accelerator, the CERN large hadron collider (LHC) [1], operates at a nominal proton-proton collision rate of 40 MHz. Due to storage constraints and technological limitations (e.g. fast enough read-out electronics), the volume of recorded data must be significantly reduced by the experiments operating around the accelerator ring. To this purpose, a set of algorithms collectively referred to as the *trigger system* are typically used to filter the incoming data stream. Trigger algorithms are designed to reduce the rate of recorded collision *events* (e.g. the collection of sensor readouts at each bunch crossing) while preserving the physics reach of the experiments. For example, at the Compact Muon Solenoid (CMS) experiment, the trigger system [2, 3] is structured in two stages using increasingly complex information and more refined algorithms:

- the Level 1 (L1) Trigger, implemented on custom-designed electronics; reduces the 40 MHz input to a 100 kHz rate in $<10 \mu\text{s}$.
- the high level trigger (HLT), a collision reconstruction software running on a computer farm; reduces the 100 kHz rate output of the L1 trigger to 1 kHz in $<150 \text{ ms}$.

With the planned LHC high-luminosity upgrade [4], the number of proton-proton collisions per second will surge approximately four-fold. The latency of legacy reconstruction algorithms will increase by more

than the factor of three as they may suffer from execution time scaling worse than linearly [5]. Along with the computing infrastructure upgrades, it is worth investigating solutions that could execute many tasks at once, while retaining accuracy and benefiting from the additional speedup offered by parallel computing architectures. Deep neural networks, such as those used for computer vision tasks, are an obvious candidate in this endeavour.

The majority of particles produced in LHC events are unstable and immediately decay to lighter particles. The new particles can decay themselves to others in a so-called decay chain. Such a process terminates when the decay products are stable particles, e.g. charged pions. This collimated shower of particles with adjacent trajectories is called a *jet*. Jets are central to many physics studies at the LHC experiments [6–9]. In particular, a successful physics program requires aggregating particles into jets (*jet clustering*), an accurate determination of the jet momentum (*momentum measurement*) and the identification of which particle kind started the shower (*jet tagging*) [10–13].

In this work, we show how jet clustering, momentum measurement, and tagging could all be handled simultaneously on parallel computing architectures. Besides the practical advantages of our approach, one could benefit from multitask learning when accomplishing more tasks at once [14]. For instance, a classifier and a regression running at once can learn that calibration constants depend on the nature of the jet, an issue which is now handled with ad-hoc post-processing [15], i.e. when factorizing the reconstruction problem to energy regression and tagging the overall performance may drop for both. Our main contributions are as follows:

- We introduce the **PFJet-SSD** algorithm to perform localization, classification and additional regression tasks on jets 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, network slimming and quantization.
- We report acceleration on different computing architectures.
- We generate and publicly share a dataset of simulated LHC collisions, pre-processed to be suited for computer 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 [16] and it is accompanied by annotated jet labels, to be used as ground truth during training.

We use the CMS detector and trigger system as an illustrative example. One could apply the same approach to other detectors, adapting the architecture to the detector granularity and latency constraints. The dataset, instructions, and code to fully reproduce our results are available at <https://github.com/AdrianAlan/PF-Jet-SSD>.

The remainder of this paper is structured as follows. In section 2 we review the key building blocks for this work, i.e. jet images, single-shot detection, attention mechanisms, and efficient model design. In section 3 we introduce the PFJet-SSD model and its quantized variants. In section 4 we describe the dataset and the training procedure. Finally, in sections 5 and 6 we discuss the results and future directions, respectively.

2. Techniques

In this section, we review the background techniques for this work, i.e. jet images, single-shot detection, attention mechanism and designing efficient inference networks with pruning and quantization. We examined architecture suggestions from [17], which lists methods for designing efficient networks for computer vision tasks achieving state-of-the-art results. Some of these methods, e.g. GELU activation layer [18], are currently unsupported by SensPro, our target hardware (see section 5.2). Thus we excluded them from this work but we suggest they are examined in further optimization studies.

2.1. Jet images

Traditional approaches to jet tagging rely on features, such as jet substructure, designed by experts that detect characteristic energy deposit patterns [19–27]. In recent years, several studies applied computer vision for event reconstruction at particle colliders, e.g. [28–43]. This was obtained by projecting the lower level detector measurements of the emanating particles onto a cylindrical detector and then unwrapping the inner surface of the calorimeter on a rectangle. Such information was further interpreted as an image with calorimeter cells as pixels, where pixel intensity maps the energy deposit of the cell, i.e. *jet images*. This approach was also applied to end-to-end reconstruction, considering not just the individual jet but the whole event [44, 45]. Building on these works, we extend the end-to-end reconstruction to include a localization task, merging the jet clustering and classification tasks in a single operation. Centralized computing

environments are the only viable options for this: end-to-end approaches require as input a raw data representation, which is not available with reduced analysis data formats. For this reason, we also consider how the model could be compressed to reduce computing footprint, having in mind an approach optimized for a trigger application.

2.2. Single shot detection

Object detection is a fundamental task in computer vision. It is defined as the classification of objects from predefined categories in the image along with their precise spatial locations. The spatial location and extent of an object can be defined coarsely using a bounding box, which is an axis-aligned rectangle tightly bounding the object. Modern object detection focuses on using primarily convolutional neural networks (CNNs) as the building block. Deep learning object detection achieved state-of-the-art results in tasks such as face [46] or pedestrian detection [47]. For a general survey on this subject, see [48, 49].

Deep-learning-based object detection models are typically divided into one- [50–54] or two-stage [55–59] detectors. Two-stage detectors generate a sparse set of regions with a high probability of an object being present first (region proposals), followed by a simple classification step. This two-step process is inefficient for real-time applications, due to task serialization. Single-step approaches classify and regress object locations concurrently (in a single feed-forward pass) and as such tend to achieve lower accuracy than two-stage detectors but are simpler and significantly more latency and memory efficient, hence having greater applicability to online problems.

The single-shot multibox detector (SSD) [60], is a simple one-stage, anchor-based detector. First, a set of default regions in an image with a fixed shape and size is predefined to discretize the output space of bounding boxes, called anchors. These anchors have a diverse set of shapes to detect objects with different dimensions, i.e. multiple scales and aspect ratios. Based on the ground truth, the object locations are matched with the most appropriate anchors to obtain the supervision signal for the anchor estimation. At inference, each anchor is refined by four box coordinates (width, height, x and y offsets) and predicts the categorical probabilities. To avoid a huge number of negative proposals dominating training gradients, hard negative mining is used to train the network, which fixes the foreground and background ratio [61]⁶. Alternatively, a focal loss [52] could be used. In this case, the price to pay would be more hyperparameters to tune. The SSD architecture is fully convolutional, with initial layers based on a pre-trained backbone architecture, such as VGG-16 [62], followed by extra convolutional and pooling layers which progressively decrease image size and thus increase the receptive field. The information in the last layer may be too coarse spatially to allow precise localization and at the same time, detecting large objects in shallow layers is non-optimal without large enough receptive fields. As a countermeasure for this issue, the SSD performs detection over multiple scales by operating on multiple feature maps, i.e. at different depths of the network. Each of these feature maps is responsible for detecting objects according to their receptive field. To detect large objects and increase receptive fields extra convolutional feature maps were added to the backbone architecture. The final prediction is made by merging all detection results from different feature maps followed by a non-maximum suppression (NMS) [60] step and producing the final detection information. NMS removes duplicate predictions originating from multiple anchors.

2.3. Attention mechanisms

Visual attention gates (AGs), e.g. [63–65], learn to suppress feature activations in irrelevant regions in an input image without additional supervision. At inference, the gates generate soft region proposals to highlight salient features useful for a specific task. Recently, the performance of deep CNNs on visual tasks was improved with scale-aware [66, 67], spatial-aware [68, 69] and channel-wise [70, 71] attention. On the contrary, most of the attention modules inevitably increase model complexity. Efficient channel attention (ECA) gate [72] is a soft attention mechanism that addresses this issue. It avoids dimensionality reduction and captures cross-channel interaction efficiently. ECA gate ω is given by $\omega = \sigma(\mathbf{W} \odot g(y))$, where $y \in \mathbb{R}^C$ is the feature map activation with channels C , g is channel-wise global average pooling, σ is the Sigmoid function and \mathbf{W} is a weight tensor of a 1D convolution of filter size k .

2.4. Quantization

Optimizing deep neural networks for efficient inference is an essential task in modern machine learning pipelines due to limitations presented by edge devices. Models should provide high accuracy with a

⁶ By background we refer to the areas without target objects, i.e. jets.

minimum of computing time and resources. Apart from accelerating inference online, e.g. through parallelization or hardware optimizations, models can be optimized offline, through compression [73].

Network compression [74] is a common technique to reduce the number of operations and model size, energy consumption, and over-training of deep neural networks. As neural network synapses and neurons can be redundant, compression techniques attempt to reduce the total number of them, effectively reducing multipliers. Several approaches have been successfully deployed without much loss in accuracy, including selective removal of parameters based on a particular ranking and regularization, i.e. parameter pruning [75–77], compact network architectures [78–80], and reducing the precision of operations and operands, i.e. quantization [81–88].

