University of BRISTOL

Neural network triggers for the CMS detector at the HL-LHC

Name
Kyle Pidgeon

Course
MSci Physics with Astrophysics

Supervisors
Dr Sudan Paramesvaran & Dr Emyr Clement

Year of submission
2021

Number of words
6943

School of Physics
University of Bristol
Tyndall Avenue
Bristol BS8 1TL

Declaration

It was suggested, by our supervisors, that we focus on the $HH \to b\bar{b}b\bar{b}$ signal in this project. Furthermore, the Monte-Carlo data was provided by Emyr Clement, and Sioni Summers (CERN) provided example VHDL code for use with the EMP framework. Apart from those inputs, my project partner and I contributed equally to all parts of the project.

Abstract

The High-Luminosity upgrade to the LHC (HL-LHC) will enable the next generation of precision measurements in particle physics. However, the resulting high-pileup environment requires improved rejection of background, whilst maintaining sensitivity to interesting physics. This report describes the development of neural network algorithms for use in the CMS detector's Level-1 Trigger system at the HL-LHC, with the aim of selecting $HH \to b\bar{b}b\bar{b}$ events against pileup. Using quantisation-aware training, a low-resource trigger was produced for this signal, with greater than 50% sensitivity at a 10kHz fake rate, and successfully deployed to FPGA hardware.

Contents

1 Introduction

The *High-Luminosity Large Hadron Collider* (HL-LHC) is a planned upgrade to the LHC that will increase the instantaneous luminosity by a factor of 5, to $\sim 7.5 \times 10^{34}$cm^{-2}s^{-1} [1], compared to the current value. In its ultimate configuration, the HL-LHC will reach an average of 200 *pileup* (PU) events per *bunch crossing* (BX), and a total of 3-4ab^{-1} of data is expected over a decade of operation. This unprecedented amount of information will enable more precise measurements of *Standard Model* (SM) parameters and stronger constraints on *beyond-the-Standard-Model* (BSM) theories, alongside the possibility of discovering the nature of dark matter.

In order to cope with the increased event rate at the HL-LHC, the *Compact Muon Solenoid* (CMS) detector [2] will undergo its own upgrade to 'Phase-2' [3]. Among the components that will see improvements is the *Level-1 Trigger* (L1T), which employs algorithms on low-latency hardware to make an initial decision for the acceptance or rejection of an LHC event. The Phase-2 upgrade will grant the L1T access to fine-granularity detector information, and aims to relax the latency and bandwidth constraints on its constituent algorithms [4]. Currently, the L1T uses mainly heuristic-based algorithms for determining whether a given event contains physics of interest; these Phase-2 improvements, however, are motivating investigations into *machine learning* (ML) for these tasks [4], which could increase sensitivity to rare signals whilst suppressing the high levels of pileup.

The work presented in this report is an example of such an investigation, where ML algorithms were developed to distinguish interesting physics from the overwhelming HL-LHC background, using image-based data. A subset of these algorithms were then implemented on hardware, similar to that targeted for the L1T upgrade, in order to verify their functionality.

2 Background

2.1 CMS at the HL-LHC

The CMS detector [2] is one of two general-purpose detectors at the LHC, alongside ATLAS [5], and makes measurements of the myriad particles in the final states of proton-proton collisions. CMS is a hermetic detector, with close to 4π coverage of the beamline, encompassing a large region of phase-space in order to observe a vast array of event topologies — given the detector geometry, its coordinate system is usually defined in terms of the *pseudorapidity*, η, and the azimuthal angle, ϕ [2]. One of the defining properties of CMS is the 4T superconducting solenoid, enabling precision measurements of charged-particle momenta. High-resolution Inner Tracker and muon detection systems make these determinations realisable, with the former also playing important roles in pileup mitigation [6] and identification of displaced vertices. Utilising these systems, together with high-granularity calorimeters, CMS has produced precise tests of SM and BSM theories. The detector has played a pivotal role in many discoveries, such as that of the Higgs boson [7] and of ultra-rare processes such as the $B_s^0 \to \mu^+\mu^-$ decay [8, 9].

Amongst other SM phenomena, the CMS Phase-2 detector presents exciting prospects for detailed studies of the *electroweak symmetry breaking* (EWSB) mechanism, and for exploiting recently observed avenues such as *vector boson scattering* (VBS) [10]. Furthermore, projections suggest that BSM-enhanced decays such as $B^0 \to \mu^+\mu^-$ will be promoted to well beyond 5σ-significance [11]. Improved suppression of pileup at the HL-LHC will be necessary for this next era of measurements; whilst changes to components such as the Inner Tracker will play a significant role in this, the upgrade to the L1T will utilise these to form an initial barrier against background, and is the system on which this project focuses.

2.2 The Phase-2 Level-1 Trigger

2.2.1 CMS trigger system

The trigger system at CMS comprises two physical tiers: the L1T, which is a collection of algorithms operating on low-latency, highly-parallel hardware, in order to produce a tentative, accept-or-reject, decision for a given BX; and the High-Level Trigger (HLT), which runs a more sophisticated set of decision-making tools via online software [4].

The L1T receives event input from various detector systems, and operates on the information using algorithms – *triggers* – that determine whether the event is associated with an interesting physics signal, or consists of background only. Several sub-triggers constitute the L1T, and ultimately the processed data arrives in the *Global Trigger* (GT); the GT implements a 'trigger menu' [12], which contains many triggers looking for various signals of interest to CMS, and which provides the final accept-or-reject

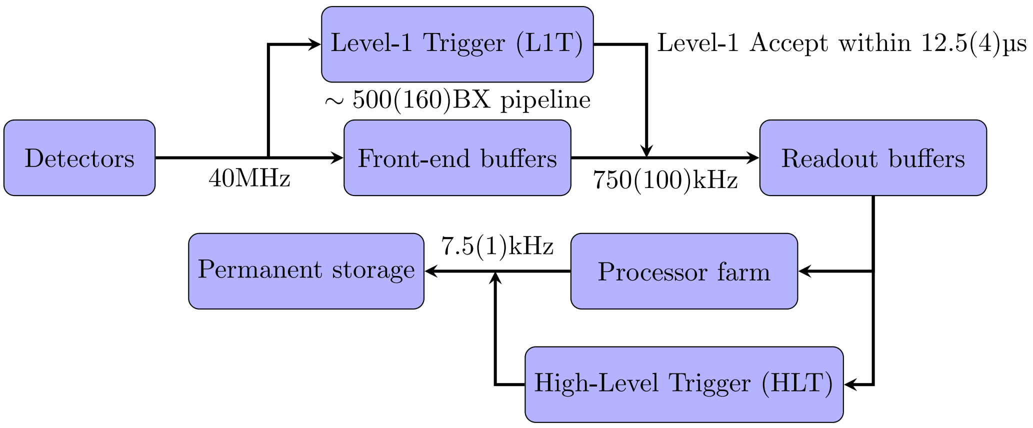

Figure 1: The flow of data through the CMS trigger system. Detector output is buffered in pipelined front-end electronics, providing a fixed latency within which the L1T must provide a trigger decision. In the Phase-2 L1T, the event rate will be reduced from 40MHz to 750kHz at the first stage, with a further 100× reduction by the HLT for permanent storage. The stated readout rates are for the Phase-2 L1T at $\langle \text{PU} \rangle = 200$, with approximate Phase-1 values given in parentheses.

decision for a given event. The combined output rates from all of the menu algorithms is constrained by the downstream systems – the HLT and, ultimately, the throughput to permanent storage – to 100kHz in the current system. Thus, each trigger algorithm must contribute a small fraction of the output bandwidth in order to search for a wide range of event properties. Moreover, the GT decision must be made within 4µs of a given BX, limited by the detector front-end electronics — this low-latency operation requires the triggers to be implemented in *field-programmable gate array* (FPGA) hardware. Downstream, the HLT further quenches the event rate by utilising more rigorous trigger algorithms on standard computer hardware. A schematic of the CMS trigger system is shown in Fig. 1 — the trigger currently reduces the LHC event rate from 40MHz to 1kHz, but this output rate will increase in the upgraded system, allowing for greater statistical power in measurements.

2.2.2 Level-1 Trigger upgrade

As part of the Phase-2 upgrade to CMS, the L1T will be upgraded to improve the rejection of background and acceptance of signals at the HL-LHC. The new system will make use of additional resolution gained by upgraded calorimeters, alongside the inclusion of tracking information, in order to implement *particle-flow* (PF) event reconstruction at the hardware level [4]. Pileup mitigation techniques, namely the *pileup per particle identification* (PUPPI) [6] algorithms formerly implemented in the HLT, will also be enabled via tracking. These methods are anticipated to be used in the first level of a new *Correlator Trigger* (CT); subsequently, in the Layer-2 Correlator, sets of PUPPI particle candidates will be used to form high-level physics objects such as jets. Due to the use of PUPPI data, the triggers produced in this project would, hypothetically, be situated in the Layer-2 Correlator trigger. Additionally, the L1T latency constraint is due to be relaxed to 12.5µs and the output bandwidth increased to 750kHz, compared to the current parameters. These changes, together with additional resources and the need for high-specificity triggers, enables and motivates the broad use of ML trigger algorithms in the L1T for the first time [4].

