IL NUOVO CIMENTO **42 C** (2019) 196
DOI 10.1393/ncc/i2019-19196-4

Colloquia: IFAE 2018

Convolutional neural network for track seed filtering at the CMS HLT

A. Di Florio(1)(2)

(1) *Dipartimento Interateneo di Fisica di Bari, Università di Bari - Bari, Italy*
(2) *INFN, Sezione di Bari - Bari, Italy*

received 31 January 2019

Summary. — Starting with Run III, future development projects for the Large Hadron Collider will constantly bring nominal luminosity increase, with the ultimate goal of reaching a peak luminosity of $5 \cdot 10^{34}$ cm^{-2} s^{-1} for ATLAS and CMS experiments. This rise in luminosity will result in an increased number of simultaneous proton collisions (pileup), up to 200, that will pose new challenges for the CMS detector and, specifically, for track reconstruction in the Silicon Pixel Tracker. One of the first steps of the track finding workflow is the creation of track seeds, *i.e.*, compatible pairs of hits, that are subsequently fed to higher-level pattern recognition steps. However, the set of compatible hit pairs is highly affected by combinatorial background. A possible way of reducing this effect is taking into account the shape of the hit pixel cluster to check the compatibility between two hits. To each doublet is attached a collection of two images built with the ADC levels of the pixels forming the hit cluster. Thus, the task of fake rejection can be seen as an image classification problem for which *Convolutional Neural Networks* (CNNs) have been widely proven to provide reliable results. In this work we present our studies on CNNs applications to the filtering of track pixel seeds.

1. – Track reconstruction at the CMS HLT

The Compact Muon Solenoid (CMS) [1] is a general-purpose detector designed for the precision measurement of leptons, photons, and jets, among other physics objects, in proton-proton as well as heavy-ion collisions at the CERN LHC [2]. The LHC is designed to collide protons at a center-of-mass energy of 14 TeV and a luminosity of 10^{34} cm^{-2} s^{-1}. At design luminosity, the pp interaction rate is about 1 GHz but only a small fraction of these collisions contains events that can be interesting for CMS physics analyses and can be stored to be accessible offline. The trigger system is devoted to the selection of these events from the totality of the inelastic collision events. In order to accomplish this task, the CMS trigger system utilizes two levels [3]. The first one (L1) is implemented in custom hardware and restricts the output rate to 100 kHz. Events are then passed to the High-Level Trigger (HLT) that further refines the purity of the physics objects, and selects an average rate of 400 Hz for final offline storage.

The HLT hardware consists of a processor farm and gathers information from all the CMS detectors. One of the first steps for the event selection at HLT is the track reconstruction that has to fulfill stringent CPU timing constraints: the event selection has to be fulfilled with an average latency smaller than 260 ms. With the instantaneous luminosity increasing up to the ultimate goal of $5 \cdot 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ for the High Luminosity LHC (HL-LHC) upgrade, this time constraint is rapidly approaching. Beside this, the number of simultaneous proton collisions per bunch crossing ("pileup") will continue to increase: in 2018 the highest pileup reached averaged over a data run exceeded 50. The track reconstruction procedure begins with the construction of seeds (seeding), *i.e.*, few (2–4) pixel hits that act as the building blocks for track candidates and define the initial estimate for trajectory parameters.

The very first step of seeding consists in collecting and mapping all the valid hits on the Silicon Pixel Detector layer. The Run 2 updated Silicon Pixel Tracker is arranged in four barrel layers (BPix) and three forward disks (FPix) in each endcap region. Then, iterating on all the couples of consecutive *seeding layers*, the algorithm selects and collects all the compatible hit pairs (*doublets*). The compatibility between two hits is evaluated only on the basis of geometrical considerations, such as cuts in η, ϕ and r. The doublet production is thus highly dominated by combinatorial background and constitutes a bottleneck for the subsequent steps. Therefore, the reduction of the track reconstruction time hinges on the rejection of seeding combinatorial background. In the next sections a method for filtering doublet seeds based on CNNs is proposed.