It has been observed that reducing the precision of the calculations, i.e. weights and biases, has little impact on performance compared to speedup and resource usage gains. This includes moving away from 32-bit floating-point calculations (or *full-precision*, FP) to fixed points, reducing bit-width and weight sharing. An example of a very aggressive strategy is reducing weight precision to ternary values restricted to $\{-1, 0, 1\}$ only, called ternary weight network (TWN) [89]. The quantization is performed during training, using a straight-through estimator [81], where ternary weights are used during the forward and backward propagation but not during the parameter update. To quantize the full precision weights \mathbf{W} to ternary ones \mathbf{W}^* , TWN uses a threshold value Δ :

$$\mathbf{W}^* = \begin{cases} +1 & \text{if } \mathbf{W} > \Delta \\ 0 & \text{if } |\mathbf{W}| \leq \Delta, \\ -1 & \text{if } \mathbf{W} < -\Delta \end{cases}$$

with approximated solution $\Delta^* \approx 0.7 \cdot E(|\mathbf{W}|)$, where E is the expectation value. To make the network perform well, TWN minimizes the Euclidian distance between \mathbf{W} and \mathbf{W}^* along a non-negative scaling factor α that can be implemented with per-network, per-layer or per-channel granularity, transforming the weights to $\alpha\mathbf{W}^*$. For any Δ the optimal α is computed as: $\alpha_{\Delta}^* = \frac{1}{|\mathbf{I}_{\Delta}|} \sum_{i \in \mathbf{I}_{\Delta}} |\mathbf{W}_i|$, where $\mathbf{I}_{\Delta} = \{i | |\mathbf{W}_i| > \Delta\}$ and $|\mathbf{I}_{\Delta}|$ denotes number of elements in \mathbf{I}_{Δ} .

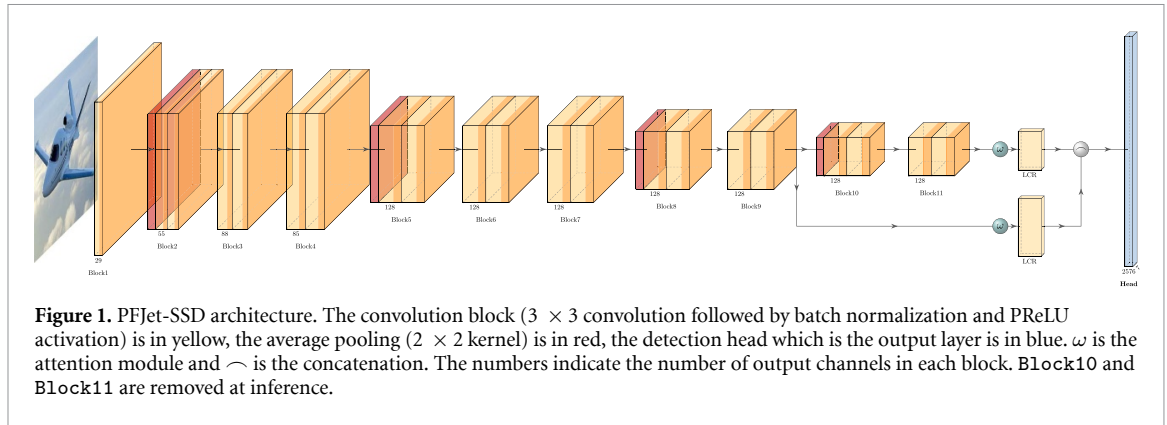
3. Methodology

The PFJet-SSD architecture is shown in figure 1. We modify the original SSD architecture [60] and Jet-SSD architecture proposed in [90]. Having in mind an HLT application with a typical latency of ≈ 150 ms, we extend the event image representation to include the information from the charged-particle reconstruction. We do so by adding a *tracker* channel to the image, in front of the calorimeter channels already introduced in [90]. We use a lightweight MobileNet architecture [78] as a backbone for our detector which replaces the convolution operation with a combination of depthwise and pointwise versions. Each convolution is followed by a batch normalization [91, 92] and parametric rectified linear unit (PReLU) [93] activation layers. We use the AveragePool layer to decrease the size of the feature map. The extra convolutional layers proposed by the original SSD do not contribute to accurate detection (recall the remark about the increasing receptive field from section 2.2). This is due to the size of the jets. As done in [90] we remove these layers already at the training time. Retaining the deeper layers of the backbone, i.e. Block10 and Block11, does not show improvements at inference but is necessary during training due to additional signals during back-propagation. Hence, these deeper layers are only purged after training, i.e. the concatenation layer ignores them only at inference. This alone reduces the number of parameters in the final model by approximately 30%.

We add two new modules to the network. First, the initial convolutional layer is now followed by spatial dropout [94] (with $p = 0.1$). Second, we attach the ECA gate [72] (with $k = 3$) before the localization classification regression (LCR) layer.

The detection head, which is the concatenation of LCR layers, outputs correspond to jet class, localization (η and ϕ offsets) and p_T value (see the definitions in section 4.1). One might easily extend this output to include jet mass regression as well (we left this out for simplicity). Each row in the detection head corresponds to an anchor box, i.e. fixed position in the image. For localization, we regress only the centre of the jet, as we can determine its size from its class. For wide jets we assume $\Delta R = 0.8$ –46 px, for narrow jets $\Delta R = 0.4$ –23 px. This allows us to set only one scale and one aspect ratio for anchors in each feature map which reduces the complexity of the network. The detection head is an input to the NMS layer.

We use magnitude pruning [95] during training to find the optimal allocation of resources between layers. Unstructured pruning generality leads to a higher compression rate and/or higher accuracy when compared to the structured version, but it requires special software or hardware accelerators to fully benefit



from it. Since the outcome of an unstructured pruning is a sparse tensor, one needs a dedicated way to handle sparse memory access on hardware to turn pruning compression into a computational advantage at inference time. We use an alternative, a version of structured pruning that removes whole channels in a convolutional block slimming the network without increasing sparsity. We target the hardware implementation that benefits from fusing batch normalization and convolution parameters at runtime. Doing so, the target filter weights \mathbf{W} of block l are $\mathbf{W}_l = \gamma_l \mathbf{W}_l^{\text{conv}}$, where the \mathbf{W}^{conv} are the weights of the convolution and γ is the scale parameter of the affine transformation of the subsequent batch normalization layer. We thus add a regularizer that pushes the influence of filters down through batch normalization γ L1 penalty, similarly to [77, 96]. We scale this penalty based on the number of operations \mathcal{O} in each layer. The sparsifying regularizer $G(\gamma)$ is calculated as $G(\gamma) = \sum_l |\gamma_l| \mathcal{O}_l$. We mark channels to prune based on the γ distribution in each layer, using the rule: $|\gamma_l| < \mu_{|\gamma_l|} - \sigma_{|\gamma_l|}^2$, where $\mu_{|\gamma_l|} = \frac{1}{N} \sum_{i=1}^N |\gamma_l^i|$, $\sigma_{|\gamma_l|}^2 = \sqrt{\frac{1}{N} \sum_{i=1}^N (|\gamma_l^i| - \mu_{|\gamma_l|})^2}$ and N is the number of channels for each layer. When this rule is not sufficient to remove the specified number of channels we simply select the remaining ones based on ascending magnitudes of γ .

Also during training, we quantize the network to homogeneous 8-bit fixed point precision for both weights and activations and 2-bit TWN with layer- and channel-dependent scaling factors. For the latter, we experimented with a grace period of frozen quantization for which the Δ and α parameters remain unchanged. Training TWN in this manner may offer greater stability, i.e. weights have time to adjust to new parameters, but in our case, the final results did not improve.

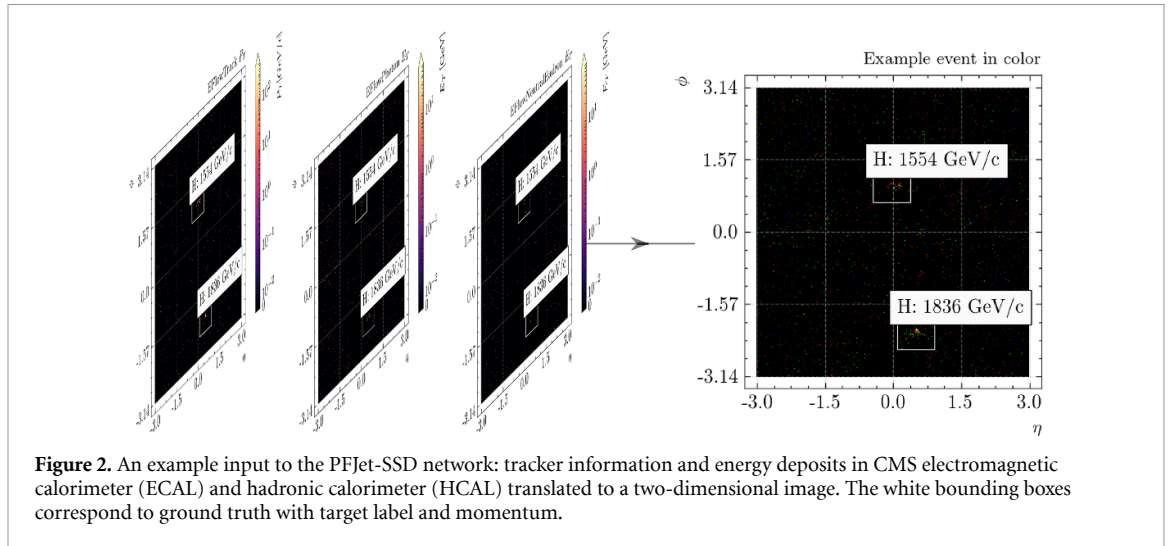
4. Experiments

In this section, we review the experimental dataset and training procedure used for the experiments:

4.1. Dataset

The input dataset consists of 13 TeV proton-proton collision events, in which Randall-Sundrum (RS) gravitons with 3.5 TeV mass are produced. This is a proxy of a sample that would give us jets of various kinds and populating a large spectrum of p_T ranges, i.e. RS gravitons decay to $b\bar{b}$, $g\bar{g}$, $q\bar{q}$, HH , WW , ZZ , or $t\bar{t}$ final states. The choice of this particular process is motivated by the possibility of creating well-defined jet pairs belonging to specific jet classes and with the same kinematic properties across classes. In addition to the hard collision, parasitic *pileup* collisions are also simulated, overlapping minimum bias events. The number of pileup collisions is sampled from a Poisson distribution. We note that this process populates a large spectrum of p_T ranges.

