Deep topology classifiers for a more efficient trigger selection at the LHC

Daniel Weitekamp III[1], Thong Q. Nguyen[2], Dustin Anderson[2], Roberto Castello[3], Maurizio Pierini[3], Maria Spiropulu[2], and Jean-Roch Vlimant[2]

[1]University of California at Berkeley, USA
[2]California Institute of Technology, USA
[3]CERN, Switzerland

Abstract

We show how a topology classification based on the list of reconstructed particles in an LHC collision event could be used to improve the online decision of keeping or rejecting an event (trigger). Collision data are represented as: (i) a sequence of particles, used to train recurrent networks; (ii) an abstract image encoding visually the relevant features of each particle and suitable for image-detection techniques. Using any of these classifiers as the last step of the trigger decision process, one could reduce by a factor ~ 10 the number of stored events while retaining between 95 and 99% of the interesting events, resulting in substantial savings in resources.

1 Introduction

The CERN Large Hadron Collider (LHC) collides protons every 25 ns. Each collision can result in any of thousands of physics processes. A few of these processes, e.g., QCD multijet production, constitute the vast majority of collisions at the LHC. Other events are rarer, such as W boson (W+jets) and top-pair production ($t\bar{t}$). Interesting events are selected in real time by a set of rule-based algorithms, known as the trigger system, which exploit the presence of rare features in the event. The selection rules are usually inclusive, i.e., more than one topology is triggered by the same requirement. A notable example is the requirement of an energetic electron or muon in the event, isolated from the rest of the particles (as a proxy for the decay of a W or Z boson). While effective in selecting interesting events (e.g., W+jets and $t\bar{t}$), these rule-based algorithms suffer from a not negligible false-positive rate (e.g., QCD multijet). This contamination at the trigger level translates into a waste of resources downstream (CPU for processing and disk for storage) on events that are later on rejected in early stages of the offline data analysis and prevents using the corresponding trigger bandwidth to select more interesting events.

In this paper, we investigate the possibility of filtering away false positives by looking at the event topology in the trigger system. This task is achieved using the list of particles (charged particles, photons, and neutral particles) reconstructed by a particle-flow [1] algorithm to train two different kinds of classifiers: (i) recurrent networks, taking as input a sequence of particles; (ii) a CNN classifier, based on an abstract representation of the reconstructed particles as an image.

2 Related work

Several studies have investigated the possibility of identifying the event topology from a limited list of high-level features, selecting according to the event topology of interest [2]. Recursive Neural Networks have been considered to define jet and event classifiers starting from a list of reconstructed particle momenta [3]. Recently, it was proposed to represent events as abstract images where reconstructed physics objects (jets, in that case) are represented as geometric shapes whose size reflects the energy of the particle [4], as a possible solution to image sparsity and irregularities in the detector geometry. We generalize this approach to work with the full list of particles.

3 Dataset

We consider proton-proton collision events similar to those produced at the LHC, with a center-of-mass energy of 13 TeV and no additional proton-proton interaction per beam crossing. Synthetic data corresponding to W+jets, $t\bar{t}$ and QCD are generated using the `PYTHIA8` [5] event generation package. The generated sample is processed with the `DELPHES` [6] package, which applies a parametric model of a detector response. The default `DELPHES` CMS card is used, running the particle-flow reconstruction algorithm distributed with `DELPHES`. The list of the reconstructed particle four-momenta is obtained, the reconstructed particles being classified as charged particles, photons, and neutral hadrons.

For each particle q, the following information is given: (i) The particle four-momentum in Cartesian coordinates (E, p_X, p_Y, p_Z); (ii) The particle three-momentum in cylindrical coordinates: the transverse momentum p_T, the azimuthal angle ϕ and the pseudorapidity η, defined as $-\log(\tan\frac{\theta}{2})$ (where θ is the polar angle); (iii) For the charged particles, the Cartesian coordinates (X, Y, Z) of the particle origin. For all other particles, zero padding is used to fill these entries; (iv) The electro-magnetic charge; (v) The particle isolation with respect to charged particles (`ChISO`), photons (`GammaISO`), or neutral particles (`NeuISO`). For each particle class, the isolation is quantified as $\texttt{ISO} = \frac{\Sigma_{p\neq q} p_T^p}{p_T^q}$, where the sum extends over all the particles of that class with distance $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} < 0.3$ from the particle q. The particle identity (electron, muon, photon, charged hadron or neutral hadron) is represented through a one-hot encoding.

Events are filtered requiring the presence of one isolated electron or muon with transverse momentum $p_T > 25$ GeV and particle-based isolation $\texttt{ChISO} + \texttt{GammaISO} + \texttt{NeuISO} < 0.3$. This lepton is the first entry of the list of particles for each event. To avoid double counting of this lepton as a charged particle, each charged particle q is required to have $\Delta R(q, \ell)$>0.0001. This baseline selection, which follows the typical requirements of an inclusive single-lepton trigger algorithm, accepts ~ 130 multijet events and ~ 110 W+jets events for every $t\bar{t}$ event. It is then a very inefficient selection to select $t\bar{t}$ events. This paper describes a more efficient strategy to select $t\bar{t}$ events, an example that could be generalized to other topologies.