2.3 Machine learning for the Level-1 Trigger

2.3.1 Related work

Although machine learning has only recently become the focus of L1T trigger studies, it has already been utilised in the Run-2 Endcap Muon Track Finder for muon momentum regression. Here, a *boosted decision tree* (BDT) was fitted offline using a discretised input space, with the resultant mapping between inputs and outputs stored in a 1.2GB look-up table in hardware [13]. This technique achieves incredibly low latency in the L1T by requiring a single operation only, and could prove useful in other applications with a small number of input variables.

Motivated by the high-pileup environment at the HL-LHC, ML algorithms are under investigation elsewhere in the L1T, such as in the Barrel Muon Track Finder [14], with the aim of reducing trigger rates

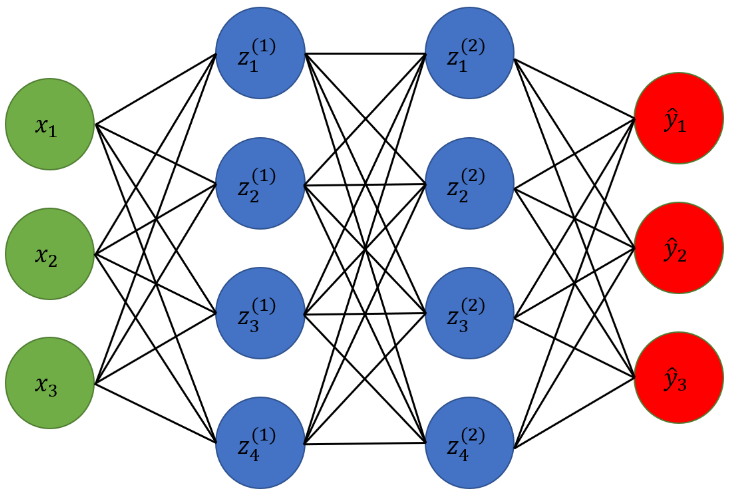

Figure 2: An example MLP architecture. The inputs to the network are shown in green on the left, and the outputs are on the right in red, with blue nodes comprising 'hidden' layers. The variable definitions are given in the text.

whilst maintaining or improving upon selection efficiency. Several preliminary studies have explored the use of *deep neural networks* (DNNs) in the GT, targeting physics of high importance at the HL-LHC, such as *vector boson fusion* (VBF) modes and the decay $HH \to b\bar{b}b\bar{b}$. For example, an autoencoder trained on a minimum bias sample was tested on such events [4] and found to be sensitive to them, even though they were unseen during training. Unsupervised methods such as this have potential for triggering on new physics, as models can be produced independent of a specific signal hypothesis. In many cases like this, physics-motivated variables are used as input to a model; however, a desirable use-case of ML in particle physics [15] is to skip the feature-engineering altogether and allow DNNs to exploit a high-dimensional feature space, which was done in the work presented in this report. Other research has shown that a combination of particle-level information and high-level features provides the best results for a $t\bar{t}$ selector [16] in software. The introduction of PF algorithms to the L1T could facilitate such methods for use in hardware triggers, although it was not studied in this project.

Finally, investigations involving *convolutional neural networks* (CNNs; described below), of particular interest here, show improvements upon traditional algorithms both at the HLT level [17, 16] and within ATLAS' Level-0 hardware trigger [18].

2.3.2 Neural networks

The above applications are examples of how neural networks can be used for estimating an arbitrary mapping from an input to an output. In the case of the archetypal 'feed-forward' neural network, the *multilayer perceptron* (MLP), this is achieved by transforming an input vector with components x_i to an output vector with components $\hat{y}_k$ via a set of intervening nodes, arranged in layers as shown in Fig. 2. The edges of the network represent weights, which encode the strength of dependence of a node in layer L on a node in layer $L-1$. More precisely, the j^{th} node in layer L produces outputs, $z_j^{(L)}$, that are a non-linear combination of N inputs from layer $L-1$, $x_i^{(L-1)}$, such that [19]

$$z_j^{(L)} = h\Big(\sum_{i=1}^{N} w_{ji} x_i^{(L-1)} + w_{j0}\Big). \tag{1}$$

Here, w_{ji} are the connection weights, w_{j0} is a bias term, and h is an activation function, such as tanh, that introduces a non-linearity to the output. Equation 1 shows how neural networks achieve their generality: with multiple hidden layers, the basis functions of the summand are adaptable alongside the weights and biases. Supervised training, as was used in this project, allows a model to learn optimal values of these parameters given some labelled data. The learning process is iterative, and occurs via backpropagation and stochastic gradient descent [20] with respect to a cost function.

With MLPs, the dense connectivity of a network, as in Fig. 2, can result in overfitting, due to the potentially large number of basis functions that can enter into Equation 1. Forms of regularisation, such as weight pruning [21], can be used to prevent this, by constraining the values of parameters. Another method is to make regularisation intrinsic to the model architecture [19] by using CNNs.

Convolutional neural networks function in a similar way to MLPs, in that they operate in a feed-forward fashion during inference, and are trained using the same optimisation methods. However, instead of the matrix multiplications like that of Equation 1, they involve convolutions of weight kernels – or *filters* – with multidimensional inputs. These weights are shared by all of the output features, which helps to significantly decrease the number of parameters when compared to MLPs. Furthermore, the convolution operations in a CNN are generally followed by pooling, a form of downsampling, which replaces the outputs by a local summary statistic. These features of CNNs, amongst others, have seen them achieve particular success in problems involving two-dimensional image topologies [20].

Given the success of machine learning algorithms for creating particle physics triggers thus far, and the aptitude CNNs possess for image-based tasks, these types of neural network were a primary focus of this project. Additionally, for use in the L1T, the implementation of neural networks in the constrained environment of an FPGA needs to be considered.

2.3.3 FPGAs

Field-programmable gate arrays are highly parallel, re-configurable hardware designs, sitting somewhere between *application specific integrated circuits* (ASICs) and CPUs. The L1T utilises FPGAs due to their ability to process information within the very tight $\mathcal{O}(1\mu s)$ latency constraints of its algorithms. Furthermore, their programmable nature provides something that ASICs do not: an adaptable physics program. This manifests itself in the GT menu, where a set of algorithms running in parallel, on FPGA hardware, can change according to CMS' physics interests.

The fundamental components of a FPGA are *flip-flops* (FFs), *look-up tables* (LUTs), and embedded *block RAM* (BRAM) [22]. A further component, the *digital signal processing* (DSP) block, integrates FFs and LUTs to provide dedicated *multiply-accumulate* (MAC) circuitry. Combined, these elements are capable of representing the complex algorithms used in the L1T, but are a finite resource, therefore restricting the types of trigger that can be implemented.

The configuration of these elements – firmware – in an FPGA can be defined at several levels of abstraction: high-level descriptions written in `C/C++`, *register-transfer level* (RTL) written in a hardware description language such as VHDL, and gate level. Firmware described at a high level can be compiled to RTL via *high-level synthesis* (HLS) — in this report, 'HLS' will also denote an algorithm that is at this stage of development. RTL can then be compiled to the lowest-level abstractions using a logic synthesis tool.

2.3.4 Neural networks on FPGAs

The heavily constrained resources of FPGAs, alongside the fact that neural networks can involve arbitrarily large numbers of operations, produces strict limitations on the complexity of models that can be implemented on them. Every multiplication involves MAC operations and, hence, a large number of DSPs are generally required to implement them. In a CNN, the number of multiplications can be reduced due to the weight-sharing of filters, although this benefit disappears if large numbers of filters are used in a given convolutional layer.

Several methods exist for reducing the resource consumption of neural networks on FPGAs, and have desirable side-effects in some cases. The three main categories of optimisation are described below.

- **Compression**: The number of MAC operations required to implement a neural network can be reduced through altering the model architecture – decreasing the number of filters in a convolutional layer, for example – or by introducing sparsity. Common methods for doing this are 'dropout' regularisation and low-magnitude pruning, both of which remove some fraction of connections from the model, and can result in improved performance [21].

- **Quantisation**: FPGAs implement fixed-point arithmetic, meaning quantities must have fixed – or *quantised* – precision, $\langle X, Y \rangle$, where X is the total bitwidth and Y the number of integer bits. Furthermore, in the resource-limited environments of edge devices like FPGAs, quantisation can be used as a tool for resource optimisation by specifying an arbitrarily small value of X. There are two primary techniques for quantising neural network parameters: *post-training quantisation* (PTQ) and *quantisation-aware training* (QAT). The former reduces the precision in a brute-force manner after training the model, whilst the latter can incorporate lower parameter precision into

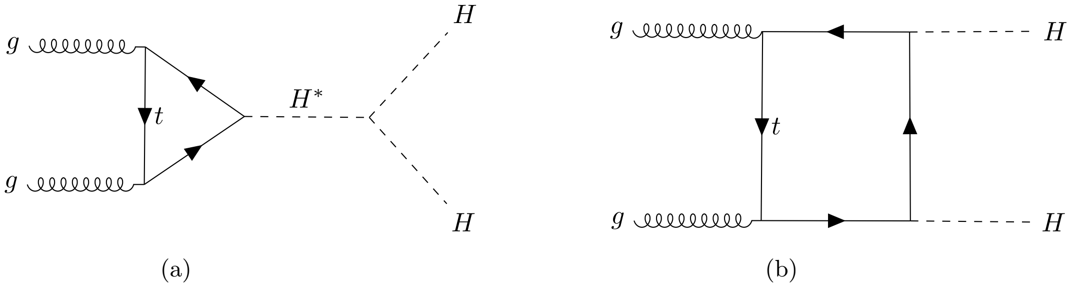

Figure 3: Feynman diagrams of Higgs pair production via gluon-gluon fusion. These processes, with the top-quark mediator, are the largest contributors to the SM di-Higgs production cross-section, with the bottom-quark variant contributing 1% at leading-order.

the training process, therefore producing a model optimised for an FPGA's fixed-point precision and potentially retaining model performance [23]. At the extreme, *binarisation* and *ternarisation* of model parameters via QAT can produce significant reductions in resource utilisation [24, 25, 26, 27], whilst maintaining or improving upon model performance. Quantisation-aware training can be implemented using `QKeras` [23], which allows *heterogeneous quantisation* of neural network components. In this project, quantisation was the focus of optimisation, since the parameter bitwidths enter model complexity at higher order than sparsity [28].