The detector effects and hadronization have an important effect on a jet substructure. Events are generated with Pythia [97]. We use the CMS Delphes [98] description to mimic the effect of detector reconstruction. To apply this algorithm to another detector (e.g. ATLAS), one would have to modify the geometry of the input layer to match the detector geometry. In addition, one would have to repeat the training. Other effects (e.g. theoretical uncertainties related to hadronization models) would be detector independent. These kinds of uncertainties also affect rule-based algorithms and are usually neglected at the trigger stage, where they are subdominant. These uncertainties are measured with data control samples at the analysis stage and, usually, they are mitigated by applying a selection on the offline object so that the trigger behaviour is stable. The same set of state-of-the-art procedures could be applied to the algorithm we present.



Being all this part of a standard data analysis workflow (and beyond the scope of this paper), we do not comment on this further.

The core of the CMS detector is a multi-layer silicon tracking device, operating in a 4 T magnetic field. Two calorimeter layers surround the tracker: the lead tungstate crystal ECAL is designed to stop particles whose main interaction is electromagnetic (photons and electrons); the brass and scintillator HCAL is designed to stop hadrons. They give a measurement of the energy of particles (charged and neutrals). Each of them is composed of a barrel and two endcap sections. Forward calorimeters extend the pseudorapidity (η) coverage provided by the barrel and endcap detectors. The calorimeter cells (towers) in the barrel region together with tracker cells are arranged in a fixed discrete space with fine segmentation in η and ϕ , where ϕ is the translated azimuthal angle. A more detailed description of the CMS detector, together with a definition of the coordinate system used and the relevant kinematic variables, can be found in [99].

Before the LHC, jets were usually reconstructed from their calorimeter deposits (known as CaloJet). With the start of the LHC, the CMS particle flow (PF) algorithm [100] demonstrated that the additional information from track reconstruction could increase the accuracy of jet reconstruction. In CMS, this was crucial to compensate for the poor energy resolution of the HCAL. In the long term, this strategy was found to be effective beyond jet momentum measurement, since the angular resolution of the tracking algorithm provided valuable information for jet tagging and substructure algorithms.

The PF algorithm for jet reconstruction was eventually adopted also by the ATLAS experiment [101]. Taking this as our starting point, we build our event image starting from the PF jet constituents (as returned by the Delphes PF algorithm), arranging the particles in three groups: charged particles, used to create the tracker channel; photons and electrons, used for the ECAL channel; neutral hadrons, used for the HCAL channel. In a real-life application, one could use the same approach or build the channels from the raw detector hits in the tracker, ECAL, and HCAL. The best approach to follow depends on the context of the application (e.g. online vs offline).

We unwrap the cylindrical detector to compose the final image which is formed by translating the calorimeter energy deposits and tracker momentum into pixels using ECAL granularity, which results in $340 \times 360 \times 3$ pixel samples. An example is shown in figure 2. Some previous studies on jet images implemented data pre-processing steps such as translation, rotation, re-pixelation, or inversion. However, in our study, we only limit the input to $\eta \in (-3, 3)$ and standardize pixel intensities.

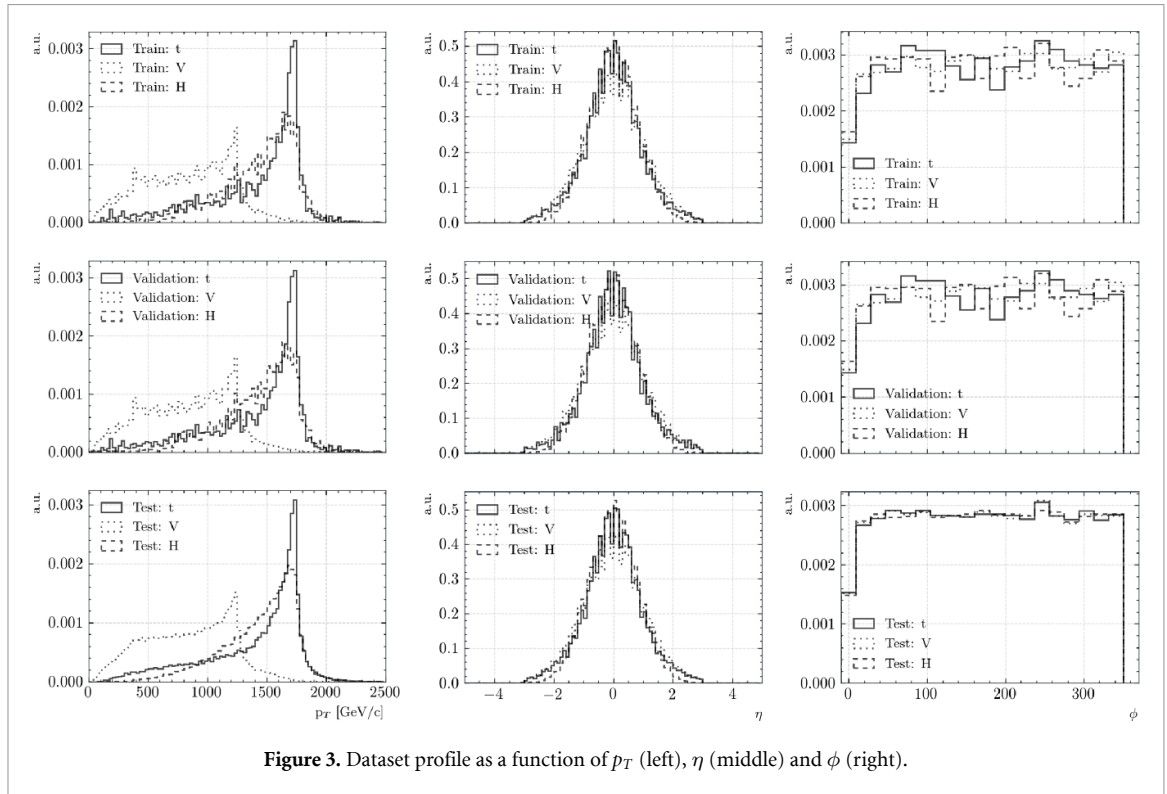
Jet labels are obtained using generator-level information. We assign the jet η (pseudo-rapidity and not rapidity as it is normally done in L1 trigger reconstruction), ϕ and p_T (transverse momentum) measurements to the properties of the same particle. The minimum jet p_T in the dataset is 7 GeV. Details on the dataset profile are given in table 1 which describes the jet statistics across datasets. Figure 3 shows the p_T , η and ϕ distributions.

4.2. Training procedure

The PFJet-SSD network is implemented on Nvidia Tesla GPUs using PyTorch [102]. For training, we use stochastic gradient descent with an initial learning rate of 10^{-3} with momentum set to 0.9 and weight decay to 0.0005. We train the network for 100 epochs with a batch size of 25, decreasing the learning rate by a factor of 2 after every 10 epochs after the 20th epoch. We use 90k and 36k samples for training and validation,

Table 1. Number of samples in the datasets.

	Train	Validation	Test
t jets	59 388	23 802	59 392
W/Z jets	118 701	47 493	118 832
H jets	59 967	23 997	59 978
Σ	238 056 (41.6%)	95 192 (16.7%)	238 202(41.6%)

**Figure 3.** Dataset profile as a function of p_T (left), η (middle) and ϕ (right).

respectively. The training is performed in mixed-precision to speed up computation and distributed across 3 GPUs. Thus, we replace the standard batch normalization layer with the SyncBatchNorm layer provided by PyTorch to synchronize statistics across the machines while training.

We minimize the following cost function:

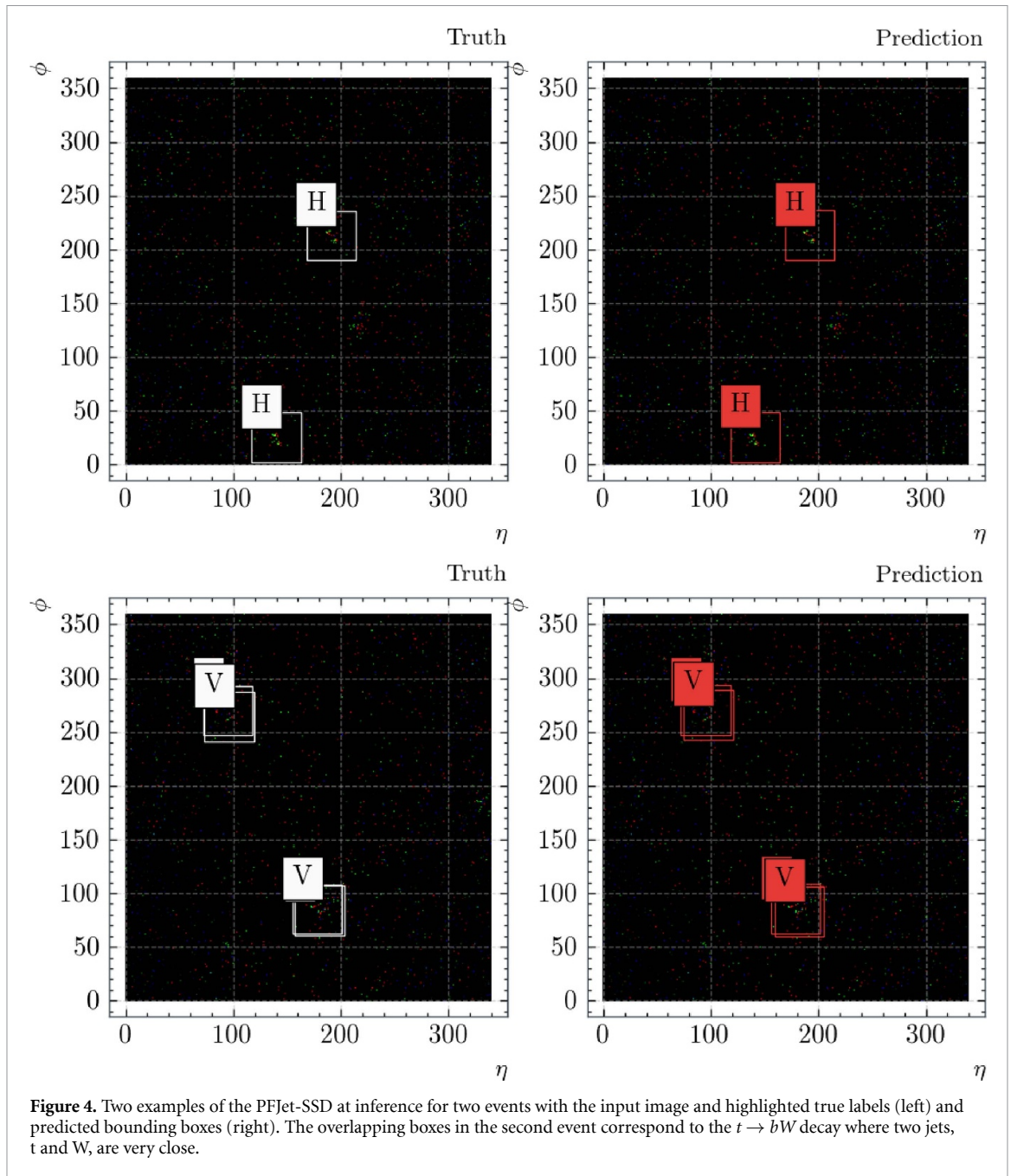
$$\mathcal{L}_{\text{SSD}} = \mathcal{L}_c + \mathcal{L}_l + \mathcal{L}_r,$$

