Optimizing electron and photon reconstruction using deep learning: application to the CMS electromagnetic calorimeter (ECAL)

Davide Valsecchi[1] **for the CMS Collaboration**

[1] ETH Zurich

E-mail: davide.valsecchi@cern.ch

Abstract. The reconstruction of electrons and photons in CMS depends on topological clustering of the energy deposited by an incident particle in different crystals of the electromagnetic calorimeter (ECAL). These clusters are formed by aggregating neighbouring crystals according to the expected topology of an electromagnetic shower in the ECAL. The presence of upstream material (beampipe, tracker, and support structures) causes electrons and photons to start showering before reaching the calorimeter. This effect, combined with the 3.8 T CMS magnetic field, leads to energy being spread in several clusters around the primary one. It is essential to recover the energy contained in these satellite clusters in order to achieve the best possible energy resolution for physics analyses. Historically, satellite clusters have been associated to the primary cluster using a purely topological algorithm which does not attempt to remove spurious energy deposits from additional pileup interactions (PU). The performance of this algorithm is expected to degrade during LHC Run 3 (started in 2022) because of the larger average PU and the increasing levels of noise due to the ageing of the ECAL detector. New methods are being investigated, which exploit state-of-the-art deep learning architectures like Graph Neural Networks (GNN). These more sophisticated models improve the energy collection and are more resilient to PU and noise. This contribution covers the results of the model optimization, and the steps to deploy it in the CMS reconstruction sequence. The inference performance is analyzed and strategies to improve it are described in this manuscript.

1. Introduction

The CMS [1] electromagnetic calorimeter (ECAL) [2] is a homogeneous calorimeter made of 75848 lead tungstate ($PbWO_4$) scintillating crystals, located inside the CMS superconducting solenoid magnet. It is made of a barrel part (EB) covering the region of pseudorapidity $|\eta| < 1.48$ with 61200 crystals and two endcaps (EE), which extend the coverage up to $|\eta| < 3.0$ with 7324 crystals each. Scintillation light is detected with avalanche photodiodes (APD) in the barrel and vacuum phototriodes (VPT) in the endcaps. The ECAL is crucial for the identification and reconstruction of photons and electrons, and the measurement of jets and of missing transverse momentum. The electrons and photons are typically reconstructed up to $|\eta| < 2.5$, the region covered by the tracker, while jets are reconstructed up to $|\eta| < 3.0$.

Several algorithms are stacked on top of each other to reconstruct electrons and photons candidates from the measurement of scintillation light in each single crystal in the ECAL detector [3]. A single electron or photon usually leaves more than one cluster of energy in the ECAL detector. The electron, bending in the strong magnetic field of the CMS solenoid (3.8 T)

while passing through the Pixel and Tracker detectors, emits bremsstrahlung photons that will leave a trace of small energy clusters in the ECAL detector near the main impact point, mostly extended in the transverse $R - \phi$ plane. The photon, instead, is converted to electron-positron pairs interacting with the several layers of the inner detectors of CMS, thus also depositing multiple clusters of energy in ECAL. Therefore, in order to improve the energy resolution for electrons and photons, an additional clustering algorithm is employed, which includes the energy of secondary clusters to form SuperClusters (SC).

2. Effect on the reconstruction performance

The effect of the new algorithm, called DeepSC, on the electron and photon objects reconstructed by CMS has been studied in detail using MC simulation [4, 5]. An energy resolution study [6] is performed after retraining the dedicated electron and photon energy corrections, both for the new algorithm and for the legacy one, called Mustache. The effect is analyzed using electrons and photons candidates reconstructed by the Particle Flow algorithm [7], which includes the information from the tracker. The energy resolution is computed by fitting the ratio between the calibrated energy, extracted with the reconstruction algorithm, and the generator level particle energy. A Cruijiff function is employed as the fit model.