- **Parallelisation**: This method refers to the degree to which hardware components are reused for performing multiple operations. A maximally parallel algorithm would, for example, use one DSP slice per MAC operation.

2.4 Physics of interest

Given that methods exist for optimising neural networks for FPGA implementation, and that they have proven effective elsewhere, it is worth considering which signals at the HL-LHC could benefit from their use. An event that was of primary interest in this project is the di-Higgs decay $HH \to b\bar{b}b\bar{b}$, and the production of a top-antitop pair was used as a secondary signal. Both of these can result in topologically rich final states that CNN triggers could exploit.

2.4.1 $HH \to b\bar{b}b\bar{b}$ decay

The Higgs boson is a consequence of the EWSB mechanism in the SM, whereby a scalar field with a non-zero vacuum expectation value generates massive electroweak gauge bosons, and assigns mass to fermions via Yukawa interactions [29]. Since its discovery, the mass of the Higgs boson has been determined to a precision greater than 0.2% [30], and the focus is now shifting towards improved measurements of its couplings and of the potential. With an expected HL-LHC significance of 2.6σ [31], pair production of the Higgs boson is one avenue for realising these measurements, providing access to properties such as the self-coupling.

The tri-linear self-coupling, λ_{HHH}, describes the curvature of the Higgs potential about the minimum, and provides a strong test of BSM physics. It is the highest-order self-coupling term that can be feasibly measured at the HL-LHC [32], and its experimental parameterisation, $\kappa_\lambda = \lambda_{HHH}/\lambda_{HHH}^{SM}$, could become more tightly constrained with Phase-2 CMS [33]. To make precision measurements of the self-coupling, the sensitivity to di-Higgs production via processes such as the one shown in Fig. 3a must be maximised, a task complicated by the existence of, for example, the diagram in Fig. 3b [29]. Due to the large branching fraction of ~ 0.339 [34], the $HH \to b\bar{b}b\bar{b}$ decay is one of the most promising final states for achieving this [4]. However, this process suffers from large multi-jet QCD background, particularly in its 'resolved' topology, which needs to be suppressed with highly specific triggers.

One of the methods used to suppress the background of the $HH \to b\bar{b}b\bar{b}$ signal is b-tagging. At CMS, tracking information from the Inner Tracker and Outer Tracker subsystems is currently only available downstream from the L1T [35], therefore b-tagging can only enter selection criteria at the HLT stage. For the Phase-2 CMS upgrade, information from the Outer Tracker for $|\eta| < 2.4$ will be included as input to the L1T [4], potentially allowing b-jet identification algorithms to run at the hardware level. These do not form part of the baseline plans for the L1T upgrade, however, and preliminary studies [4] have found that track trigger rates increase significantly when considering displaced vertices, with additional resource usage that is the focus of ongoing studies. Since b-tagging at the hardware level is not currently guaranteed, it is worth considering alternative methods for triggering on $HH \to b\bar{b}b\bar{b}$, such as the use of ML algorithms.

Tracking information will be important for triggering on multi-jet events at the HL-LHC, although cuts on variables such as transverse momenta, p_T, will still remain. During Run-2, a $HH \to b\bar{b}b\bar{b}$ trigger was implemented in the GT menu with requirements on the hadronic activity, H_T, and minimum cuts on the jet p_T [12]. With the inclusion of tracking and particle flow data from the L1T, these thresholds could be lowered [4], though the trigger rate would eventually increase. The work in Ref. [4] shows that, despite the high background, sensitivity to this decay increases as the p_T threshold on the final state jets decreases. Therefore, the statistics for this process are limited by the thresholds imposed by cut-based triggers, even with the help of tracking in the L1T. This presents the opportunity for a dedicated neural network trigger that is inclusive of the low jet-p_T final states but with strong rejection of the QCD background, which could enable the kinds of precision Higgs measurements targeted by Phase-2 CMS.

2.4.2 $t\bar{t}$ production

A second signal of interest at the HL-LHC is $t\bar{t}$ production. The top-quark occupies a unique position in the SM; it has the largest Yukawa coupling to the Higgs boson, $y_t \sim 1$ [29], making it a strong probe for Higgs physics, and its short lifetime means that it is the only directly observable SM particle. The dominant production process for $t\bar{t}$ pairs is shown in Fig. 4a, whilst the dominant decay path for a top-quark is shown in Fig. 4b — the W boson can decay leptonically or to a quark-antiquark pair, the latter of which is another source of background for the $HH \to b\bar{b}b\bar{b}$ signal. For its role in Higgs physics, plus the need for more precise measurements of top-quark properties for understanding EWSB, $t\bar{t}$ production will be an important event at the HL-LHC, thus the need for efficient triggering.

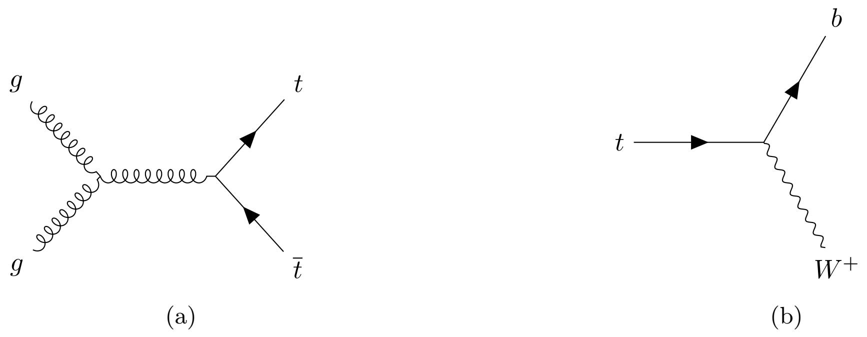

Figure 4: Feynman diagrams for (a) the dominant $t\bar{t}$ pair production process and (b) the dominant decay mode of a t-quark.

3 Experimental methods

3.1 Input data

The data used to train the neural network classifiers for triggering on the signals described in Section 2.4 originated from *Monte-Carlo* (MC) simulations of the events at the LHC. These comprise general-purpose event generators [29] and simulated detector response via CMSSW, through to the Correlator Layer-2. As shown in Fig. 5, the subdetector information will have passed through PF reconstruction and PUPPI algorithms in the Layer-1 Correlator, meaning some pileup mitigation will have been performed on the input data.

Raw event data was represented using a 120×72 2D histogram of PUPPI candidate p_T over the $\eta - \phi$ detector space, as for the PF jet-clustering algorithm described in Ref. [4], encompassing $|\eta| < 5$ and $\phi \in [0, 2\pi]$. For each event of interest, $HH \to b\bar{b}b\bar{b}$ and $t\bar{t}$, a 160000-sample dataset was created, with equal parts signal and *minimum bias* background. The LHC running parameters adopted in producing this data were $\sqrt{s} = 14$TeV and PU = 200. Figure 3.1 shows example histograms for $HH \to b\bar{b}b\bar{b}$ and minimum bias events in their raw form, and the data is noticeably sparse; for this reason, alongside the

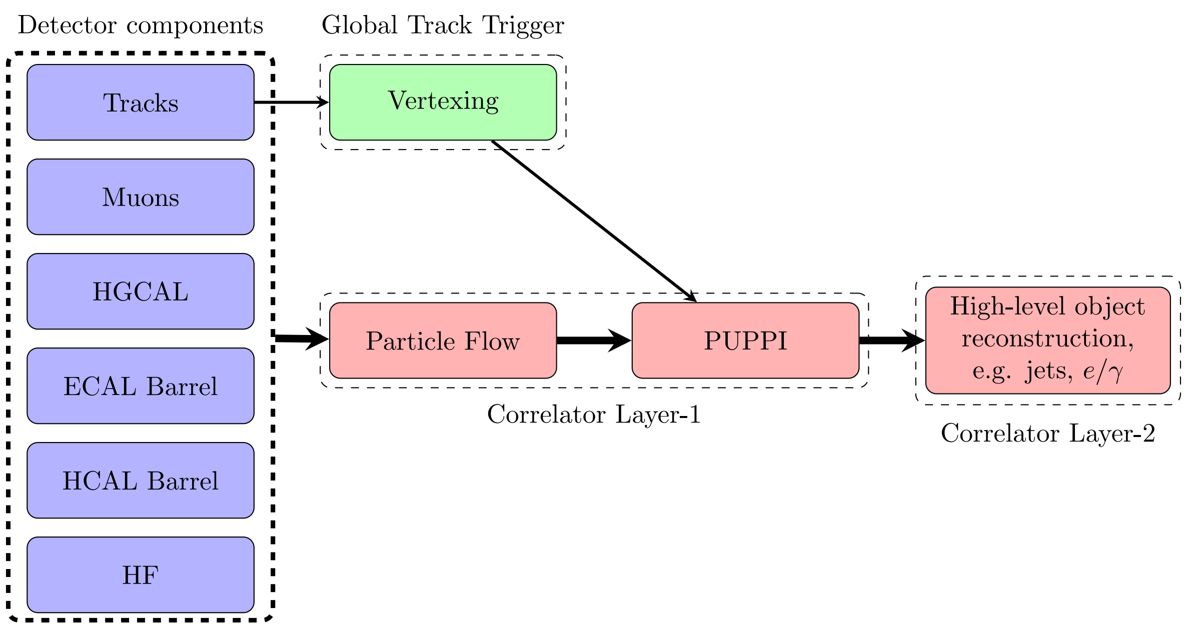

Figure 5: A high-level schematic (simplified from [4]) showing the flow of data from detector components to the Layer-2 Correlator trigger. Object reconstruction occurs using PF algorithms in the Layer-1 Correlator trigger, and PUPPI algorithms combine vertexing information with the PF candidates to suppress background for downstream processing. Correlator Layer-2 receives PUPPI candidates that can be used to form high-level physics object such as jets, and the 2D histograms of p_T used in this investigation.

fact that a larger network architecture (consuming more resources) would be necessary for processing the raw data, the input histograms were re-binned to a smaller size.