where the \mathcal{L}_c is the classification loss, the \mathcal{L}_l is the localization loss, the \mathcal{L}_r is the regression loss. We use cross-entropy with smooth labels ($\alpha = 0.1$) for classification [103], and Huber loss [57] ($\delta = 1$) for localization and regression.

A common challenge when training object detection models from scratch is the insufficient amount of training data which may lead to overfitting⁷. Thus it is common to see practitioners pre-loading weights from pre-trained classification models on the real-world ImageNet [104] dataset. We found that such a procedure slows down our learning as the real-world images have little relation to our calorimeter images. The full precision network (FPN) can learn faster by using Xavier uniform initialization [105] (which helps with the sparsity of the input). We also augment the training dataset by random flips along η and ϕ dimensions, which we find to greatly stabilize the training. We did not experiment with other augmentation techniques such as changing brightness, contrast, saturation and hue as jets are not invariant to such transitions. The experiments with other commonly used techniques such as Mix-Up [106] or Mosaic [107] yield subpar results, again. This is likely because of the different nature of our input.

We perform five steps of iterative pruning, each with 20 epochs of retraining a gradually decreasing number of channels in each block. We then retrain the network for the last time for 100 epochs. We found out that pre-loading FPN weights when training the quantized versions, i.e. TWN and 8-bit fixed-precision (INT8) network, greatly speeds up convergence.

⁷ That is not a problem in our case as we can generate more events with low cost.



5. Results

In this section, we present the detection and latency performance of PFJet-SSD.

5.1. Detection performance

As a proof of concept, we investigate the tagging of the top-quark (t), W and Z bosons (V) and Higgs boson (H) jet. An example of the PFJet-SSD output is shown in figure 4. PFJet-SSD outputs predicted categorical label, prediction confidence and the centre coordinates of the object. In object detection true positive is defined as prediction with predicted category equal to the ground truth label and intersection over union (IoU) above the predefined threshold, usually 0.5. Successful prediction meets both criteria, otherwise, it is considered as a missed detection. In our case we substitute the IoU requirement with the distance metric $d = \sqrt{\Delta\phi^2 + \Delta\eta^2} < 33$ pixels as we regress only the centre of the box and box dimensions are universal across target classes.

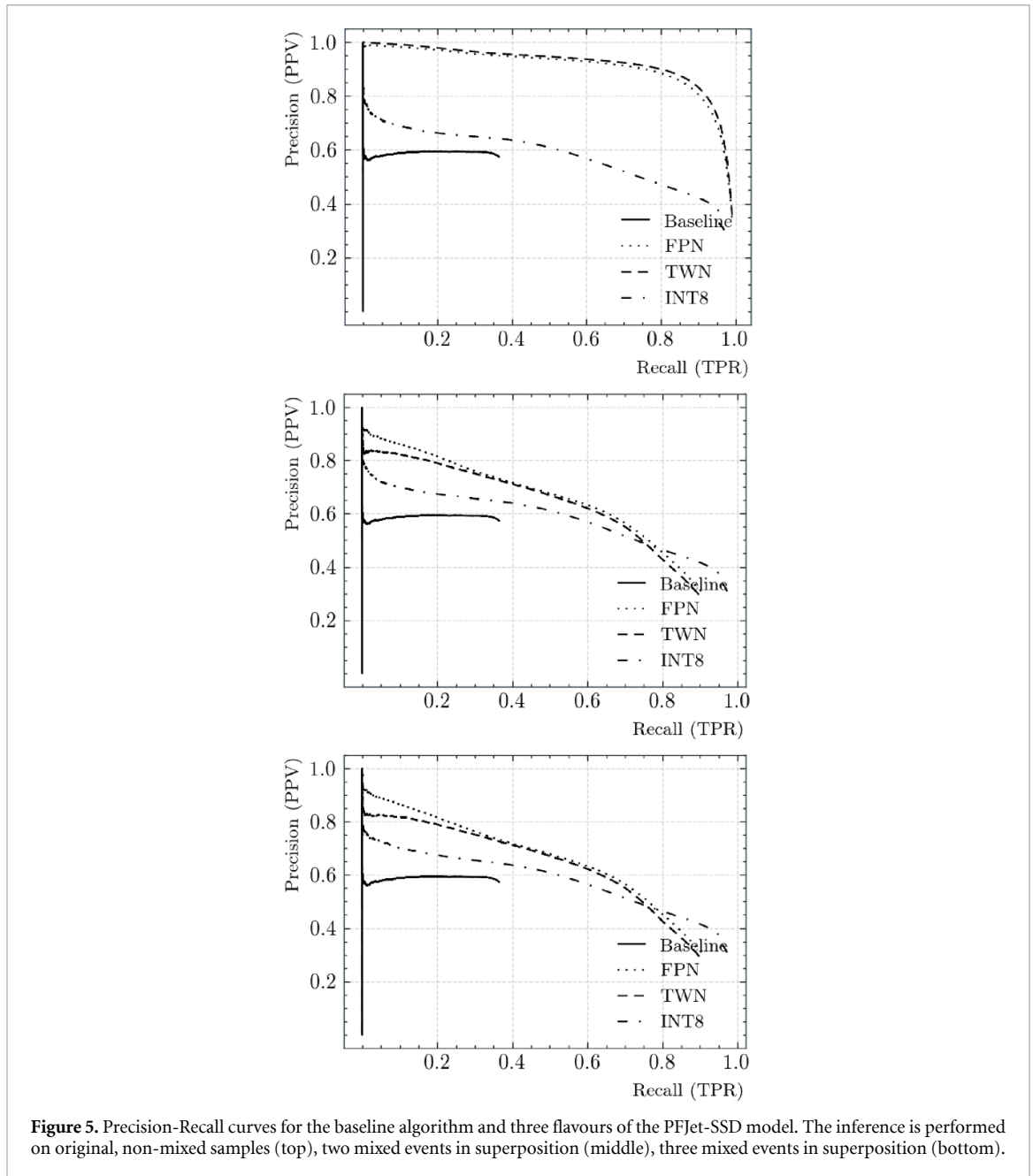


Figure 5. Precision-Recall curves for the baseline algorithm and three flavours of the PFJet-SSD model. The inference is performed on original, non-mixed samples (top), two mixed events in superposition (middle), three mixed events in superposition (bottom).

Our investigation into inference does not find any systematic issues. Occlusion, such as the one in $t \rightarrow bW$ decay, where jets are near, is not an obstacle against correct detection. Also, the jets close to the image edges are, generally, correctly classified.

To evaluate the model we use precision (or positive predictive value, PPV, $\frac{TP}{TP+FP}$) and recall (true positive rate, TPR) curve, and an average precision metric (AP), see figure 5. Intuitively, precision measures how accurate the predictions are while recall measures the quality of the positive predictions. Collectively, they determine how well the found set of jets corresponds to the set we expect to find. To draw a precision-recall (PR) curve, the predictions are first sorted in order of confidence followed by calculation of PPVs and TPRs for each confidence threshold. We held out 90k samples as our test dataset. The TWN network results are closely matching the results of the FPN. TWN benefits from the long retraining period, as it yields marginally better AP. For performance details across target jet classes see table 2.

For non-mixed samples in figure 5 TWN and FPN remarkably agree and yield an appealing precision for any given recall. Fix-point INT8 network drops in detection precision hinting that the network is sensitive to activation but not weight quantization. Hence, in the future, a mixed-precision should be explored as it is likely that not all layers contribute equally to this reduced performance. All flavours of the PFJet-SSD

Table 2. Detection performance for the baseline and PFJet-SSD algorithms, reporting the number of parameters (NoP), the number of operations (NoOps), the precision of weights/activations (W/A), average precision (AP) and precision at 0.3 (P@R = .3) and 0.5 (P@R = .5) recall. The table does not report parameters and bit precision for the baseline as it is a non-parametric method: not applicable (N/A). The baseline is also unable to reach 0.3 and 0.5 in several cases: no statistics (N/S).

		Physics baseline	PFJet-SSD		
			FPN	TWN	INT8
NoP		N/A		111 228	
NoOps		N/A		1.095G	
W/A		N/A	32/32	2/32	8/8
AP		.161	.848	.857	.566
t jet	AP	.420	.865	.872	.473
	P@R = .3	.736	.985	.988	.531
	P@R = .5	.627	.975	.980	.453
W/Z jet	AP	.245	.847	.859	.629
	P@R = .3	.584	.944	.955	.673
	P@R = .5	N/S	.929	.943	.653
H jet	AP	.107	.860	.872	.335
	P@R = .3	N/S	.992	.996	.453
	P@R = .5	N/S	.978	.986	.400

outperform the physics baseline. We also experimented with two and three events overlaid as the input to the network. This creates much noisier input and results in visibly reduced performance of PFJet-SSD. However, the network was not trained on such samples and such a drop is expected. Besides, the difference between two and three mixed events is minor. In the future, we suggest training the network with Mix-Up [106] or Mosaic [107] augmentations (techniques for mixing multiple samples) which could improve performance on noisier inputs.

