Deployment of ML in Changing Environments

Marco Barbone[1], *Christopher* Brown[1,*], *Benjamin* Radburn-Smith[1], and *Alexander* Tapper[1]

[1]Imperial College London, South Kensington, London, United Kingdom

Abstract. The High-Luminosity LHC upgrade of the CMS experiment will utilise a large number of Machine Learning (ML) based algorithms in its hardware-based trigger. These ML algorithms will facilitate the selection of potentially interesting events for storage and offline analysis. Strict latency and resource requirements limit the size and complexity of these models due to their use in a high-speed trigger setting and deployment on FPGA hardware. It is envisaged that these ML models will be trained on large, carefully tuned, Monte Carlo datasets and subsequently deployed in a real-world detector environment. Not only is there a potentially large difference between the MC training data and real-world conditions but these detector conditions could change over time leading to a shift in model output which could degrade trigger performance. The studies presented explore different techniques to reduce the impact of this effect, using the CMS track finding and vertex trigger algorithms as a test case. The studies compare a baseline retraining and redeployment of the model and episodic training of a model as new data arrives in a continual learning context. The results show that a continually learning algorithm outperforms a simple retrained model when degradation in detector performance is applied to the training data and is a viable option for maintaining performance in an evolving environment such as the High-Luminosity LHC.

1 Introduction

Machine learning (ML) is often deployed in environments where the training and inference environment differ. These differences could be part of the training process itself, for example using Monte Carlo to train a model and then deploying it on real data. Alternatively these differences could arise over time with the inference environment shifting from the training environment in a variety of ways. This difference in the training and inference environment can lead to a model's performance changing, often in unknown ways. In systems, such as a trigger system, where the inference of the model is used to make critical choices, this loss of performance could lead to loss of data, unquantifiable inefficiencies or unknown biases. There are several strategies to avoid these issues:

- Do nothing. Allow the model's output to drift and change how this output is used. This is detrimental if the output changes dramatically or the changing environment is poorly understood.
- Retrain the model. When the model's output drifts it can be retrained and redeployed. This requires storing additional data and the computational overhead of the retraining.

*e-mail: cebrown@cern.ch

- Design a robust model. Train the model so that the output doesn't change with the environment, for example Ref. [1]. This can be difficult if the environmental change is unknown or a large departure from the initial training.
- Quantify the uncertainty of the model, for example Ref. [2]. Use ensemble techniques or Bayesian ML [3] to understand how sure the model is of a prediction. This requires additional computational cost and inference time and does not compensate for the changing environment, but informs downstream users that something has changed.
- Continually train the model as the environment changes. Use a continual learning approach [4] such as replay [5] or regularisation [6] to allow the model to learn about the new data whilst not forgetting about its initial training. This can result in dynamically changing models and needs a continual stream of labelled data for training.

2 CMS as a changing environment

The CMS Phase-2 Level-1 (L1) trigger will use many ML models to enhance its algorithms [7]. However, CMS is a changing environment and this ML could suffer from degraded inference and also has additional constraints that make certain techniques difficult. The models are all:

- Lightweight. Resource constraints mean highly robust models cannot be trained because they are too small to fully capture a wide range of possible inputs.
- Focused. The models make inference on specific detector data and will be directly impacted by these detectors changing.
- Trained on MC. All models are trained on MC data which will be different to the real detector, sometimes in unpredictable ways.

The CMS detector environment changes at a variety of time scales, all of which could cause differences in the inference of ML in the trigger. At the fastest level, within a fill, the beam conditions of the LHC change, however this is unlikely to make large impact on the trigger ML due to the small changes. It would be useful to have some level of uncertainty quantification in the ML models as this would allow their robustness to such changes to be monitored over the fill and downstream users can adapt based on the certainty of the predictions. Most uncertainty quantification relies on multiple models being evaluated at inference time which is currently difficult in the trigger environment where low latency is a priority.

Between fills there could be larger changes to the detector such as general degradation to the subsystems due to radiation (seen in the CMS ECAL crystals and currently corrected for using a laser monitoring system [8]) or sudden changes to the detector such as a loss of high voltage or cooling. To avoid these changes impacting the trigger, there is scope for the ML of the trigger to be updated in some way. This would require a stream of labelled truth data be generated either from full offline analysis or from the L1 scouting system that runs at 40 MHz collecting unbiased data [9]. With this data the trigger ML could have some level of retraining or a continual learning top-up training where the existing ML is updated in a small way, governed by a CL algorithm.