3.1.1 Pre-processing

The 120×72 input histograms described above were resized to 20×12, as shown in Fig. 3.1. The aspect ratio of the original data was maintained, and the new dimensions were restricted to divisors of the originals. While alternative sizes were considered, particularly those producing a similarly small area, this one was chosen as it resulted in firmware that functioned correctly, whilst others did not.

Scaling of the input data was also performed. Instead of scaling based on the maximum p_T value encountered across the training set, the choice was made to scale values such that the majority are in $[0, 1]$; then, it makes sense for those values to occupy as much of that range as possible, particularly when precision is reduced via quantisation. For this reason, scaling was based off of the 99.99th percentile of the non-zero p_T distribution (sample-wise).

3.2 Machine learning workflow

The `Keras` [36] Python API was employed, using the `TensorFlow` [37] backend, for producing neural networks, with the `QKeras` extension used for QAT. The data described above were partitioned to form a 3:1 train-test split, with 20% of training data used for validation. For all models, training took place over 50 epochs with binary cross-entropy loss [19], whilst using ‘early stopping’ to avoid overfitting [20].

Convolutional neural network architectures were the primary target in this project, based on the image-like input data and the availability of a more straightforward FGPA implementation (see Section 3.3) compared to MLPs. Models were designed to contain blocks comprising convolutional, pooling, batch normalisation [38] and rectified linear unit (ReLU) [39] activation components. These blocks were followed by a dense layer, and a sigmoid activation function for interpretation of the output as a scalar probability [20]. Additionally, ‘max pooling’ layers, when used, were situated before the batch normalisation, as the two commute. This reduces the total number of operations, which is important when targeting FPGA devices.

A final model architecture, shown in Fig. 7, was found via Bayesian optimisation using the `Keras Tuner` [40], for one of the signals. The aim of model selection would usually be to choose the optimal

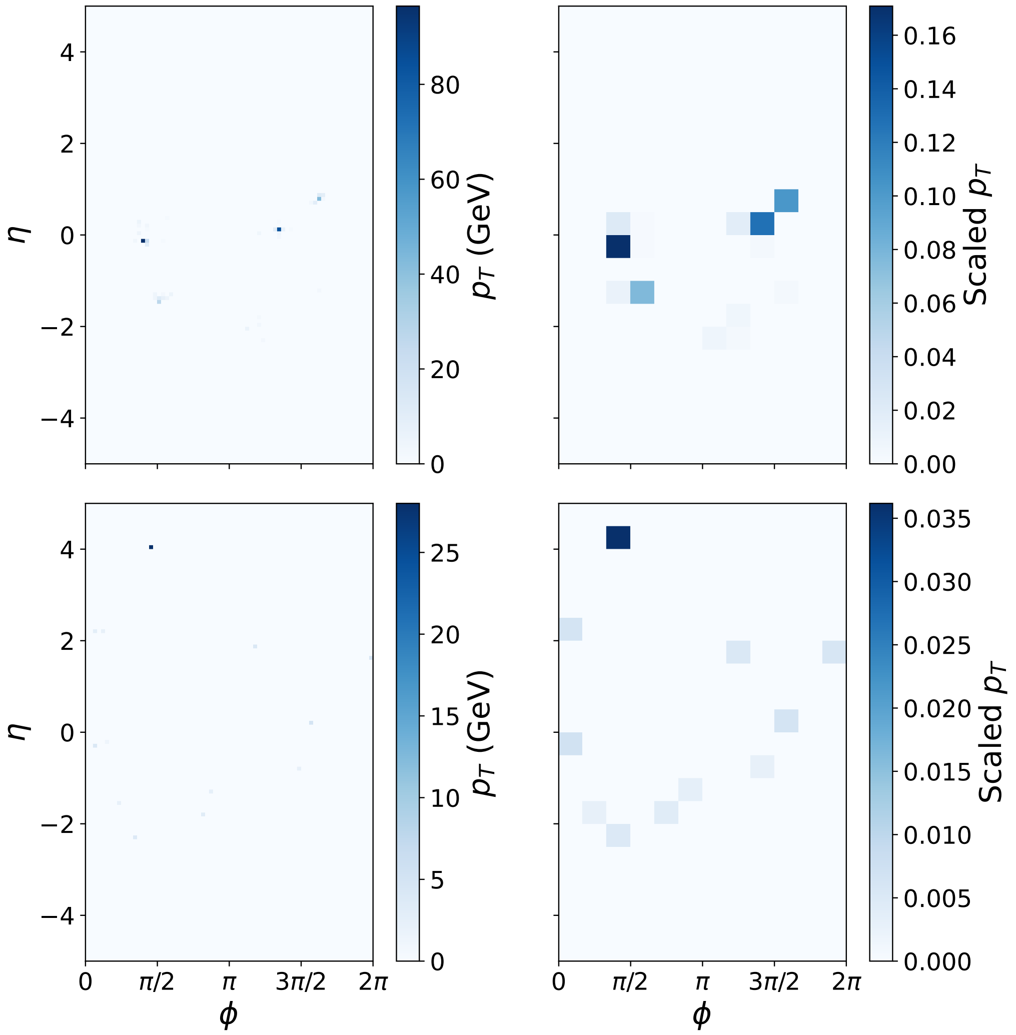

Figure 6: *Top left*: Raw event data for an example $HH \rightarrow b\bar{b}b\bar{b}$ event, with 120×72 bins in the $\eta - \phi$ detector coordinates, weighted with PUPPI p_T. *Top right*: Event data for the same $HH \rightarrow b\bar{b}b\bar{b}$ event after pre-processing, where the histogram has been resized to 20×12 bins, and the p_T re-scaled. *Bottom left*: Raw event data for a minimum bias background event. *Bottom right*: Pre-processed data for the same minimum bias event.

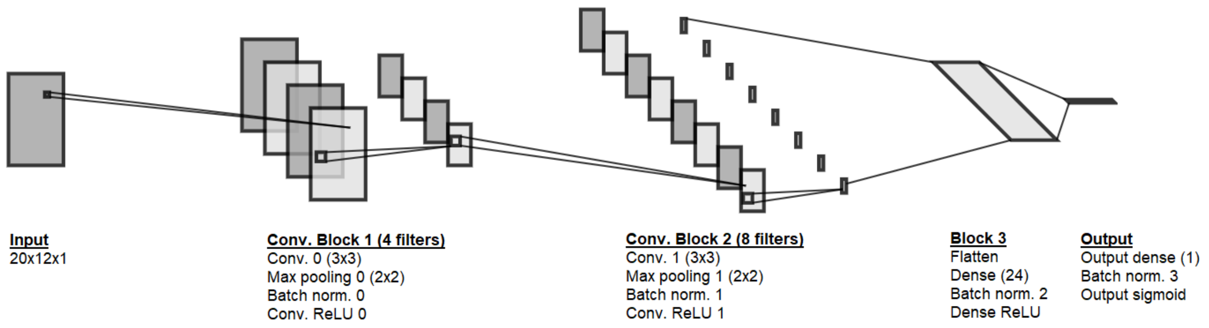

Figure 7: The convolutional neural network architecture found via Bayesian optimisation. Convolutional layers are accompanied by max pooling, batch normalisation and ReLU activations. The final dense layers also have batch normalisation appended, with a sigmoid activation on the final layer for interpretation as a probability. Filter sizes and number of nodes are shown in parentheses where relevant. Bias terms were only present for batch normalisation layers and the final ‘Output dense’ layer.

architecture individually for different use cases; however, due to limitations during deployment to an FPGA (see Section 5), once a functioning architecture was found it was also used for triggering on the other signal. At this stage a set of PTQ algorithms, for a range of data types with between 7 and 16 total bits and 6 integer bits, were created for each target signal using the tooling described in Section 3.3.

To produce QAT models, the architecture was quantised homogeneously, except for the input and output layers, and subsequently re-trained quantisation-aware via the `QKeras` API. `QKeras` offers a range of quantisers [23] when configuring a model for QAT, adding more hyperparameters to choose from. For simplicity in this work, the quantisers were restricted to `quantized_bits` [23] for dense and convolutional layers, and `quantized_relu` for the ReLU activations. A range of QAT algorithms, scanning bitwidths from 4 to 17 with 1 integer bit, were produced for each signal, with the aim of minimising FPGA resource utilisation whilst maintaining some of the floating-point classifier performance.