2. – Convolutional neural networks for doublet seeds filtering

A traditional deep neural network is a machine learning model whose goal is to approximate a function $\hat{f}$ by composing a sequence of simpler functions. It consists of an input layer, one or more hidden layers and an output layer [4]. To a first approximation, each layer acts on the input as a composition of a matrix multiplication and an activation function. Convolutional neural networks [5] are a specialized kind of neural network for processing data that has a grid-like structure, such as 2D images. The building block of a CNN is a layer that uses *discrete convolution* in place of general matrix multiplication [4]. A convolutional layer takes in input a $n \times m \times r$ two-dimensional image, where r is the number of channels or depth. The output of one or more stacked convolutional layers is usually passed to pooling layers that perform dimensionality reduction. A $p \times p'$ *pooling layer* replaces a $p \times p'$ region of the image with a single value, *e.g.*, the maximum value in that region for a max pooling layer.

As described above, while building seeds, the compatibility between two hits is evaluated only on the basis of geometrical considerations. A possible way of reducing the doublet fake ratio is taking into account that each hit is actually a cluster of pixels with its own shape. Each pixel is characterized by its 16-bit A.D.C. level ($\max_{\mathrm{ADC}} = 2^{16} - 1$) and its local position (x, y) on the layer. For each hit a 15×15 squared matrix M is built with the pixel local x on the rows and the local y on the columns. The matrix center is matched with the hit center of charge and each element m_{ij} is set to the A.D.C. level of the corresponding (x_i, y_j) pixel. Those pixels that stride over the hit cluster

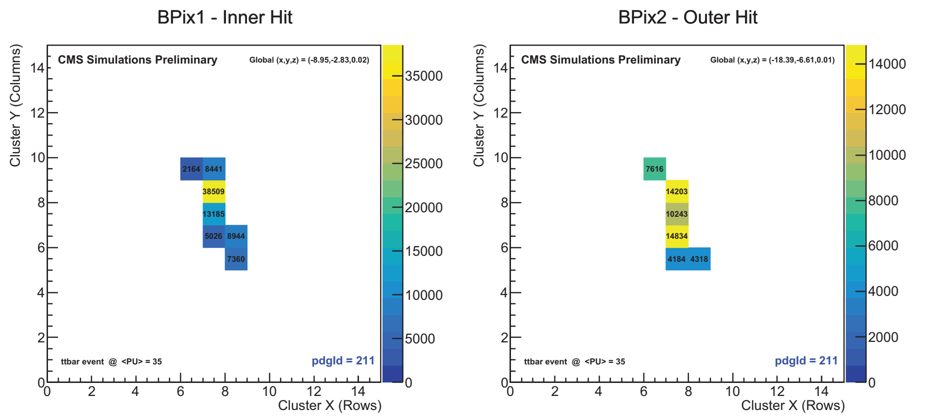

Fig. 1. – Example of a BPix1-BPix2 true doublet generated from $t\bar{t}$ simulated events corresponding to a π^+ track.

boundaries are set to zero (zero-padding technique). With this procedure each doublet can be considered as a collection of two 15×15 matrices, see fig. 1. Thus, the rejection of fake doublets is reduced to an image/pattern recognition task, perfectly suitable for being dealt with CNNs.

To test the feasibility of this kind of approach, the generation (via PYTHIA 8 [6]) and the reconstruction of $t\bar{t}$ events at energy of the center of mass of $\sqrt{s} = 13\,\text{TeV}$, with average pileup $\langle PU \rangle = 35$ and bunch time spacing of 25 ns has been simulated within the CMS software framework (CMSSW [7]). A doublet is labeled as *true* only if it is formed by pixel hits belonging to the same tracking particle. An $O(10^5)$ doublet is produced per each event and the ratio between *true* and *fake* doublets is an $O(10^2)$. For each doublet 537 parameters are stored: 225 + 225 *pixels* for the inner and the outer hit; 63 *doublet features* defined for each doublet and that include hit coordinates and further cluster characteristics; 24 *track labels* defined only for matched doublets, *e.g.*, the corresponding particle p_T and η.