During long (year end) technical stops where CMS is recommissioned and the detector environment is fully evaluated, the ML in the trigger could be completely retrained and updated. This could be carried out based on extensive MC campaigns or previously collected data and could be used as a time to evaluate and improve algorithms.

3 Using trigger ML in a changing CMS detector

To explore the implications of using ML in a changing detector environment such as the CMS detector and use of different approaches to training this ML, an example convolutional neural network (CNN) that identifies fake vertices was used. This model is designed to find where the current CMS Level-1 trigger algorithms have misidentified the primary vertex in an event. To emulate a changing environment MC samples were used with degraded tracking performance.

3.1 Tracker Degradation

Initially, a large top quark pair production sample used as the base training for all models. This represents the large sample used to train models before their deployment in the changing environment. Four further samples with different tracker degradation scenarios were then subsequently used. The first of these degraded samples introduces a bias rail inefficiency (BRI) to the tracker modules which sees an inefficiency in the silicon of the tracker causing fewer hits from charged particles to be registered. This represents an environment with an intrinsic difference to what was trained on. The other three samples contain the bias rail inefficiency and then have a random removal of 1, 5, and 10% of strip modules of the tracker to emulate the gradual degradation of the tracker over time.

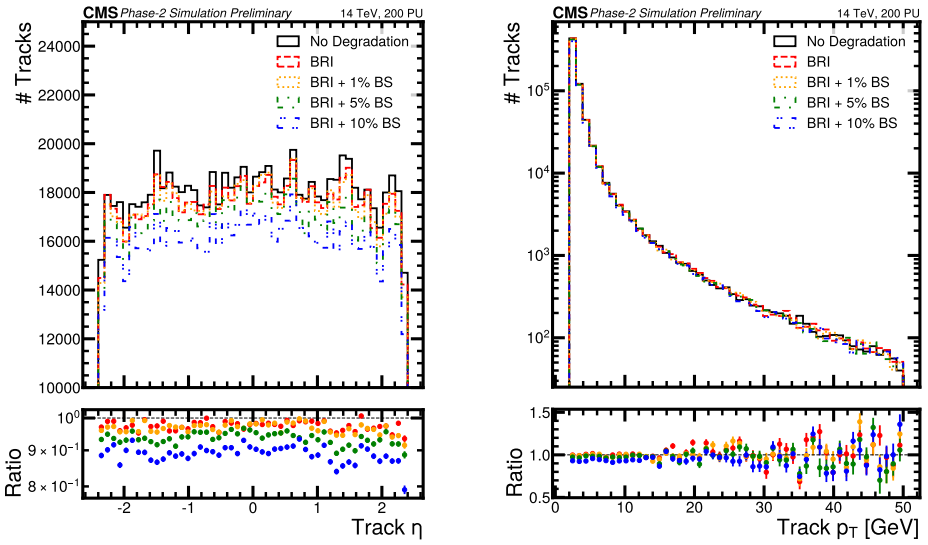
The reconstructed pseudorapidity (η) and transverse momentum (p_T) of the Level-1 trigger reconstructed tracks are shown in Figures 1a and 1b. These figures demonstrate no geometrical dependence to the loss in tracker performance, shown by the flat η distribution, but show a drop in the number of tracks being produced. This feature is very clear in Figure 1a with higher levels of degradation removing more tracks. The loss in tracks occurs at lower p_T as shown in the lowest p_T bins of Figure 1b.

3.2 Fake Vertex Identification

To test the impact of the degraded samples and to evaluate different training schemes, a fake vertex identification task was used as an example Level-1 trigger ML algorithm. This was a five layer CNN that aimed to identify fake vertices being produced by the Level-1 vertex reconstruction algorithm. This is of importance to downstream users of the vertex such as the pile-up per particle identification algorithm which is vital for the performance of jet finding and energy sums in high pileup environments [7]. The CNN was trained on two-dimensional histograms made up of Level-1 track features first histogrammed in z_0 then stacked on top of one another. Examples of a real and fake vertex are shown in Figures 2 and 3, respectively. The true vertex histogram clearly shows a band of tracks with high p_T at $z_0 = 1$ cm, this band has been identified by the vertex algorithm giving a correct vertex. Figure 3 shows a more ambiguous case with a band of tracks at $z_0 = -5$ cm which is the true vertex but an isolated high p_T fake track at $z_0 = 10$ cm. This fake track has been returned as the vertex leading to an incorrect vertex prediction.