3.3 Trigger algorithm firmware

After producing sets of QAT and PTQ algorithms for both signals, they were converted to FPGA firmware using `hls4ml`[1] [41, 28], which transpiles the `Keras` models to pre-built HLS components. `hls4ml` provides a host of parameters for configuring these network components, and supports the conversion of `QKeras` models. The parallelisation of the firmware is controlled through a `ReuseFactor` parameter, set to 1 in this project, and fixed-precision data types can be specified on a layer-by-layer basis, applying PTQ to the model parameters. This allows for a large space of configurations; for PTQ models, data types were specified homogeneously across all layers.

Further configuration included the clock frequency of 240MHz and the target FPGA. A Xilinx Kintex Ultrascale+ KU15P[2] was the primary target, since this was the FPGA available for full deployment. Moreover, a streaming implementation of CNNs [42] was used.

In this project, Xilinx's Vivado and Vivado HLS tools[3] were used for logic synthesis and HLS, respectively. The `hls4ml` APIs were used for interfacing with Vivado HLS for running C-simulation, synthesising the models to RTL, and exporting the neural network IP-cores. The result was a representation of firmware for a standalone trigger algorithm, without any data pre-processing, that could be incorporated into a full design. The workflow described up to this point is encapsulated in Fig. 8.

3.4 Determining trigger performance

Since the trigger algorithms used here are binary classifiers, performance measures can be formed in terms of the confusion matrix elements: *true positives* (TP), *true negatives* (TN), *false positives* (FP) and *false negatives* (FN). From these quantities, the *true positive rate* (TPR) – or 'sensitivity' – and *false positive rate* (FPR) can be defined as

$$\begin{aligned} \mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\ \mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}. \end{aligned} \tag{2}$$

In the present context, the TPR corresponds to signal efficiency and 1−FPR – the *true negative rate* (TNR), or 'specificity' – to the background rejection efficiency. The evaluation of the confusion matrix is dependent on the discriminator threshold chosen for producing a binary classification; since in this work the classifier output is just the probability of a given event being a signal, an event with a signal probability greater than the threshold is associated with the signal class.

The TPR and FPR can be determined and visualised for different values of the discriminator threshold, producing a *receiver operating characteristic* (ROC) curve [43]. To make this measure more contextual, the FPR was normalised to the background event rate of $\sim$ 40MHz to give a trigger – or 'fake' – rate, under the assumption that the signal cross-section produces a negligible signal rate (up to normalisation, this is still a ROC curve, and will be denoted as such for the remainder of this report).

Using ROC curves in this way, the trade-off between acceptance of signals and the cost of accepting too much background becomes clear. In order to benchmark binary classification algorithms, it is typical

[1] v0.5.0
[2] Part: `xcku15p-ffva1760-2-e`
[3] v2019.2

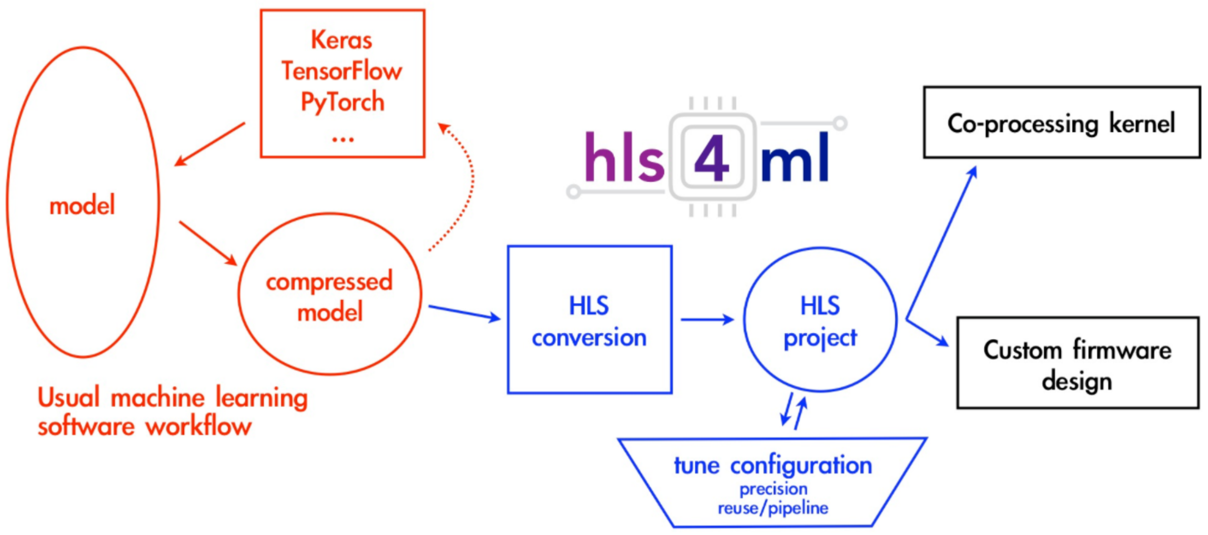

Figure 8: The general workflow for producing neural network algorithms and converting them to HLS via `hls4ml`. Taken from Ref. [41].

to use a single-valued metric such as the accuracy or the *area-under-the-ROC-curve* (AUC); however, these often consider the entire range of trigger rates. Given that low trigger rates are most important in the L1T, the sensitivity at or below a given trigger rate was used for model comparison, determined via the `SensitivityAtSpecificity` metric of the `Keras` API. Ultimately, three reference trigger rates – 10kHz, 30kHz and 50kHz – were evaluated for each trigger algorithm, using C-simulation of the HLS firmware. This allowed the selection of optimal triggers for FPGA implementation, while giving some idea as to whether allocating additional bandwidth would provide useful additional efficiency. Alongside performance measurements, resource utilisation and timing estimates were extracted from post-C-synthesis reports, with the observation that these values did not change by a noticeable amount when fully implementing the algorithms.

3.5 FPGA implementation & testing

To produce an implementable design the neural network firmware was integrated with IP from the *Extensible Modular (data) Processor* (EMP) framework[4] , which provides clocking infrastructure, I/O ports, and an `emp_payload` wrapper for the trigger algorithms. This process was carried out using *IPbus Builder*[5] (IPBB), which streamlines dependency management and interfaces with Vivado in order to implement the full design and to produce a bitstream.

The target FPGA was mounted on a Serenity carrier card, shown in Fig. 9. Using Serenity and the EMP tooling, pattern files encoding event input data were injected for FPGA inference, and the output buffers were monitored for the trigger classification result.

4 Results

4.1 Baseline triggers

The CPU performance of the full-precision CNN triggers, for both signals, is shown in Fig. 10. Since the ROC curve is normalised to the LHC event rate, the majority of the curve is not of interest, but is extended to 1MHz in this instance to illustrate the general trend. Figure 10 represents a baseline level of performance that other models could be compared against; the goal was to retain as much of the classification power as possible when moving from the floating-point model to one in which the model parameters are quantised. Of note is that the $t\bar{t}$ trigger has a lower signal efficiency than the di-Higgs trigger across all of the fake rates shown, meaning that the classifier finds it more difficult to distinguish the events containing a top-quark pair from background. This could be due to the final states being

[4]v0.3.6
[5]v1.8

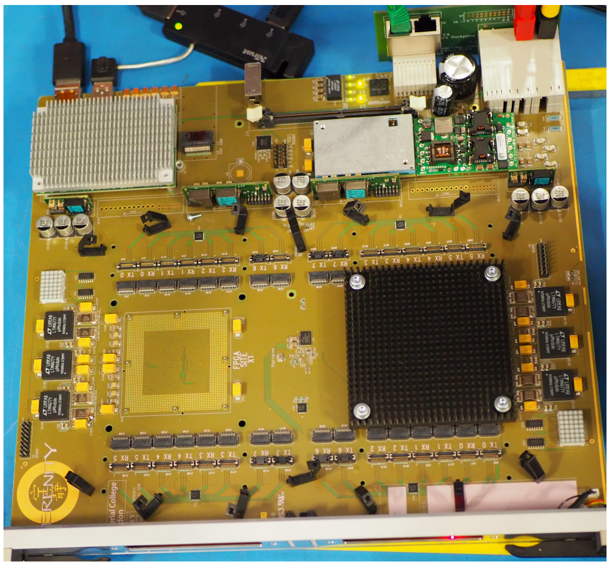

Figure 9: The Serenity carrier card used in this project. Serenity provides daughter board sites for mounting FPGAs, alongside communication links and management infrastructure.

more similar to the background or due to a larger variety of topologies. The latter of these could mean that any single type of final state is represented to a lesser degree in the training data when compared to the $HH \to b\bar{b}b\bar{b}$ decay, which predominantly results in a resolved quad-jet system.