Throughout, we compare PFJet-SSD to the *baseline* which is a physics-based algorithm combining a jet soft-drop mass [108], m , selection (under a specific mass hypothesis) and threshold requirement on the appropriate ratio of N-subjettiness [109] variables, τ . In particular, we require $105 < m < 210$ GeV for t jets and use the τ_3/τ_2 N-subjettiness ratio as a score defined in [0, 1]. With this score we make a performance assessment that we can directly compare to that obtained with the PFJet-SSD algorithm. Similarly, we require $65 < m < 105$ GeV for V jets and $105 < m < 140$ GeV for H jets, using the τ_2/τ_1 N-subjettiness ratio as a score for these baseline taggers. This physics-motivated baseline has performance that is typical of a rule-based state-of-the-art substructure jet tagger, with the typical recall of 0.3 for the precision of 0.6.

Figure 6 shows the dependence of the precision at fixed recall across different jet classes. The precision is rather flat in all cases. The TWN results match closely the FPN ones, while an overall drop in performance (approximately constant across η , ϕ , and p_T) is observed for the INT8 network. A drop is observed at the boundaries of the η region, as a consequence of jets leaking out of acceptance at the edge of the endcaps (missing information of a part of the shower). Such a drop is not observed in the ϕ dimension suggesting that the network can handle the periodicity of the image. The precision across p_T stays relatively flat, however, the sudden drop in the high p_T region of V jets is due to the low number of samples in that region, see the details in section 4. Notice that the drop in TRP at low jet p_T for top, W/Z, and H tagging is induced by transition from a boosted-jet to a resolved jets regime. Despite the fact that there so far little use of boosted jets from heavy particles in this low p_T regime, it is interesting to notice that the drop in efficiency of the PFJet-SSD in the low- p_T regime is less pronounced than for the baseline algorithm, which could be interesting to increase the reconstruction efficiency in transition between boosted and resolved topologies.

Figure 7 shows the residual in the determination of η and ϕ and the ratio of the reconstructed-to-true jet p_T , as a function of the jet p_T for the different classes.

Finally, we visualize the most repeating filters of the TWN in figure 8. Remarkably, the network optimizes to use a set very similar to the commonly used ones, e.g. smoothing, corner detection or edge detection filters.

5.2. Latency and power measurements

We investigate the latency and throughput of the proposed algorithm on architectures where parallel computing is more adequate. We compare the baseline, running native PyTorch inference on the Intel Xeon Silver 4114 CPU with ONNX accelerated version and TensorRT optimized version on Nvidia Tesla V100. Results are given in figure 9, separately for CPUs and GPUs. Having in mind an offline application, one could

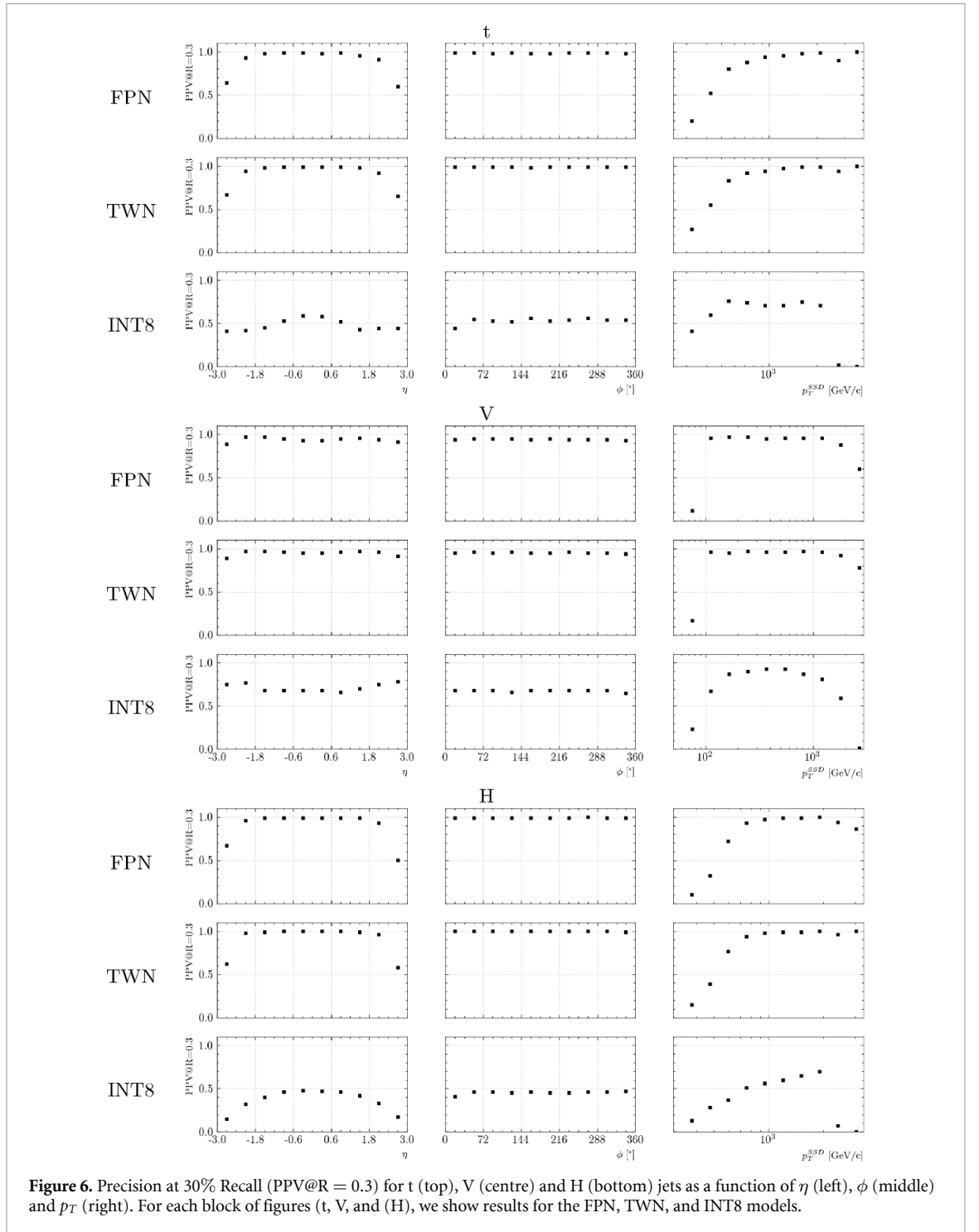


Figure 6. Precision at 30% Recall (PPV@R = 0.3) for t (top), V (centre) and H (bottom) jets as a function of η (left), ϕ (middle) and p_T (right). For each block of figures (t, V, and H), we show results for the FPN, TWN, and INT8 models.

maximize the throughput by running the network at once across batches of events, e.g. implementing the inference-as-a-service concept discussed in [110].

While the inference-as-a-service paradigm could also be implemented online, the current design of HLT farms foresees that processing parallelization is achieved by sending different events to different computing units. In this context, the batch size is constrained to one, since the inference of the proposed SSD model happens per event. In this case, execution on CPU would be borderline, within the average event processing latency but consuming most of it. On the other hand, moving the execution to a GPU would reduce the execution time to negligible levels. This could be particularly interesting under the assumption that GPUs would be used to run the local reconstruction [111–113] and the creation of PF candidates [114].

Deep learning inference at scale requires high power consumption, especially with the use of GPUs and CPUs. It is possible to keep the power and die area at more manageable levels by deploying an AI-specific