3.3 No Retraining Model

A CNN was trained on the large "no degradation" sample to identify the fake vertices. It was then evaluated on a testing sample of this "no degradation" sample as well as testing samples of the degraded tracker. No retraining was performed to emulate a model that has been left



(a) (b)

Figure 1. Level-1 trigger reconstructed track parameter distributions for different levels of tracker degradation. Plot (a) shows the η distribution across the different samples with BRI referring to the bias rail inefficiency and X% bad strips (BS) referring to a random removal of random strips from the outer tracker. Plot (b) shows the p_T distribution for the same set of samples. Both demonstrate a drop in the number of tracks being produced as the inefficiencies are added with 10% fewer tracks seen with the worse degradation. The lost tracks tend to be lower p_T with a larger drop in the smallest p_T bin.

in the changing environment.

Figure 4 shows the performance of this model. The black line clearly demonstrates that when evaluated on the testing set of its training the model performs very well, demonstrating the task is being effectively learnt. As soon as the degradation is introduced the performance of the model drops significantly and is worse as the degradation increases. This model is ill-prepared for a changing environment.

3.4 Top-Up training

In order to improve this performance, the model was retrained. For each of the degraded samples the model was retrained in a top-up manner. This meant that the original model was kept and new samples are introduced to the model with a lower learning rate to slowly train the model. Each of the four degraded samples were used to train the model one by one with the evaluation of the corresponding test sample performed after each training. The final model that had been top-up trained on every degraded sample was then evaluated on the test sample of the original, "no degradation" sample.

Figure 5 demonstrates that the model has improved its performance across the degraded samples with the area under the ROC curve improving by around 2% across all samples. The model has been able to learn from the degraded samples which is expected as the samples

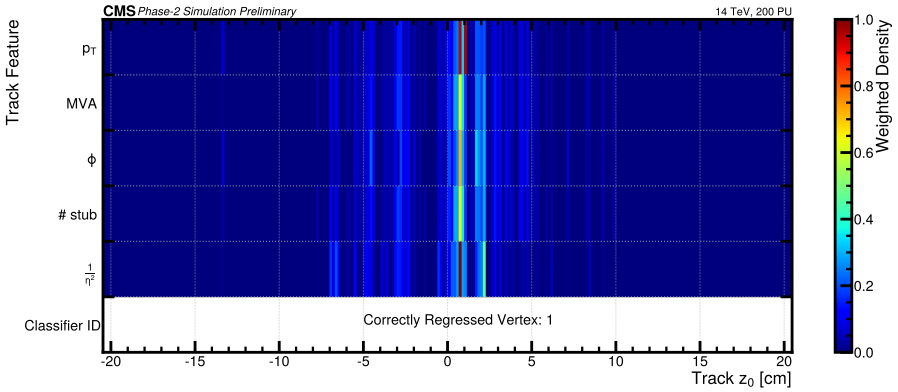


Figure 2. L1 trigger reconstructed track parameter two-dimensional histogram generated from histogramming all tracks in z_0 weighted by different track parameters. This specific two-dimensional histogram, taken from a top quark pair production sample shows an event where the Level-1 trigger vertex finding algorithm has correctly identified the vertex. The large band of tracks at $z_0 = 1$ cm is the true primary vertex with many tracks with high p_{T} and high quality tracks. Because of the high p_{T} peak, the vertex is well reconstructed by the algorithm. MVA refers to a boosted decision tree trained to identify fake tracks.