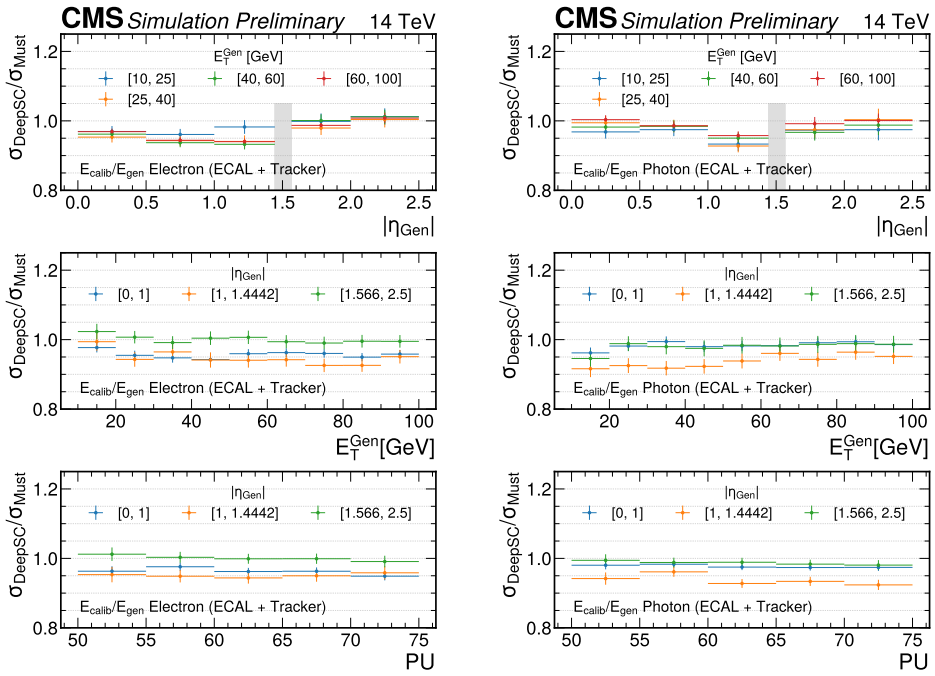

Figure 1: Improvement in the energy resolution for electrons (left) and photons (right) from the DeepSC algorithm with respect to the Mustache one ($\sigma_{DeepSC}/\sigma_{Must}$) at the final reconstruction level. Electrons on the left, photons on the right. [6]

The tracker information is crucial to improve the energy resolution of electrons at low energy, but the new algorithm brings an improvement also at higher energies where the ECAL contribution is dominant, specially in the $1.0 < |\eta| < 1.5$ region. Photons, which do not have information from the tracker, are consistenly improved. The DeepSC algorithm makes the resolution flatter versus the PU, also at the global event description level.

3. Implementation in the CMS software

The DeepSC algorithm has been implemented in the CMS reconstruction sequence replacing transparently the legacy one. The evaluation proceeds in steps, separately for each event:

3.1. Window preparation

- detector windows are built around each cluster with transverse energy $E_T > 1$ GeV (upper left panel in Fig. 2), called seed.
- the seed is connected to all the clusters in each window to form a graph
- Input features are computed for each cluster from basic quantities. The position and energy of each crystal forming the clusters is also used.

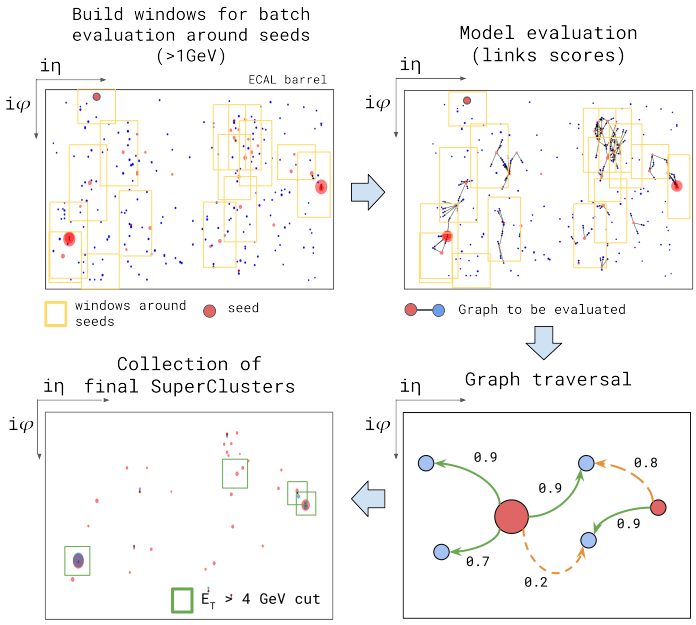

Figure 2: Schema of the DeepSC algorithm evaluation steps in the CMS reconstruction software

3.2. Model evaluation

- The TensorFlow model is evaluated on batches of detector windows: the output is the probability of each edge between the seed and the nearby clusters.
- The complete graph is traversed, starting from the highest energy seed, to solve overimpositions and collect the final SC (bottom right in Fig. 2),
- a final cut of $E_T^{SC} > 4$ GeV is applied to remove SuperCluster formed mainly by detector noise and very low energy PU (bottom left in Fig. 2).