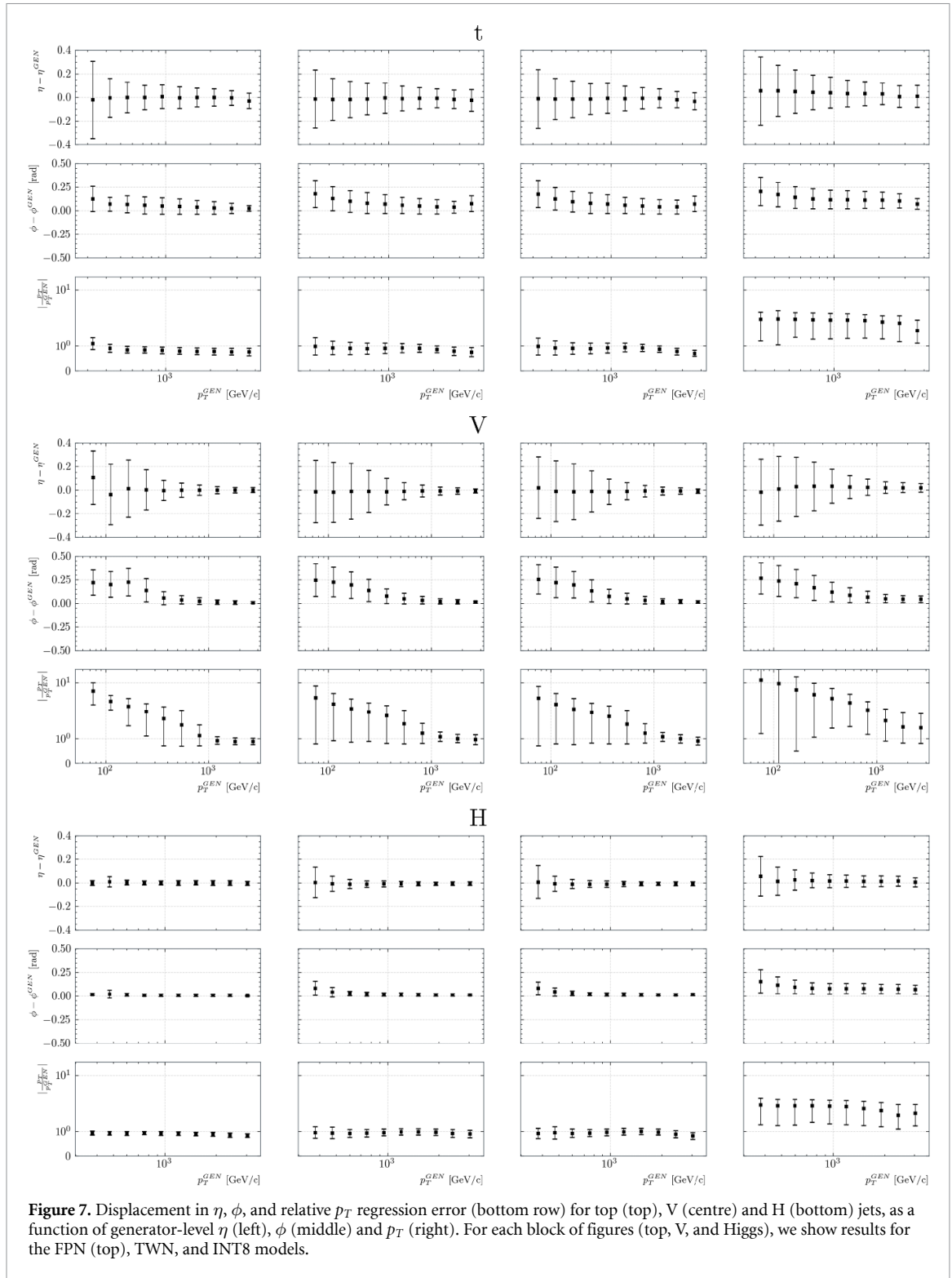


Figure 7. Displacement in η , ϕ , and relative p_T regression error (bottom row) for top (top), V (centre) and H (bottom) jets, as a function of generator-level η (left), ϕ (middle) and p_T (right). For each block of figures (top, V, and Higgs), we show results for the FPN (top), TWN, and INT8 models.

hardware platform as used in edge devices. Since edge devices usually operate on batteries where power is a limited resource, AI-specific hardware platforms for edge devices are highly power efficient. With smaller die areas, manufacturing costs and power consumption can be reduced.

SensPro is a family of ultra-light AI DSPs that can perform efficient inference while consuming only a fraction of the power and area used by GPUs and CPUs. CEVA's hardware platform for jet detection consists of a stack of ten SensPro (SP) DSP cores. Each core delivers 2 TOPS. An additional SP core is added to serve as a controller. This solution delivers 20 TOPS and can run TWN natively, reaching latency comparable to a GPU running an 8-bit network. This proposed layout has orders of magnitude lower area and power consumption than GPU and CPU, see table 3. The SP ultra-light solution can also be synthesized to an FPGA and used in collision detection.

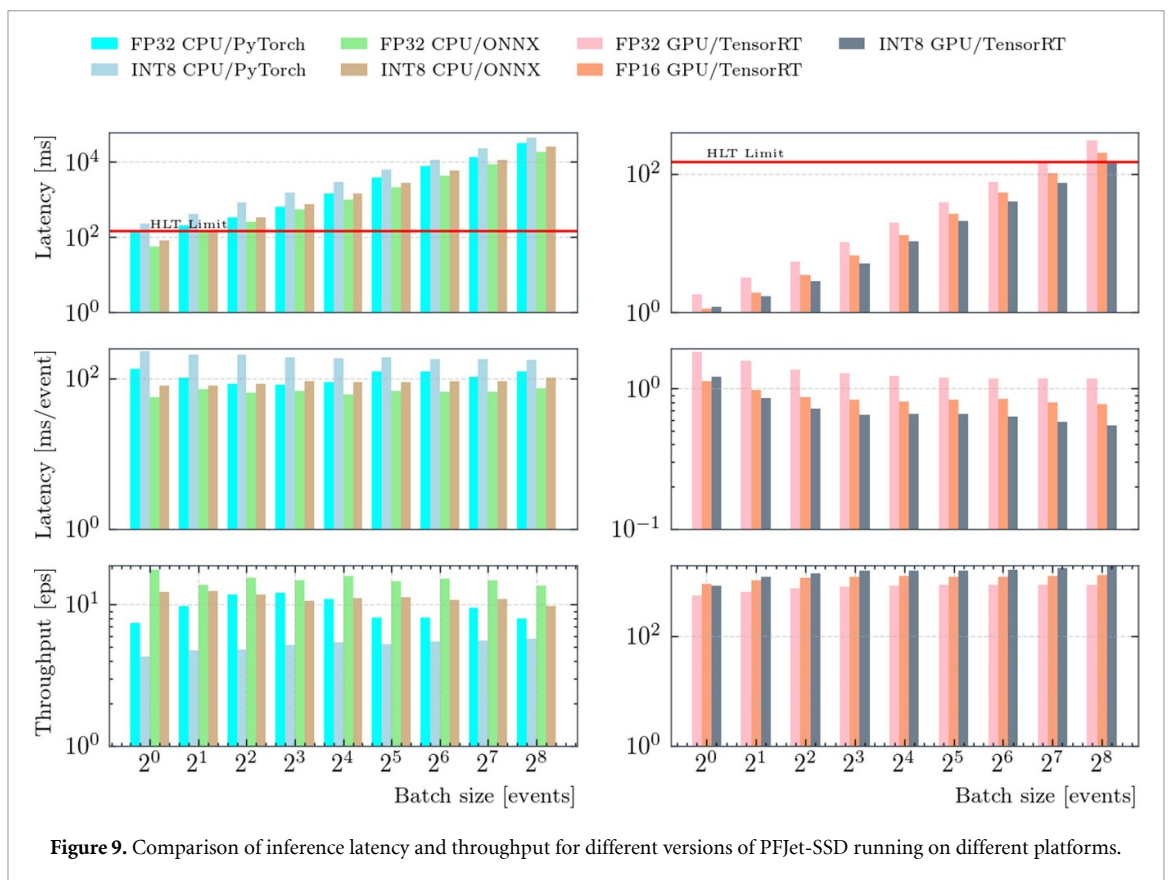
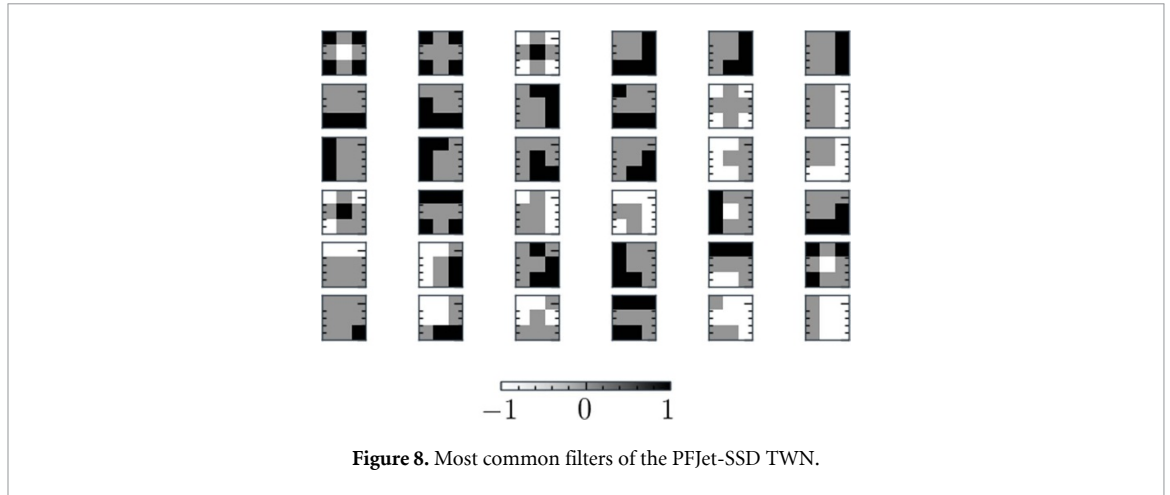


Table 3. Die area, power and latency measurements for different hardware architectures. The latency is measured for inference on a single input. The reason for this is that collision detection is done sequentially in real-time. The input data is fed to the network directly from the sensor without storing it.

	Die area (mm ²)	Power (W)	Latency (ms)
DSP CEVA SP1000 2x8	0.77	0.75	8.5
DSP CEVA 10xSP1000 + Controller 2x8	8.47	8.25	0.9
GPU Nvidia Tesla V100 8x8	815	250	1.1
CPU Intel Xeon Silver 4114. 32float	4294	85	134

6. Conclusions

We propose a fast and lightweight detection algorithm for jet tagging and reconstruction based on computer vision techniques. Naturally high precision and generalization are required, but nuisance factors of variations can break the algorithm. That makes this problem hard. Intra-class variations, such as perspective distortion,

e.g. rotation; densely arranged jets (occlusion); or blurred signatures (the detector response may not be clear) are common challenges. Besides, as jets are small objects, a reappearing issue with object detectors and background pileup may further disturb their visual appearance. Thus, robustness to detector effects, its imperfections and failures is required.

Even after a successful proof-of-concept deployment to production will still produce challenges as many of the problems lay outside of the simulation. More importantly, the real-time detection requirements force further investigations into more optimizations on algorithm and hardware runtime.

The PFJet-SSD paves the way for solving these issues. The algorithm did not experience accuracy drops during pruning, suggesting that the depth of the network is more important than the width. The number of channels can likely be reduced further and thus speed up computations. We observed a gap between TWN and INT8 performance which suggests to us that the optimal quantization level could be achieved through mixed-precision, a possible direction for future studies.

From the physics point of view, the algorithm manifests an interesting behaviour in low momentum regions out of reach for the baseline model, see high precision results in figure 6, which could help increasing the reconstruction efficiency for all-hadronic decays of heavy particles in the transition regime between boosted and resolved topologies.