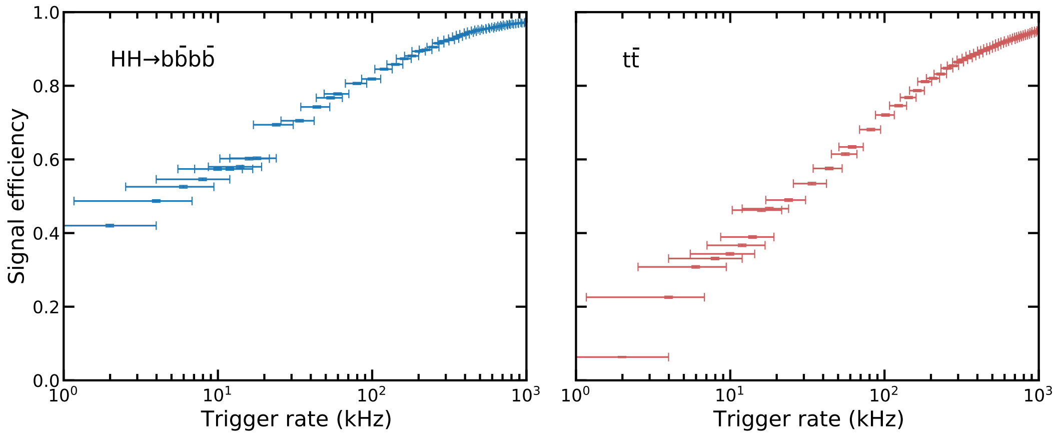

Figure 10: ROC curves showing the performance of the baseline $HH \to b\bar{b}b\bar{b}$ (left) and $t\bar{t}$ (right) binary classifiers, as evaluated on CPU hardware with floating-point precision. The errors are binomial.

4.2 Post-training-quantised triggers

An example of the effect of quantisation on the representation of model parameters is shown in Fig. 11, for the $HH \to b\bar{b}b\bar{b}$ signal. Here, the distribution of non-zero outputs, given the test data, is provided layer-by-layer for the full floating-point classifier evaluated in `Keras`, alongside those from the $\langle 16, 6\rangle$-precision PTQ model evaluated through C-simulation of a HLS model. Several layers that are present in the `Keras` model are absent in the HLS profile, since `Flatten` layers do not transform their inputs [23] and are therefore redundant in hardware, and because `BatchNormalization` layers are fused with `Dense` layers [42]. Grey boxes represent the range of values that can be represented by the fixed-point data types of the HLS model, and the change in output caused by this is evident. The changes in the distributions of layer outputs when translating to the fixed-point model are due to the increased spacing between values; when using more severe quantisation, the distributions of outputs and parameters would be expected to change even more. There is no way to tell exactly how the performance of a given model

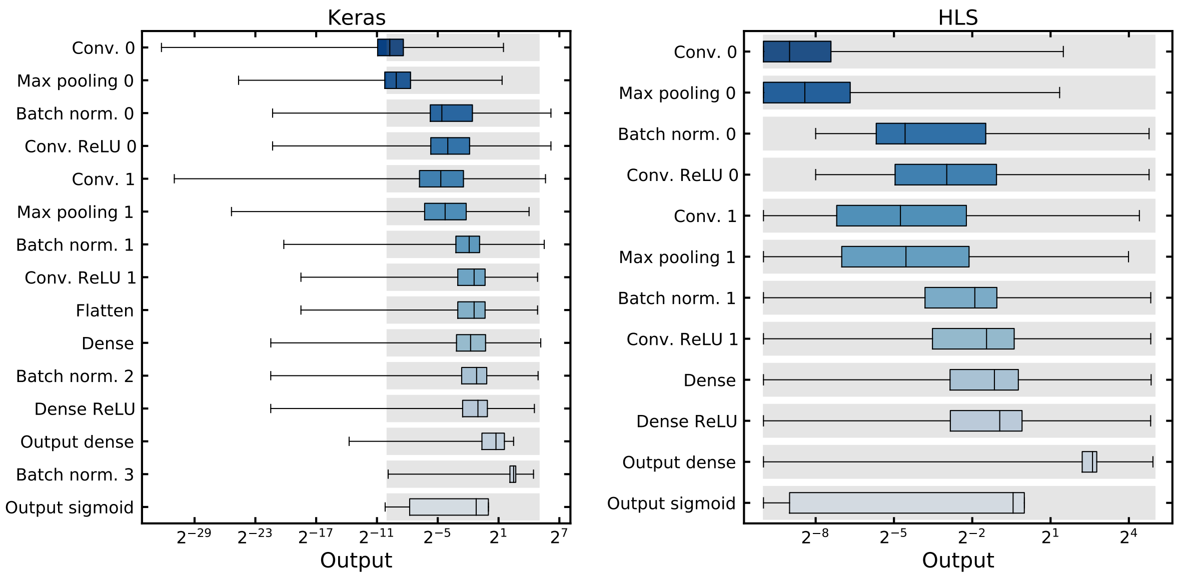

Figure 11: Distributions of the non-zero outputs of neural network components for the baseline floating-point `Keras` model (*left*) and a $\langle 16, 6\rangle$-precision PTQ model (*right*), using the test data for the $HH \rightarrow b\bar{b}b\bar{b}$ signal. The PTQ model outputs are achieved via C-simulation of the HLS model, and the grey boxes represent the range of values representable by the reduced precision. There are fewer layers for the PTQ model due to fusion of batch normalisation and dense layers, and the fact that `Flatten` layers perform no transformation of inputs.

will change under PTQ, though to retain as much of the CPU performance as possible, the data type used should provide good coverage of the floating-point values [42]. Importantly, the grey boxes in Fig. 11 should cover as much of the upper tail of the distribution as possible in order to prevent integer overflow, as this would cause a significant loss in the range of values that can be represented, when compared to underflow due to a lack of fractional precision. For this reason, 6 integer bits were maintained for all PTQ models.

For each of the signals considered, Fig. 12 shows how the performance and resource utilisation varies for a range of PTQ models with fixed-point $\langle W, 6\rangle$ data types. The signal efficiencies (solid lines) are evaluated at three reference trigger rates, alongside the baseline floating-point performance (dashed lines) for comparison. The performance of the triggers, for both signals and at all three rates, diminishes as the bitwidth decreases, as expected due to the loss of precision. For both signals the efficiency drops completely for $W \leq 8$, which corresponds to using only two bits for representing fractional parameter values. In keeping with the floating-point performance, the $t\bar{t}$ models perform worse overall. Moreover, the $HH \rightarrow b\bar{b}b\bar{b}$ trigger performance curves converge for $W \leq 10$, implying that there is a region of classifier output for which no signals exist. In fact, this artefact is a consequence of the heavily discretised output coupled with the metric used; for the models with $W \leq 10$ the vast majority of events were classified as 0, and the results in Fig. 12 then arise since the discriminator threshold can take a minimum value of 0.

The resource utilisation is shown for each of the four available FPGA resources. DSPs are utilised heavily at large bitwidths and their numbers decrease as the bitwidth does, whilst BRAM, FFs and LUTs do not exhibit such a rapid decline. This is likely due to fact that DSPs handle the multiplications within a neural network, which are abundant, alongside the quadratic dependence of the number of bit operations on parameter bitwidth [44]. The observed resource usage is practically the same for both signals over the the range of models, due to the use of identical neural network architectures — the only way in which the trends could differ is if one model's parameters converged on values that included more zeros than the other, since these zero-multiplications are optimised away during hardware implementation [42].

For comparison, Fig. 13 shows the resource utilisation of these triggers when targeting a Xilinx Virtex Ultrascale+ VU9P device, which is the FPGA of choice for much of the Phase-2 L1T upgrade [4]. These results show that, due to increased availability of resources on the VU9P, the total utilisation is reduced by a factor of ~ 3 when compared to the KU15P targeted in this work.

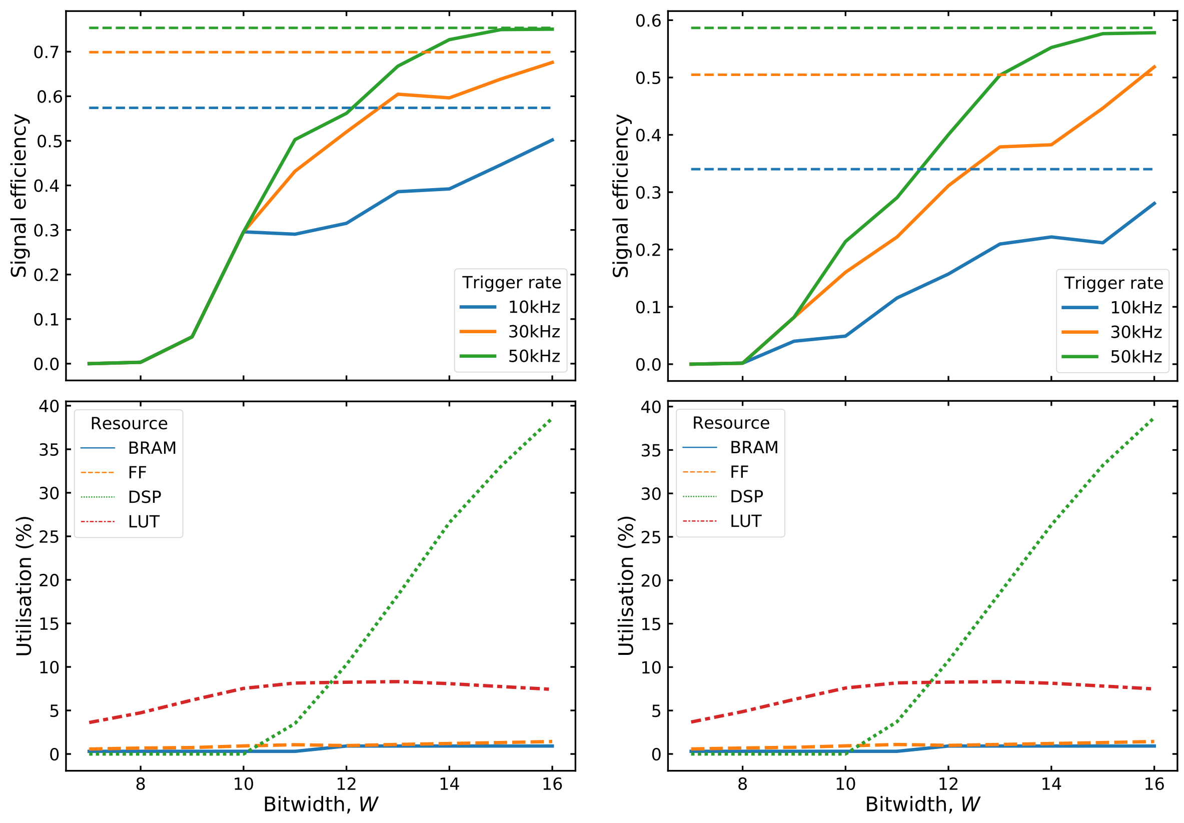

Figure 12: Performance (*top*) and resource usage (*bottom*) as a function of the bitwidth, W, for PTQ models with $\langle W,6\rangle$-precision, for the $HH \to b\bar{b}b\bar{b}$ (*left*) and $t\bar{t}$ (*right*) signals. The signal efficiencies for the PTQ triggers (*solid*) are shown for three trigger rates, as evaluated using C-simulation, along with the corresponding floating-point CPU values (*dashed*).