On the whole, 1000 events are simulated and split into training, test and validation dataset. The training and the validation set are balanced so that the ratio between

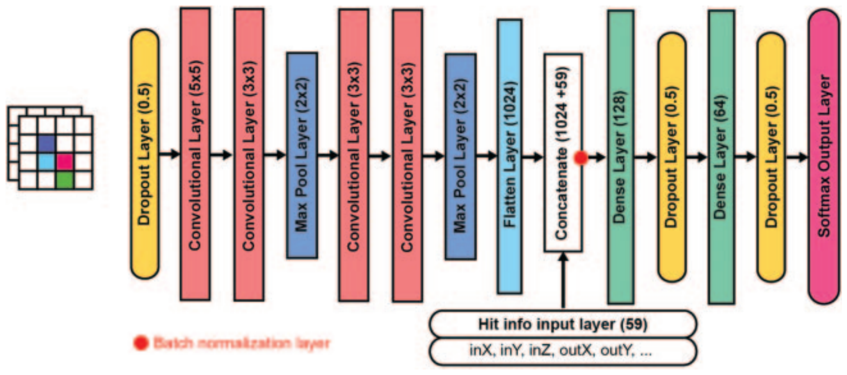

Fig. 2. – Layer map model architecture.

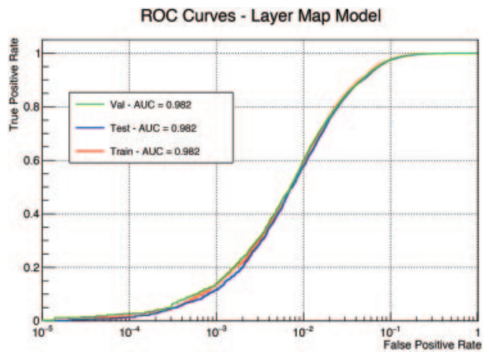

Fig. 3. – ROC curves for the train, test and validation dataset.

fake and true doublets is one. The whole balanced training dataset is composed of approximately 2.5 millions doublets.

3. – CNN classifier: The layer map model

The architecture for the doublet filtering classifier (*layer map model*) is shown in fig. 2 and it concatenates:

1) *CNN block*: conventional stack of convolutional layers (4) with 5×5 or 3×3 filters and max pooling layers (2) whose output is reduced to a one-dimensional structure through a flattening layer that returns a 1024-elements vector.

2) *Dense block*: stack of two fully connected layers that are fed with the one-dimension reduced images from the previous block and 59 further doublets information (such as hits' detectors and coordinates).

The two output neurons, coming from a *softmax* layer, return the complementary probabilities that a doublet is fake or true ($p_{true} + p_{fake} = 1.0$). The 15×15 doublets pads, before being fed to the first block, are pre-processed as follows. First, each pixel content is standardized with the mean and the standard deviation pre-computed on the whole 2.5 millions doublet dataset: $\mu = 13382.00$, $\sigma = 10525.13$. Then, the input is split into 20 *detector channels*, 10 for the inner hit and 10 for the outer hit. Every channel corresponds to a single layer of the detector (4 barrels and 6 endcaps) and is initialized to a 15×15 zero matrix. Then only the two channels corresponding to the inner and the outer layer are set, respectively, to the inner hit and the outer hit cluster matrix. This approach allows us to separate the hit shape clusters based on the specific layer and to apply a different transformation.