Data availability statement

The data that support the findings of this study are openly available at <https://doi.org/10.5281/zenodo.4883651>.

Acknowledgments

We thank Loukas Gouskos and Huilin Qu for useful discussions and suggestions. A A P, M P, S S and V L are supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 772369). A A P is supported by CEVA under the CERN Knowledge Transfer Group.

ORCID iDs

Adrian Alan Pol  <https://orcid.org/0000-0002-9034-0230>
Thea Aarrestad  <https://orcid.org/0000-0002-7671-243X>
Ekaterina Govorkova  <https://orcid.org/0000-0003-1920-6618>
Vladimir Loncar  <https://orcid.org/0000-0003-3651-0232>
Jennifer Ngadiuba  <https://orcid.org/0000-0002-0055-2935>
Maurizio Pierini  <https://orcid.org/0000-0003-1939-4268>

References

- [1] The LHC Study Group 1995 The large Hadron Collider, conceptual design *Technical Report* CERN/AC/95-05 (LHC) (Geneva)
- [2] Khachatryan V et al 2017 *J. Instrum.* **12** 01020
- [3] CMS Collaboration 2016 arXiv:1609.02366
- [4] Apollinari G, Brüning O, Nakamoto T and Rossi L 2017 arXiv:1705.08830
- [5] Albrecht J et al 2019 *Comput. Softw. Big Sci.* **3** 1–49
- [6] Butterworth J M, Davison A R, Rubin M and Salam G P 2008 *Phys. Rev. Lett.* **100** 242001
- [7] Skiba W and Tucker-Smith D 2007 *Phys. Rev. D* **75** 115010
- [8] Baumgart M, Leibovich A K, Mehen T and Rothstein I Z 2014 *J. High Energy Phys.* **2014** 173
- [9] Aad G et al 2015 *J. High Energy Phys.* **2015** 1–39
- [10] Adams D et al 2015 *Eur. Phys. J. C* **75** 409
- [11] Abdesselam A et al 2011 *Eur. Phys. J. C* **71** 1661
- [12] Altheimer A et al 2012 *J. Phys. G: Nucl. Part. Phys.* **39** 063001
- [13] Altheimer A et al 2014 *Eur. Phys. J. C* **74** 2792
- [14] Caruana R 1997 *Mach. Learn.* **28** 41–75
- [15] Sirunyan A M et al CMS 2020 *Comput. Softw. Big Sci.* **4** 10
- [16] Pol A A and Pierini M 2020 Jet single shot detection (available at: <https://zenodo.org/record/4883651>)
- [17] Liu Z, Mao H, Wu C Y, Feichtenhofer C, Darrell T and Xie S 2022 arXiv:2201.03545
- [18] Hendrycks D and Gimpel K 2016 arXiv:1606.08415
- [19] Plehn T, Spannowsky M, Takeuchi M and Zerwas D 2010 *J. High Energy Phys.* **2010** 1–20
- [20] Larkoski A J, Marzani S, Soyez G and Thaler J 2014 *J. High Energy Phys.* **2014** 146
- [21] Thaler J and Van Tilburg K 2011 *J. High Energy Phys.* **2011** 15
- [22] Larkoski A J, Salam G P and Thaler J 2013 *J. High Energy Phys.* **2013** 108
- [23] Krohn D, Thaler J and Wang L-T 2010 *J. High Energy Phys.* **2010** 84

- [24] Ellis S D, Vermilion C K and Walsh J R 2010 *Phys. Rev. D* **81** 094023
- [25] Dasgupta M, Fregoso A, Marzani S and Salam G P 2013 *J. High Energy Phys.* **2013** 29
- [26] Dasgupta M, Fregoso A, Marzani S and Powling A 2013 *Eur. Phys. J. C* **73** 1–32
- [27] Dasgupta M, Powling A and Siodmok A 2015 *J. High Energy Phys.* **2015** 1–54
- [28] Cogan J, Kagan M, Strauss E and Schwartzman A 2015 *J. High Energy Phys.* **2015** 118
- [29] Almeida L G, Backović M, Cliche M, Lee S J and Perelstein M 2015 *J. High Energy Phys.* **2015** 1–21
- [30] Baldi P, Bauer K, Eng C, Sadowski P and Whiteson D 2016 *Phys. Rev. D* **93** 094034
- [31] de Oliveira L, Kagan M, Mackey L, Nachman B and Schwartzman A 2016 *J. High Energy Phys.* **2016** 1–32
- [32] Guest D, Collado J, Baldi P, Hsu S-C, Urban G and Whiteson D 2016 *Phys. Rev. D* **94** 112002
- [33] de Oliveira L, Paganini M and Nachman B 2017 *Comput. Softw. Big Sci.* **1** 1–24
- [34] Pearkes J, Fedorko W, Lister A and Gay C 2017 arXiv:1704.02124
- [35] Kasieczka G, Plehn T, Russell M and Schell T 2017 *J. High Energy Phys.* **2017** 6
- [36] Komiske P T, Metodiev E M and Schwartz M D 2017 *J. High Energy Phys.* **2017** 110
- [37] Barnard J, Dawe E N, Dolan M J and Rajcic N 2017 *Phys. Rev. D* **95** 014018
- [38] Macaluso S and Shih D 2018 *J. High Energy Phys.* **2018** 121
- [39] Butter A, Kasieczka G, Plehn T and Russell M 2018 *SciPost Phys.* **5** 028
- [40] Lan S-Q 2018 *Adv. High Energy Phys.* **2018** 4350287
- [41] Kasieczka G et al 2019 *SciPost Phys.* **7** 014
- [42] Bhimji W et al 2018 *J. Phys.: Conf. Ser.* **1085** 042034
- [43] Nguyen T Q, Weitekamp D, Anderson D, Castello R, Cerri O, Pierini M, Spiropulu M and Vlimant J-R 2019 *Comput. Softw. Big Sci.* **3** 1–14
- [44] Andrews M, Paulini M, Gleyzer S and Poczos B 2020 *Comput. Softw. Big Sci.* **4** 6
- [45] Andrews M, Alison J, An S, Burkle B, Gleyzer S, Narain M, Paulini M, Poczos B and Usai E 2020 *Nucl. Instrum. Methods Phys. Res. A* **977** 164304
- [46] Zhang K, Zhang Z, Li Z and Qiao Y 2016 *IEEE Signal Process. Lett.* **23** 1499–503
- [47] Zhang L, Lin L, Liang X and He K 2016 Is faster R-CNN doing well for pedestrian detection? *European Conf. on Computer Vision* (Springer) pp 443–57
- [48] Zou Z, Shi Z, Guo Y and Ye J 2019 arXiv:1905.05055
- [49] Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X and Pietikäinen M 2020 *Int. J. Comput. Vis.* **128** 261–318
- [50] Redmon J, Divvala S, Girshick R and Farhadi A 2016 You only look once: unified, real-time object detection 2016 *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE) pp 779–88
- [51] Redmon J and Farhadi A 2017 YOLO9000: better, faster, stronger 2017 *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE) pp 6517–25
- [52] Lin T Y, Goyal P, Girshick R, He K and Dollar P 2017 Focal loss for dense object detection 2017 *IEEE Int. Conf. on Computer Vision (ICCV)* (IEEE) pp 2999–3007
- [53] Fu C Y, Liu W, Ranga A, Tyagi A and Berg A C 2017 arXiv:1701.06659
- [54] Zhou X, Wang D and Krähenbühl P 2019 arXiv:1904.07850
- [55] Girshick R, Donahue J, Darrell T and Malik J 2014 Rich feature hierarchies for accurate object detection and semantic segmentation 2014 *IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE) pp 580–7
- [56] Ren S, He K, Girshick R B and Sun J 2015 Faster R-CNN: towards real-time object detection with region proposal networks *Annual Conf. on Neural Information Processing Systems 2015 (Montreal, Quebec, Canada, 7–12 December 2015) (Advances in Neural Information Processing Systems)* eds C Cortes, N D Lawrence, D D Lee, M Sugiyama and R Garnett vol 28 pp 91–99 (available at: <https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html>)
- [57] Girshick R 2015 Fast R-CNN 2015 *IEEE Int. Conf. on Computer Vision (ICCV)* (IEEE) pp 1440–8
- [58] Dai J, Li Y, He K and Sun J 2016 R-FCN: object detection via region-based fully convolutional networks *Annual Conf. on Neural Information Processing Systems 2016 (Barcelona, Spain, 5–10 December 2016) (Advances in Neural Information Processing Systems)* eds D D Lee, M Sugiyama, U von Luxburg, I Guyon and R Garnett vol 29 pp 379–87 (available at: <https://proceedings.neurips.cc/paper/2016/hash/577ef1154f3240ad5b9b413aa7346a1e-Abstract.html>)
- [59] Xu H, Lv X, Wang X, Ren Z, Bodla N and Chellappa R 2018 Deep regionlets for object detection *Proc. European Conf. on Computer Vision (ECCV)* pp 798–814
- [60] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C Y and Berg A C 2016 SSD: single shot multibox detector *European Conf. on Computer Vision* (Springer) pp 21–37
- [61] Felzenszwalb P F, Girshick R B, McAllester D and Ramanan D 2010 *IEEE Trans. Pattern Anal. Mach. Intell.* **32** 1627–45
- [62] Simonyan K and Zisserman A 2015 Very deep convolutional networks for large-scale image recognition (arXiv:1409.1556)
- [63] Jetley S, Lord N A, Lee N and Torr P H S 2018 Learn to pay attention 6th *Int. Conf. on Learning Representations, ICLR 2018 (Vancouver, BC, Canada, 30 April–3 May 2018)* (OpenReview.net) (available at: <https://openreview.net/forum?id=HyzbhfWRW>)
- [64] Anderson P, He X, Buehler C, Teney D, Johnson M, Gould S and Zhang L 2018 Bottom-up and top-down attention for image captioning and visual question answering 2018 *IEEE/CVF Conf. on Computer Vision and Pattern Recognition* (IEEE) pp 6077–86
- [65] Oktay O et al 2018 arXiv:1804.03999
- [66] Li Y, Chen Y, Wang N and Zhang Z X 2019 Scale-aware trident networks for object detection 2019 *IEEE/CVF Int. Conf. on Computer Vision (ICCV)* (IEEE) pp 6053–62
- [67] Yi J, Wu P and Metaxas D N 2019 *Comput. Vis. Image Underst.* **189** 102827
- [68] Woo S, Park J, Lee J Y and Kweon I S 2018 Cbam: Convolutional block attention module *Proc. European Conf. on Computer Vision (ECCV)* pp 3–19
- [69] Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z and Lu H 2019 Dual attention network for scene segmentation 2019 *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE) pp 3146–54
- [70] Hu J, Shen L and Sun G 2018 Squeeze-and-excitation networks 2018 *IEEE/CVF Conf. on Computer Vision and Pattern Recognition* (IEEE) pp 7132–41
- [71] Chen Y, Kalantidis Y, Li J, Yan S and Feng J 2018 A2-nets: double attention networks *Annual Conf. on Neural Information Processing Systems 2018, NeurIPS 2018 (Montréal, Canada, 3–8 December 2018) (Advances in Neural Information Processing Systems)* ed S Bengio, H M Wallach, H Larochelle, K Grauman, N Cesa-Bianchi and R Garnett vol 31 pp 350–9 (available at: <https://proceedings.neurips.cc/paper/2018/hash/e165421110ba03099a1c0393373c5b43-Abstract.html>)
- [72] Zhou D and Bie Ju L 2021 Deep convolutional neural networks (arXiv:1910.03151)