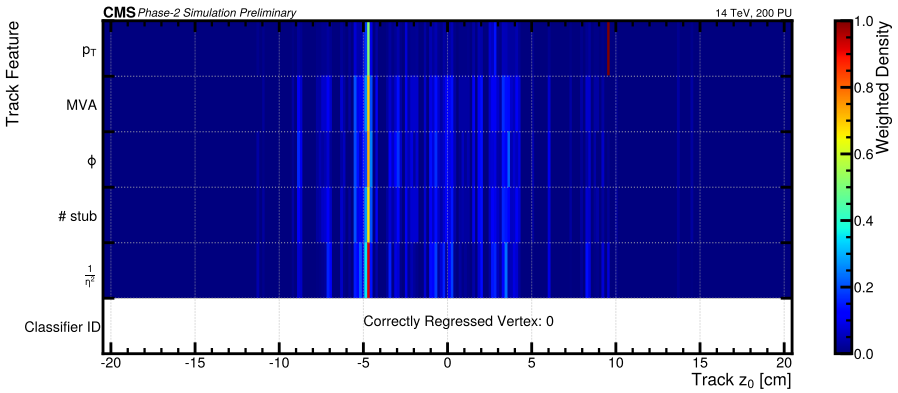


Figure 3. Level-1 trigger reconstructed track parameter two-dimensional histogram generated from histogramming all tracks in z_0 weighted by different track parameters. This specific two-dimensional histogram, taken from a top quark pair production sample shows an event where the baseline Level-1 trigger vertex finding algorithm has incorrectly identified the vertex. The high, many track peak at $z_0 = -5$ cm is the true primary vertex. However, the high p_{T}, low quality isolated track, which is likely fake, at $z_0 = 10$ cm has higher p_{T} so is incorrectly selected as the primary vertex.

themselves do not have major changes to the overall features. The model has lost performance in its original training, a feature that is common across retrained models and termed "catastrophic forgetting". While the forgetting in this model is not catastrophic it shows that with more retraining a model will lose performance as the model has not been able to learn to ignore the noise that the degraded samples introduces, instead it has just fit to the degradation.

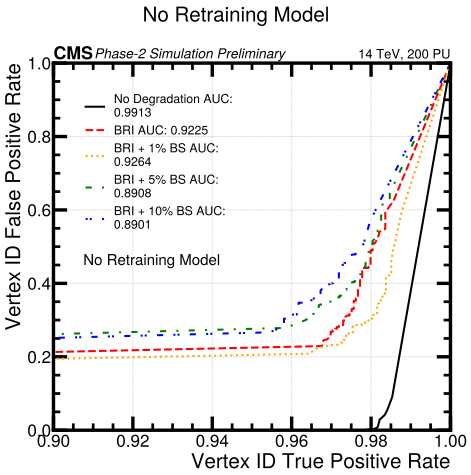


Figure 4. Receiver operating characteristic curve for the CNN trained only on the "no degradation" sample. Performance in the corresponding test dataset of the "no degradation" sample is shown in black, the other degraded samples are shown in the other colours. The model's performance in its training domain is very high, shown by the black curve. The drop in performance on the other samples is due to the degradation; as the degradation increases the performance drops. There is a small difference when comparing the BRI and BRI + 1% BS samples due to the relatively low statistics of the testing samples.

3.5 Continual Learning Top-Up Training

A method to improve a model's training when performing these kinds of top-up trainings is to use a continual learning (CL) algorithm. While this is a very open area of research with many different techniques [10], one of the simplest was chosen for this example. A replay buffer was introduced to the training such that samples from previously seen training rounds were kept and replayed to the model mixed in with the current training round so that the model is always seeing previous experiences. The model was trained as before in the top-up case but with the buffer added that kept some of the degraded samples; samples from the original "no degradation" sample were also added to this buffer. The replay training was performed using the Avalanche package [11] for CL which allows various CL algorithms to be implemented and has the utility to treat datasets as individual experiences being introduced to the model one at a time.

Figure 6 shows that the CL training has improved performance across all the degraded samples. The false positive rate has dropped to close to 0% in the degraded samples with very high true positive rates. This improvement in performance demonstrates that the model has learnt underlying features in the data and can ignore the noise that the degradation introduces. The model produced is far more robust to detector changes and can perform well across the full range of degradation. The performance in the original training data set, the "no degradation" sample, has also improved relative to the standard top-up training workflow in section 3.4 meaning the model has not forgotten its original training. This is due to examples of the original training being mixed in the replay buffer. The performance is not as good as the original model's performance in this sample (section 3.3) but this could be tuned with different CL strategies or a larger emphasis of these samples in the replay buffer. One of the