3.3. Evaluation dimensions

The number of clusters in each window, and their number of crystals (rechits) determine the amount of data evaluated by the model at each iteration. Figure 3 (left) shows their distribution in a sample of $\bar{t}t$ Monte Carlo (MC) events, simulated under LHC Run 3 conditions, with an average PU of 65. Figure 3 (center) shows the cumulative distribution of the number of windows, accumulated over all the events, with the number clusters and maximum number of rechits per window over a threshold. The model needs to be evaluated with a predefined zero-padded input size: the distribution of cluster and rechits falls exponentially, but it has a long tail of window with up to 60 clusters or 60 rechits. This poses a challenge for the inference time of the model in the CMS software.

4. Inference time optimization

4.1. Input tensors zero-padding size

CPU time can be saved by executing the inference with two models with different zero-padding size, one prepared for a small number of clusters and rechits input, and one for the most inclusive case (60 clusters, 60 rechits). A single model has to be trained, then it can be exported with different internal tensors dimensions. The smaller model, much faster CPU wise, is executed for most of the window in each event, wherease the larger model is needed only sparsingly. The average fraction of windows per event which have to be evaluated with the largest model is shown in the Figure 3 (right), for different thresholds of number of clusters or rechits. By

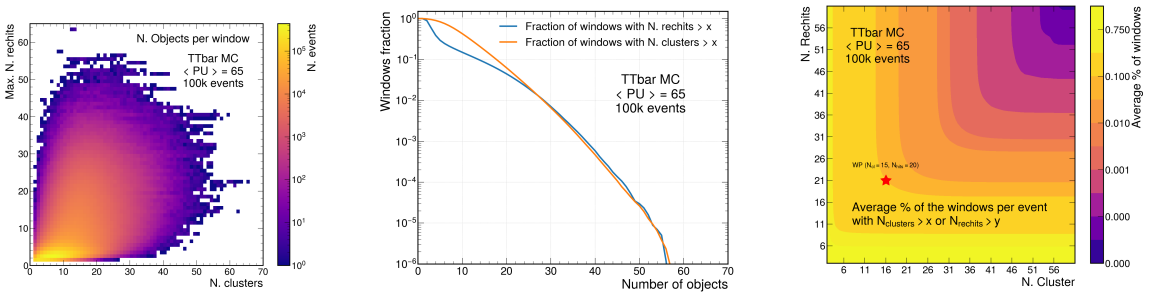

Figure 3: (Left) Number of clusters and maximum number of rechits in a detector window observed in a $\bar{t}t$ Monte Carlo sample. (Center) Cumulative distribution of the number of windows (accumulated over all the events) with more than x number of clusters or more than x maximum number of rechits. (Right) Average fraction windows per event with more than x clusters or x rechits.

choosing 15 cluster and 20 rechits as maximum dimension for the small model, on average only 10% of the windows in an event will be evaluated with the slower model.

4.2. Model architecture optimization

The architecture of the DeepSC model consists of three steps: encoding (or graph building), graph elaboration, and decoding. The rechits information are preprocessed with a dedicated convolution layer before concatenating their information to the clusters basic features. Ref.[5] describes the model architecture details. Different variations of the same network schema have been tested to improve the performance time, while keeping the physics performance unchanged:

- **Default**: initial best candidate $\sim 390k$ weights. Includes a Graph Convolution Network (GCN) on rechits and one on clusters, followed by self-attention layers.
- **Simpler rechits layer**: Removed GCN layer on rechits to improve the timing and slimmed the model. ($\sim 170k$ weights)
- **Without rechits layer**: Removed completely the rechits inputs and increased the capacity of the other layers (two versions, $\sim 165k$ and $\sim 370k$ weights).

4.3. Tensorflow inference time comparison

The inference time cost of the different models has been evaluated with the Tensorflow profiler on a single CPU thread process, to mimic the most common CMS computing environment, where the tensorflow internal threading is switched off. The models are profiled with small and large zero-padding dimensions, with a batch size of 32 windows. Figure 4 on the left shows the comparison of the inference time for the different models flavours described above.