Table I. – *Layer map network scores for train, validation and test dataset.*

	AUC	Acc	Rej at Eff = 0.99	Eff at Rej = 0.5
Train	0.982	0.938	0.854	0.9997
Test	0.981	0.940	0.853	0.9996
Val	0.980	0.939	0.852	0.9996

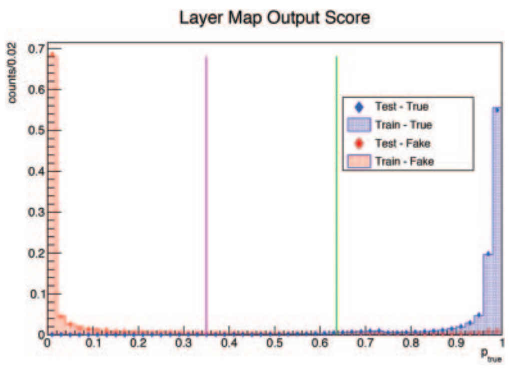

Fig. 4. – Network score for true (blue) and fake (red) doublets for train (filled histogram) and test dataset (diamond markers). In purple the 0.94 accuracy threshold. In green the 0.99 efficiency threshold.

The model has been developed and trained with the `Keras` package [8], a high-level neural networks API, running on the top of `TensorFlow` library [9]. The network has been trained on NVIDIA Tesla K20 nodes at Bari Physics Department Tier 2 (ReCaS) [10] and on NVIDIA GTX1080 Ti GPUs at CERN.

4. – Model testing and results

Once tested and tuned on a smaller sample, the model has been trained on the whole dataset (2.5M doublets) with a *categorical cross-entropy* loss function [4], using *Adam* [11] optimizer and accuracy as evaluation metric and with an *early stopping* callback, to avoid over-fitting. The training took about 5 days on an NVIDIA Tesla K20 node.

The ROC curves for validation, test and training dataset, shown in fig. 3, completely overlay each other, and the area under the curves (AUCs) is more than 0.98. While assuring a 0.99 efficiency (true positive rate), the network's sensitivity (true negative rate) th reaches 0.85. The highest accuracy reached is about 0.94 for all the three datasets. See table I for further network performance results. The normalized output score, namely the network estimated probability that a doublet is true (p_{true}) shows optimal separation between fake and true doublet samples. Both train and test p_{true} distributions are plotted in fig. 4 together with the cut for an efficiency of 0.99. In order to compare them, a two-sided Kolmogorov-Smirnov test has been performed. This tests whether two samples are drawn from the same distribution [12]. For both true and fake histograms, the resulting score is $KS \approx 0.070$ corresponding to a p-value $p_{val} \approx 0.961$, that assures us that the two histograms come from the same distribution with a very high level of confidence.

5. – Conclusions and acknowledgments

In conclusion, the results described show that CNN techniques for mitigating combinatorial explosion look very promising and need to be further explored. Ongoing work includes the verification of the effect on the downstream track reconstruction, the exploration of different hardware architectures for fast inference and the final integration in the CMS reconstruction framework.

REFERENCES

[1] CMS Collaboration, *JINST*, **3** (2008) S08004.
[2] Evans L. and Bryant P., *JINST*, **3** (2008) S08001.
[3] CMS Collaboration, *JINST*, **12** (2017) P01020.
[4] Goodfellow I., Bengio Y. and Courville A., *Deep Learning* (The MIT Press) 2016.
[5] LeCun Y. *et al.*, *IEEE Commun. Mag.*, **27** (1989) 4146.
[6] Sjstrand T., Mrenna S. and Skands P., *JHEP*, **05** (2006) 026; *Comput. Phys. Commun.*, **178** (2008) 852.
[7] `https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCMSSWFramework`.
[8] Chollet François *et al.*, Keras (2015) GitHub, `https://github.com/fchollet/keras`.
[9] TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, `http://tensorflow.org`.
[10] ReCas is a project financed by the italian MIUR (*PONa3_00052, Avviso 254/Ric.*); its web page is `http://www.recas-bari.it/index.php/en/`.
[11] Kingma D. P. and Ba J., *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980.
[12] Eadie W. T. *et al.*, *Statistical Methods in Experimental Physics* (North-Holland, Amsterdam) 1971.