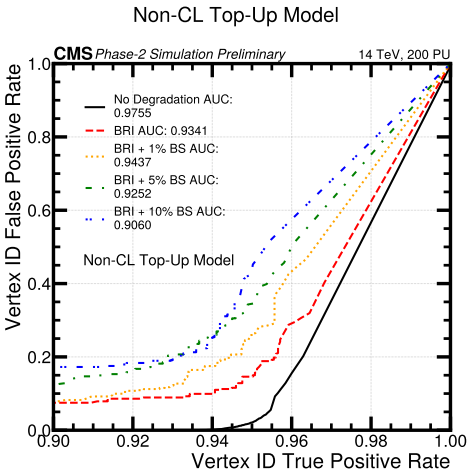


Figure 5. Receiver operating characteristic curve for the CNN trained on the "no degradation" sample and then top-up trained on the degraded samples. Performance in the corresponding test dataset of the "no degradation" sample is shown in black, the other degraded samples are shown in the other colours. The top-up trained model has improved its performance in the degraded samples as it has actually seen examples of the degradation and learnt from them, it is clear that larger degradations are still harder to learn from. The model has lost its performance in the original domain because of the top-up training which is an example of a model forgetting its previous training.

advantages of the CL approach is that specific performance can be emphasised by altering the training mechanism.

4 Conclusions

The use of machine learning (ML) in the changing environment of the CMS Level-1 Trigger could result in problems due to a changing inference environment. Many contribute to important trigger decisions so their inferences need to be robust and reliable. When an example CMS Level-1 trigger model is introduced to a changing environment with a degraded tracker sample, its performance becomes worse, demonstrating that strategies need to be implemented to improve these models.

Two different approaches to retraining this model were explored. The first was a standard top-up training workflow which was shown to improve the model's performance in the degraded sample at the expense of worse performance in the model's original training domain. A continual learning (CL) approach was also explored where some training samples were stored in a replay buffer and played back to the model as it learns. This improved the models robustness to the changing environment in all the degraded samples; it also retained its performance in the original training domain. This is the first use of CL in CMS, or any of the LHC experiments. Further research is needed to understand how this workflow can be applied to existing ML algorithms being used in online systems across these experiments.

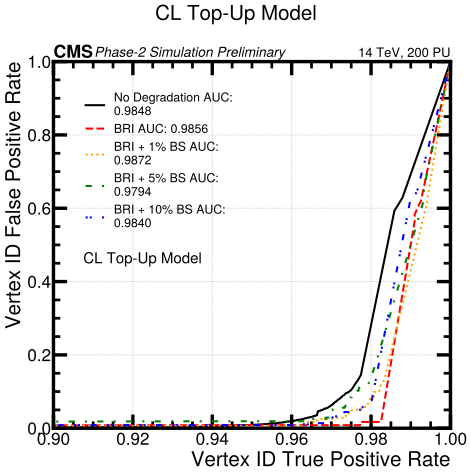


Figure 6. Receiver operating characteristic curve for the CNN trained on the "no degradation" sample and then top-up trained on the degraded samples using a replay buffer continual learning algorithm. Performance in the corresponding test dataset of the "no degradation" sample is shown in black, the other degraded samples are shown in the other colours. The performance across all degraded samples has improved, when comparing Figures 4 and 5, the CL model is more robust to the changing environment. The performance in the original domain, while better than the top-up trained model in Figure 5, is not as good as the model solely trained in this domain in Figure 4. The CL model is able to forget less about its original training but still suffers from some level of forgetting.

References

[1] L. Schmidt et al., *Adversarially Robust Generalization Requires More Data*, in *Advances in Neural Information Processing Systems* (2018), Vol. 31

[2] M. Abdar et al., Information Fusion **76**, 243 (2021)

[3] L.V. Jospin, H. Laga, F. Boussaid, W. Buntine, M. Bennamoun, IEEE Computational Intelligence Magazine **17**, 29 (2022)

[4] Hadsell, Raia and Rao, Dushyant and Rusu, Andrei A. and Pascanu, Razvan, Trends in Cognitive Sciences **24**, 1028 (2020)

[5] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, G. Wayne, *Experience Replay for Continual Learning*, in *Advances in Neural Information Processing Systems* (2019), Vol. 32

[6] J. Kirkpatrick et al., Proceedings of the National Academy of Sciences **114**, 3521 (2017)

[7] The CMS Collaboration (CMS), Tech. rep. (2020), `https://cds.cern.ch/record/2714892`

[8] M.A. Wadud (CMS), Tech. rep. (2019), `https://cds.cern.ch/record/2797776`

[9] T.O. James (CMS), Tech. rep. (2023), `https://cds.cern.ch/record/2852916`

[10] L. Wang, X. Zhang, H. Su, J. Zhu, *A Comprehensive Survey of Continual Learning: Theory, Method and Application* (2023), `2302.00487`

[11] V. Lomonaco et al., *Avalanche: an End-to-End Library for Continual Learning*, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2021), pp. 3595–3605