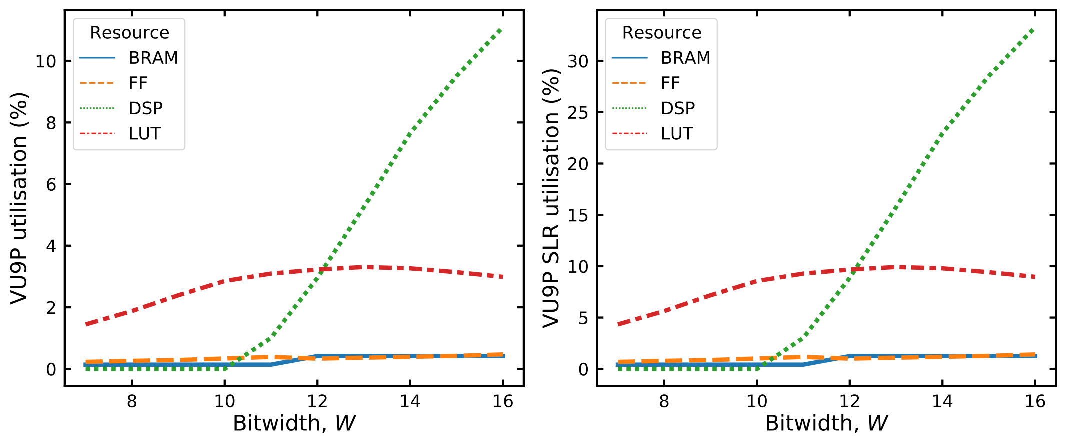

Figure 13: The estimated resource utilisation of the PTQ triggers if implemented on a VU9P FPGA. The VU9P is partitioned into three SLRs, so the utilisation is shown here as a percentage of the total available resources (*left*) and as a percentage of a single SLR region (*right*).

4.3 Quantisation-aware triggers

Figure 14 shows the effect of QAT on the parameters and layer outputs of a $\langle 16,1\rangle$ model, evaluated through `QKeras` using CPU hardware. The distributions of model parameters are shown on the left, with the suffixes 'w' and 'b' corresponding to weights and biases, respectively. Of note here is that not all layers have been quantised to the same precision; however, the bulk of computation is the

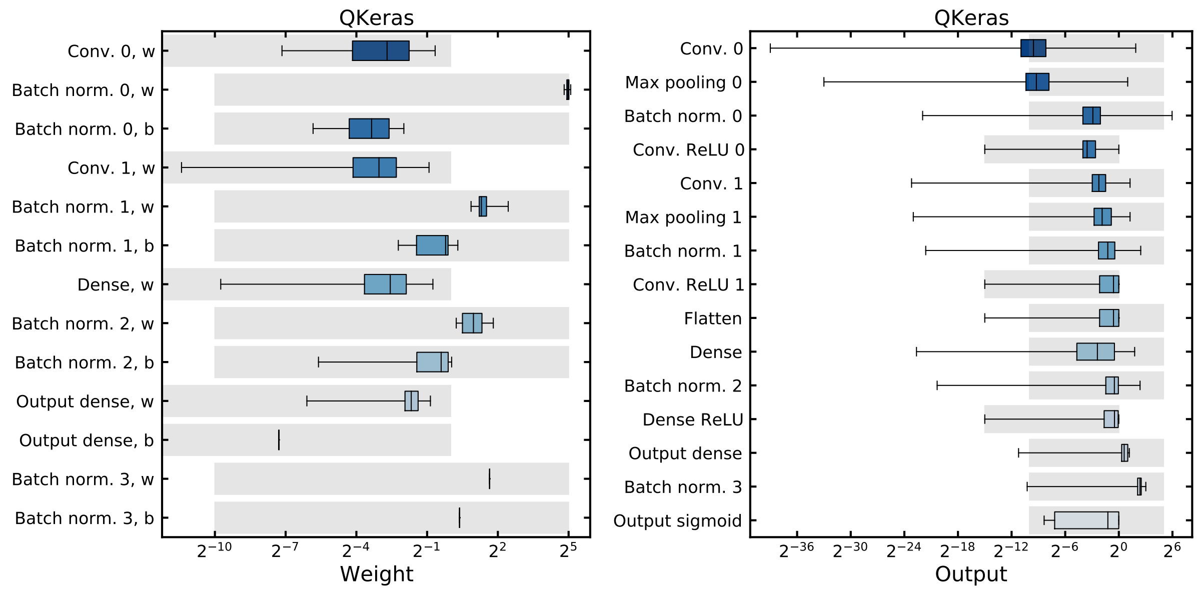

Figure 14: Example distributions of non-zero weights (*left*) and outputs (*right*) for the $HH \to b\bar{b}b\bar{b}$ QAT $\langle 16, 1\rangle$ algorithm, evaluated via `QKeras` using test data. The grey boxes represent the range of values representable by the reduced precision.

multiplication of weights and activation values, thus it is most important that these are quantised for resource optimisation. Importantly, the QAT methods have constrained the values of parameters to a range that can be represented by the fixed-point data type specified. Similarly, the activations of the convolutional and dense blocks, shown on the right of Fig. 14, are kept within the desired range where specified.

Although the output distributions change when the model is converted to HLS, the model has 'learned' about the reduced precision and performance loss is minimised in many cases, as is shown in Fig. 15. Here, the performance and resource consumption for a set of models, quantised to varying degrees, is compared as for the PTQ triggers. In this case, the signal efficiency is only provided at 10kHz for clarity, with evaluations both before (`QKeras` CPU) and after (FPGA C-simulation) conversion to HLS. For the PTQ triggers, the efficiencies decreased as the bitwidth was reduced, but this is not the general trend with the QAT models. In the case of $HH \to b\bar{b}b\bar{b}$, the signal efficiency keeps rising towards small W, with the maximum achieved by a model using 5-bit parameters. Furthermore, the performance of this trigger degrades by a small amount when converted to HLS, and has an efficiency only $\sim 2\%$ lower than the baseline floating-point model. That the performance of QAT models can increase as precision is reduced is surprising, but reinforces the empirical finding that quantisation can act as a regulariser [24]. This also indicates a possible reason for the sharp decrease at, for example, $W = 12$: there is a trade-off between sufficient bit-representation and regularisation effects. The $t\bar{t}$ results show lower efficiency overall, in keeping with the baseline and PTQ models, with a maximum at $W = 6$ as evaluated via simulation of the HLS model. The QAT triggers, for both signals, exhibit greater efficiency for low bitwidths, and notably out-perform the largest-bitwidth PTQ models at the 10kHz reference point.

On the whole, the resource utilisation has the same trend for both signals. LUT and DSP usage is, in general, higher than for the PTQ triggers, possibly due in part to the `BatchNormalization` layers not being fused with the `Dense` layers in this case. The sharp decrease in DSP usage at $W = 10$ is due to multiplications being implemented using LUTs for low-bitwidth data types [42].