- [73] Han S, Mao H and Dally W J 2015 arXiv:1510.00149
- [74] Cheng Y, Wang D, Zhou P and Zhang T 2018 *IEEE Signal Process. Mag.* **35** 126–36
- [75] LeCun Y, Denker J and Solla S 1989 Optimal brain damage *Advances in Neural Information Processing Systems (Denver, Colorado, USA, 27–30 November 1989)* vol 2 pp 598–605
- [76] Louizos C, Welling M and Kingma D P 2018 Learning sparse neural networks through L_0 regularization *6th Int. Conf. on Learning Representations, ICLR 2018 (Vancouver, BC, Canada, 30 April–3 May 2018)* (OpenReview.net) (available at: <https://openreview.net/forum?id=H1Y8hhg0b>)
- [77] Gordon A, Eban E, Nachum O, Chen B, Wu H, Yang T J and Choi E 2018 MorphNet: fast and simple resource-constrained structure learning of deep networks *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (IEEE)* pp 1586–95
- [78] Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M and Adam H 2017 arXiv:1704.04861
- [79] Iandola F N, Han S, Moskewicz M W, Ashraf K, Dally W J and Keutzer K 2016 arXiv:1602.07360
- [80] Cohen T and Welling M 2016 Group equivariant convolutional networks *Proc. 33rd Int. Conf. on Machine Learning, ICML 2016 (New York City, NY, USA, 19–24 June 2016) (JMLR Workshop and Conf. Proc.)* vol 48, ed M Balcan and K Q Weinberger (JMLR.org) pp 2990–9 (available at: <http://proceedings.mlr.press/v48/cohen16.html>)
- [81] Courbariaux M, Bengio Y and David J 2015 BinaryConnect: training deep neural networks with binary weights during propagations *Annual Conf. on Neural Information Processing Systems 2015 (Montreal, Quebec, Canada, 7–12 December 2015) (Advances in Neural Information Processing Systems)* ed C Cortes, N D Lawrence, D D Lee, M Sugiyama and R Garnett vol 28 pp 3123–31 (available at: <https://proceedings.neurips.cc/paper/2015/hash/3e15cc11f979ed25912dff5b0669f2cd-Abstract.html>)
- [82] Courbariaux M, Hubara I, Soudry D, El-Yaniv R and Bengio Y 2016 arXiv:1602.02830
- [83] Zhou S, Wu Y, Ni Z, Zhou X, Wen H and Zou Y 2016 arXiv:1606.06160
- [84] Rastegari M, Ordonez V, Redmon J and Farhadi A 2016 XNOR-Net: imagenet classification using binary convolutional neural networks *European Conf. on Computer Vision (Berlin: Springer)* pp 525–42
- [85] Hubara I, Courbariaux M, Soudry D, El-Yaniv R and Bengio Y 2017 *J. Mach. Learn. Res.* **18** 6869–98
- [86] Zhu C, Han S, Mao H and Dally W J 2017 Trained ternary quantization *5th Int. Conf. on Learning Representations, ICLR 2017 (Toulon, France, 24–26 April 2017)* (OpenReview.net) (available at: https://openreview.net/forum?id=S1_pAu9xl)
- [87] Lee E H, Miyashita D, Chai E, Murmann B and Wong S S 2017 LogNet: energy-efficient neural networks using logarithmic computation *2017 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP) (IEEE)* pp 5900–4
- [88] Cai Z, He X, Sun J and Vasconcelos N 2017 Deep learning with low precision by half-wave Gaussian quantization *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (IEEE)* pp 5406–14
- [89] Li F, Zhang B and Liu B 2016 arXiv:1605.04711
- [90] Pol A A et al 2021 *EPJ Web Conf.* **251** 04027
- [91] Ioffe S and Szegedy C 2015 Batch normalization: accelerating deep network training by reducing internal covariate shift *Proc. 32nd Int. Conf. on Machine Learning, ICML 2015 (Lille, France, 6–11 July 2015) (JMLR Workshop and Conf. Proc.)* eds F R Bach and D M Blei (JMLR.org) vol 37 pp 448–56 (available at: <http://proceedings.mlr.press/v37/ioffe15.html>)
- [92] Sari E, Belbahri M and Nia V P 2020 How does batch normalization help binary training? (arXiv:1909.09139)
- [93] He K, Zhang X, Ren S and Sun J 2015 Delving deep into rectifiers: surpassing human-level performance on ImageNet classification *2015 IEEE Int. Conf. on Computer Vision (ICCV) (IEEE)* pp 1026–34
- [94] Ghiasi G, Lin T and Le Q V 2018 DropBlock: a regularization method for convolutional networks *Annual Conf. on Neural Information Processing Systems 2018, NeurIPS 2018 (3–8 December, 2018)* eds S Bengio, H M Wallach, H Larochelle, K Grauman, N Cesa-Bianchi and R Garnett pp 10750–60 (available at: <https://proceedings.neurips.cc/paper/2018/hash/7edcfb2d8f6a659ef4cd1e6c9b6d7079-Abstract.html>)
- [95] Han S, Pool J, Tran J and Dally W J 2015 arXiv:1506.02626
- [96] Liu Z, Li J, Shen Z, Huang G, Yan S and Zhang C 2017 Learning efficient convolutional networks through network slimming *2017 IEEE Int. Conf. on Computer Vision (ICCV) (IEEE)* pp 2755–63
- [97] Sjöstrand T, Mrenna S and Skands P 2008 *Comput. Phys. Commun.* **178** 852–67
- [98] De Favereau J, Delaere C, Demin P, Giammanco A, Lemaître V, Mertens A and Selvaggi M 2014 *J. High Energy Phys.* **2014** 57
- [99] CMS Collaboration 2008 *J. Instrum.* **3** S08004
- [100] Sirunyan A M et al 2017 *J. Instrum.* **12** 10003
- [101] Aaboud M et al ATLAS 2017 *Eur. Phys. J. C* **77** 466
- [102] Paszke A et al 2019 PyTorch: an imperative style, high-performance deep learning library *Annual Conf. on Neural Information Processing Systems 2019, NeurIPS 2019 (Vancouver, BC, Canada, 8–14 December 2019) (Advances in Neural Information Processing Systems)* ed H M Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E B Fox and R Garnett vol 32 pp 8024–35 (available at: <https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html>)
- [103] Szegedy C, Vanhoucke V, Ioffe S, Shlens J and Wojna Z 2016 Rethinking the inception architecture for computer vision *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 2818–26
- [104] Deng J, Dong W, Socher R, Li L, Li K and Li F 2009 ImageNet: a large-scale hierarchical image database *2009 IEEE Conf. on Computer Vision and Pattern Recognition (IEEE)* pp 248–55
- [105] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. Thirteenth Int. Conf. on Artificial Intelligence and Statistics (Proc. of Machine Learning Research) (Chia Laguna Resort, Sardinia, Italy)* vol 9, ed Y W Teh and M Titterton (JMLR Workshop and Conf. Proc.) pp 249–56
- [106] Zhang H, Cissé M, Dauphin Y N and Lopez-Paz D 2018 Mixup: beyond empirical risk minimization *6th Int. Conf. on Learning Representations, ICLR 2018 (Vancouver, BC, Canada, 30 April–3 May 2018)* (OpenReview.net) (available at: <https://openreview.net/forum?id=r1Ddp1-Rb>)
- [107] Bochkovskiy A, Wang C Y and Liao H Y M 2020 arXiv:abs/2004.10934
- [108] Larkoski A J, Marzani S, Soyez G and Thaler J 2014 *J. High Energy Phys.* **2014** 146
- [109] Thaler J and Van Tilburg K 2011 *J. High Energy Phys.* **2011** 015
- [110] Krupa J et al 2021 *Mach. Learn.: Sci. Technol.* **2** 035005
- [111] Bocci A, Innocente V, Kortelainen M, Pantaleo F and Rovere M 2020 *Front. Big Data* **3** 601728
- [112] Rovere M, Chen Z, Di Pilato A, Pantaleo F and Seez C 2020 *Front. Big Data* **3** 591315
- [113] Qasim S R, Long K, Kieseler J, Pierini M, Nawaz R, Pierini M, Nawaz R and CMS 2021 *EPJ Web Conf.* **251** 03072
- [114] Pata J, Duarte J, Vlimant J-R, Pierini M and Spiropulu M 2021 *Eur. Phys. J. C* **81** 381