All particles are ranked in decreasing value of their p_T. For each event, besides the isolated electron or muon, we store: the first 450 charged particles, the first 150 photons, and the first 200 neutral hadrons. This correspond to a total of 801 particles per event, each characterized by 19 features.

To feed these particles into a recurrent network, particles are ordered according to their increasing or decreasing distance from the isolated lepton. Different physics-inspired metrics are considered to quantify the distance (ΔR, $\Delta\phi$, $\Delta\eta$, k_T [7], or anti-k_T [8]). The best results are obtained using the ΔR decreasing distance ordering.

Events are also represented as abstract images in the (η, ϕ) plane. Each particle is represented as a semi-transparent (30% blending) geometric shape, centered at its (η, ϕ) coordinates and with size proportional to $\log p_T$, using the `scikit-image` library [9]. The image size and colors are uniquely associated to one of the particle classes: (i) green pentagons for the selected electron or muon; (ii) blue triangles for photons; (iii) red squares for charged hadrons; (iv) yellow circles for neutral hadrons. The missing transverse energy in the event is represented as a black circle. The image is then translated into an RGB array of size 3x500x314. An example of this event representation is shown in Fig. 1.

4 Recursive classifiers

We use the long term memory capable RNNs to aggregate the input sequence of particle flow candidate features into a fixed size encoding. The fixed encoding is fed into a fully connected layer with 3 softmax activated nodes. The LSTM [10] and GRU [11] models utilize the `KERAS` [12] defaults of a tanh activation, with hard-sigmoid inner activation. Since the list of particle flow candidates is

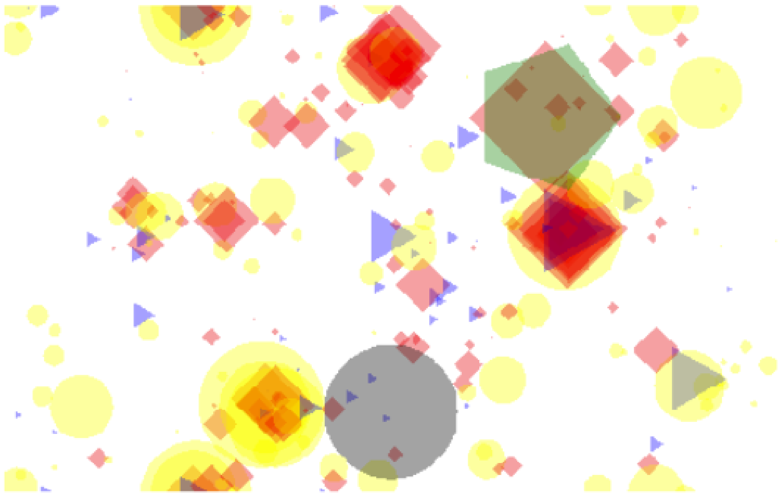

Figure 1: Example of a $t\bar{t}$ event, represented as an abstract image.

of variable sizes, the zero padded entries at the end of each list is masked at the beginning of the network. Additionally we normalize the data at training time so that each feature has zero mean and unit standard deviation.

The network training is performed in `KERAS`, using `THEANO` [13] as a back-end. For each network training we utilize early stopping with a patience of 8 epochs and a hard cap of 100 epochs. Training a single model takes roughly 12-24 hours over the course of 30-80 epochs on a single GeForce GTX 1080 GPU. The best internal width of the recurrent layers was determined by K-fold cross validation for each width trained on a different K-fold subsets of the training set each totaling 300,000 events. We found that for LSTMs a fixed width of 50 gave the best overall, and so in the interest of achieving the best validation performance we proceeded with this width. We tested the effects of the network architecture (GRU or LSTM) on the classification performance. In general, the GRU architecture out-performed the LSTM architecture, and we found that standardizing the particle features improved the results further.

5 CNN classifier

To classify events represented as abstract images, we use a CNN-based Densely Connected Convolutional Network (DenseNet) [14] model. The architecture of a DenseNet model consists of n densely connected *dense blocks*. Each block contains k sets of a batch normalization layer, a 3x3 convolutional layer, and an optional dropout layer with rate d. In between the dense blocks are *transition layers*, each of which includes a batch normalization layer, a 1x1 convolutional layer, an optional dropout layer with rate d, and a 2x2 average pooling layer. The input is fed into a convolutional layer with f output channels before entering the first dense block. A global average pooling is performed at the end of the last dense block, followed by a softmax classifier. In our experiment, we choose number of dense blocks $n = 2$, $k = 3$, dropout rate $d = 0.5$, and $f = 12$. We also replace the rectified linear units as the activation function of the batch normalization layers with the exponential linear units (ELUs) [15].

The training setup is similar to that of the recurrent networks of the previous section. The training is performed using the Adam optimizer [16] with learning rate 10^{-3} for 50 epochs. The checkpoint of the best epoch is used as the starting point of another 20-epoch long training, with learning rate reduced by a factor ten. This last operation is then repeated to obtain the optimal model.