The Graph Convolution Network on rechits is an expensive operation in the default model: a large input dimension makes the default model very slow. On the contrary, the model with simplified rechits layer is still fast also with a large input size. As expected, the model without rechits layer is the fastest.

4.4. Dynamic zero-padding reconstruction

The mechanims to dynamically choose which model is run during inference, depending on the window inputs size, has been implemented in the CMS software. Figure 4 on the right, shows the fraction of the time in the standard CMS reconstruction sequence used by the DeepSC algorithm

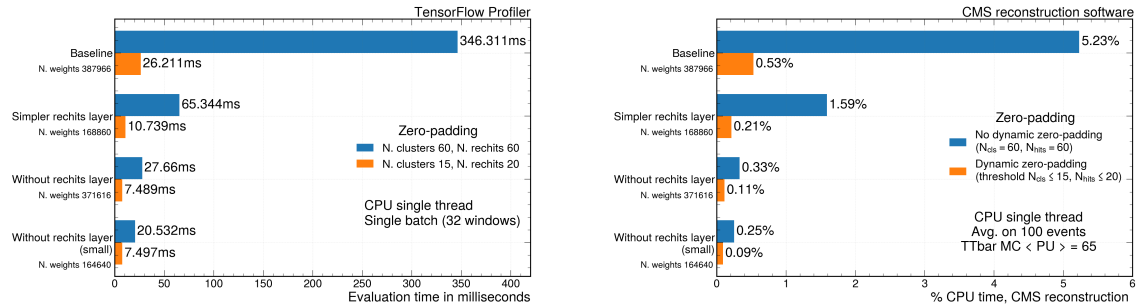

Figure 4: (Left) Inference time for different zero-padding dimensions and different model architectures in TensorFlow. (Right) Inference time of the DeepSC algorithm in the CMS reconstruction software: comparison between fixed or dynamic zero-padding strategy

with a fixed or dynamical zero-padding strategy on a $\bar{t}t$ MC sample. The plot shows in blue the fraction of CMS reconstruction time used if only a single model with the maximum zero-padding (60 clusters, 60 rechits) is always run, whereas in orange if the dynamic zero-padding strategy is activated, with a threshold to 15 clusters and 20 rechits for the small model [8]. The improvement in speed is very large, up to a factor of 10, for the default and simplified rechits layer models. For the models which do not use the rechits input, the speedup is smaller since they do not contain the expensive operations on the rechits collection of each cluster. The threshold to use can be further tuned to reach the optimal improvement.

5. Conclusion and next steps

The inference performance of the DeepSC model has been studied in a realistic scenario on $\bar{t}t$ Run3 MC. Different strategies to improve the timing have been explored, both on the side of the model architecture and on the inference mode one. The complexity of the model can be reduced with negligible physics performance changes (and with a proper hyper-parameters optimization) to strongly decrease the running time. A speedup of 10 times can be easily reached by implementing a dynamic zero-padded strategy with two models tailored for small and large windows.

References

[1] Chatrchyan S *et al.* (CMS) 2008 *JINST* **3** S08004

[2] 1997 CMS: The electromagnetic calorimeter Tech. Rep. CERN-LHCC-97-33, CMS-TDR-4 URL `https://cds.cern.ch/record/349375`

[3] Sirunyan A M *et al.* (CMS) 2021 *JINST* **16** P05014 (*Preprint* `2012.06888`)

[4] 2021 ECAL SuperClustering with Machine Learning Tech. Rep. CERN-CMS-DP-2021-032 URL `https://cds.cern.ch/record/2792321`

[5] Valsecchi D and for the Collaboration C 2023 *Journal of Physics: Conference Series* **2438** 012077 (*Preprint* `2204.10277`)

[6] 2022 ECAL SuperClustering with Machine Learning - Corrected energy performance Tech. Rep. CERN-CMS-DP-2022-032 URL `https://cds.cern.ch/record/2826227`

[7] Sirunyan A M *et al.* (CMS) 2017 *JINST* **12** P10003 (*Preprint* `1706.04965`)

[8] 2022 ECAL DeepSC: Optimization of the DeepSC model inference strategy for reconstruction speedup Tech. Rep. CERN-CMS-DP-2022-058 URL `https://cds.cern.ch/record/2841537`