6 Results

A comparison of the ROC curves for the three classifiers are shown in Fig. 2. The recursive classifiers select 99% of the $t\bar{t}$ events in the sample, reducing the QCD multijet and W+jets contamination to 8.6% of the original values. The DenseNet classifier allows to retain 90% of the $t\bar{t}$ events for a comparable false-positive rate.

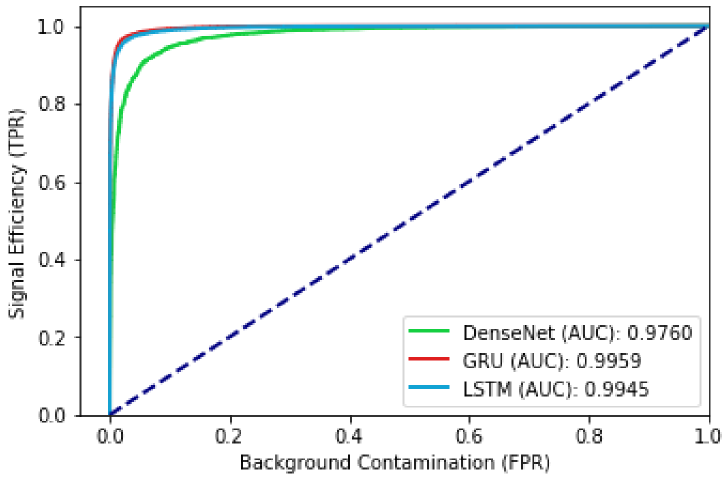

Figure 2: ROC curves for the classifiers described in the paper.

Considering that $\sim 30\%$ of the trigger bandwidth and disk resources are allocated to inclusive single-lepton triggers, adding a classifier like the one presented here for each event topology could allow to save a considerable amount of the total resources, which could then be used to reduce the applied thresholds of rule-based algorithms or to implement new trigger algorithms for new topologies.

7 Conclusion

We show how deep neural networks can be used to train a topology classifier for LHC collision events, associating the list of reconstructed particles to the process that originated them. Such an application could be used to reduce the amount of unwanted events currently accepted in the trigger system of the LHC experiments. On single-lepton events, it would be possible to suppress by an order of magnitude the background from QCD multijet events while retaining $> 90\%$ of the interesting events. The savings in resources (disk and bandwidth) could be used to extent the physics program of the ATLAS and CMS experiments.

8 Acknowledgments

This project is partially supported by the United States Department of Energy, Office of High Energy Physics Research under Caltech Contract No. DE-SC0011925. JR is partially supported by the Office of High Energy Physics HEP-Computation. MS is grateful to Caltech and the Kavli Foundation for their support of undergraduate student research in cross-cutting areas of machine learning and domain sciences.

References

[1] **CMS** Collaboration, A. Sirunyan, A. Tumasyan, W. Adam, et al., *Particle-flow reconstruction and global event description with the cms detector*, *Journal of Instrumentation* **12** (2017), no. 10 P10003.

[2] P. Baldi, P. Sadowski, and D. Whiteson, *Searching for exotic particles in high-energy physics with deep learning*, *Nature Communication* **5** (07, 2014) 4308.

[3] G. Louppe, K. Cho, C. Becot, and K. Cranmer, *QCD-Aware Recursive Neural Networks for Jet Physics*, `arXiv:1702.00748`.

[4] C. F. Madrazo, I. H. Cacha, L. L. Iglesias, and J. M. de Lucas, *Application of a Convolutional Neural Network for image classification to the analysis of collisions in High Energy Physics*, `arXiv:1708.07034`.

[5] T. Sjöstrand, S. Ask, J. R. Christiansen, et al., *An Introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159–177, [`arXiv:1410.3012`].

[6] **DELPHES 3** Collaboration, J. de Favereau, C. Delaere, P. Demin, et al., *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, *JHEP* **02** (2014) 057, [`arXiv:1307.6346`].

[7] M. Cacciari, G. P. Salam, and G. Soyez, *The anti-k_t jet clustering algorithm*, *Journal of High Energy Physics* **2008** (2008), no. 04 063.

[8] S. Catani, Y. L. Dokshitzer, M. H. Seymour, and B. R. Webber, *Longitudinally invariant K_t clustering algorithms for hadron hadron collisions*, *Nucl. Phys.* **B406** (1993) 187–224.

[9] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, et al., *scikit-image: image processing in Python*, *PeerJ* **2** (6, 2014) e453.

[10] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural Comput.* **9** (Nov., 1997) 1735–1780.

[11] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches*, *CoRR* **abs/1409.1259** (2014) [`arXiv:1409.1259`].

[12] F. Chollet et al., "Keras." `https://github.com/fchollet/keras`, 2015.

[13] **Theano Development Team** Collaboration, R. Al-Rfou, G. Alain, A. Almahairi, et al., *Theano: A Python framework for fast computation of mathematical expressions*, *arXiv e-prints* **abs/1605.02688** (May, 2016).

[14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[15] D. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus)*, *CoRR* **abs/1511.07289** (2015) [`arXiv:1511.07289`].

[16] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, *ArXiv e-prints* (Dec., 2014) [`arXiv:1412.6980`].