4.4 FPGA implementation

A total of six triggers, based on performance and resource utilisation, were selected for deployment to an FPGA. For each signal, these consisted of the PTQ $\langle 16, 6\rangle$ and $\langle 11, 6\rangle$ models, alongside the optimal QAT algorithm as determined by the measurements described above. The signal efficiency – evaluated via C-simulation at 10kHz – and resource utilisation, alongside the latency and *initiation interval* (II) estimates after C-synthesis of the HLS firmware, are given in Table 1 for each of the implemented triggers. Signal efficiencies differ slightly here from those determined in the previous sections — they result from a

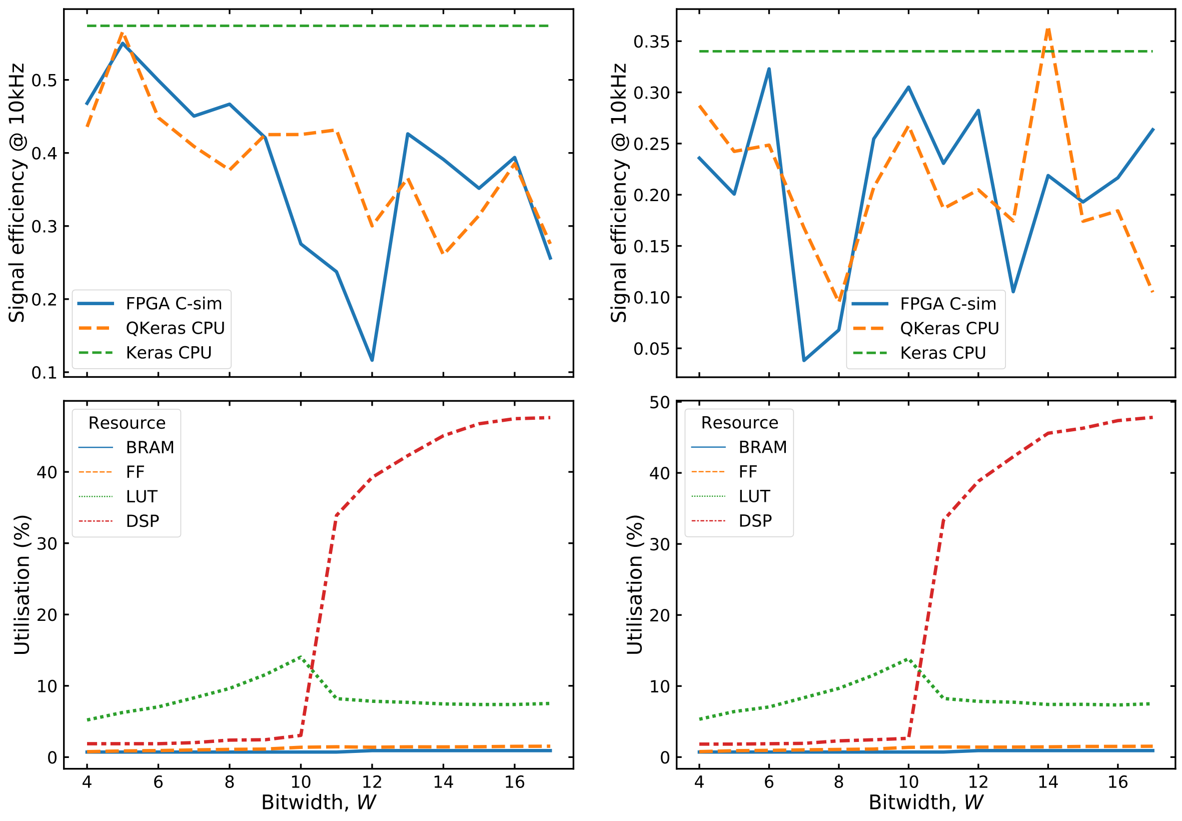

Figure 15: Performance (*top*) and resource usage (*bottom*) as a function of the bitwidth, W, for QAT models with $\langle W, 1\rangle$-precision, for the $HH \to b\bar{b}b\bar{b}$ (*left*) and $t\bar{t}$ (*right*) signals. The signal efficiencies (*solid*) are shown at a reference trigger rate of 10kHz, as evaluated using both C-simulation of HLS firmware (*solid*) and CPU inference via `QKeras` (*dashed*).

crude approximation of logarithmic behaviour for low trigger rates in the ROC curves, and interpolation to 10kHz, examples of which are shown in Fig. 16.

For the $HH \to b\bar{b}b\bar{b}$ signal, the efficiency of the optimal QAT trigger is ~ 2 times greater than was achieved in Ref. [4], evaluated at a 10kHz trigger rate. Similarly, the optimal $t\bar{t}$ algorithm has $\sim$ 2-3 times the signal efficiency achieved in Refs. [45, 46]. However, whilst the latency of each algorithm is $\mathcal{O}(1\mu\text{s})$, sufficiently low for L1T applications, the initiation intervals are too large. It is worth noting that there is an absolute minimum number of clock cycles[6] for latency and II, due to the serial streaming of input features [42]. Consequently, the II is ~ 40 times larger than required; in the L1T, triggers running at 240MHz must have II ≤ 6 clock cycles [4] in order to manage the 40MHz event rate. This does not cause a problem at the present level of implementation since events are injected to the Serenity board in isolation, without consideration of the LHC event rate. However, the triggers investigated here would not be suitable for L1T applications as they are.

As for the selection of an optimal working point with respect to signal efficiency and trigger rate, the choice of a model with sufficiently low resource usage depends upon the resources available in the L1T, and is ultimately a design choice for the Phase-2 upgrade. Amongst the triggers chosen for FPGA implementation in this project, the QAT models were considered optimal for each of the signals, given the low resource utilisation and superior performance retention compared to the PTQ versions. The resource usage of a single algorithm on the target FPGA is increased by the EMP framework, however this would represent an approximately constant overhead when implementing multiple algorithms on the same device, therefore it is not considered in detail here.

Successful deployment of these algorithms was achieved, with all of them meeting timing requirements for implementation at this stage. A representation of the QAT $\langle 5, 1\rangle$-precision $HH \to b\bar{b}b\bar{b}$ model is shown in the payload region of Fig. 17, alongside the EMP infrastructure. The hardware algorithms, and the corresponding HLS firmware descriptions, were tested using the same events, with 1000 signal samples

[6]240 for the algorithms used in this work, corresponding to the 20×12 input dimensions

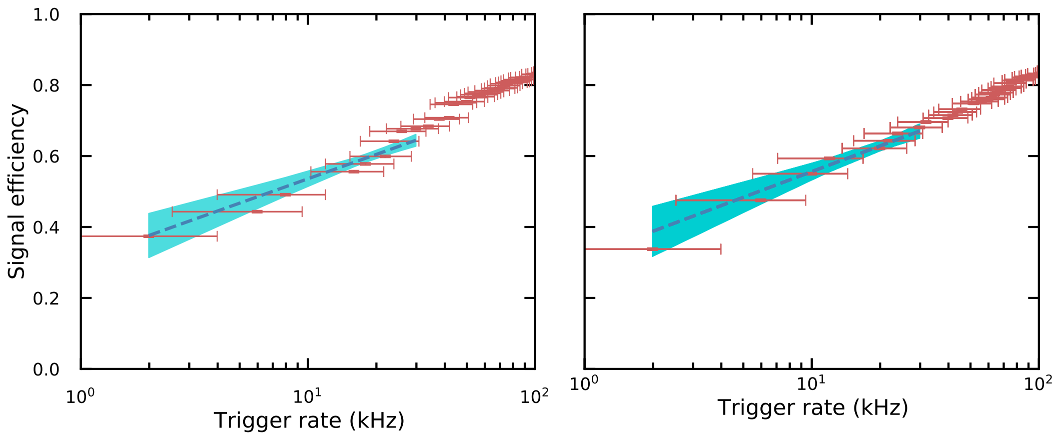

Figure 16: The performance of two of the fully implemented $HH \to b\bar{b}b\bar{b}$ triggers – PTQ $\langle 16, 6\rangle$ and QAT $\langle 5, 1\rangle$ – as evaluated via C-simulation of HLS firmware. The errors on the data points are binomial, and the low trigger rate region is approximated as logarithmic, with a bootstrapped fit for interpolation at 10kHz.

Table 1: Performance, resource utilisation, and timing measurements for the six trigger algorithms that were fully implemented in an FPGA. All values are taken from C-synth reports, and the triggers that were considered optimal in this work are highlighted in bold. The signal efficiency errors result from a bootstrapped interpolation of the ROC curve measurements.

Model	Signal efficiency @ 10kHz	BRAM (%)	FF (%)	LUT (%)	DSP (%)	Latency (ns)	II (clocks)
$HH \to b\bar{b}b\bar{b}$ PTQ $\langle 16, 6\rangle$	0.54 ± 0.02	0.91	1.43	7.42	38.57	1167	247
$HH \to b\bar{b}b\bar{b}$ PTQ $\langle 11, 6\rangle$	0.30 ± 0.02	0.30	1.06	8.15	3.51	1167	247
$HH \to b\bar{b}b\bar{b}$ QAT $\langle 5, 1\rangle$	**0.56 ± 0.02**	**0.71**	**0.87**	**6.26**	**1.88**	**1196**	**246**
$t\bar{t}$ PTQ $\langle 16, 6\rangle$	0.35 ± 0.03	0.91	1.43	7.49	38.72	1167	247
$t\bar{t}$ PTQ $\langle 11, 6\rangle$	0.11 ± 0.02	0.30	1.09	8.19	3.71	1167	247
$t\bar{t}$ QAT $\langle 6, 1\rangle$	**0.26 ± 0.04**	**0.71**	**0.90**	**6.41**	**1.83**	**1196**	**246**

and 1000 background samples. This is the main reason why performance estimates were made via C-simulation, since the number of events tested on hardware do not provide sufficient statistics in the low trigger rate region.

Although fully functional, discrepancies were observed in terms of the classification of events, when comparing the hardware results to those from C-simulation. Figure 18 (left) shows the residual counts from hardware, and the average change in prediction on the FPGA relative to C-simulation (right), for the QAT $\langle 5, 1\rangle$ $HH \to b\bar{b}b\bar{b}$ classifier. Background events were observed to be predicted closer to zero, and signals closer to one, on the average. Similar is seen for the QAT $\langle 6, 1\rangle$ $t\bar{t}$ trigger, the results for which are given in Fig. 19. Importantly, in all cases, the hardware model did not predict more background events in the top bin than during C-simulation, which would impact the trigger rate negatively. The origin of these discrepancies could be in the method used to encode events for injection to Serenity, or due to inaccuracies within Vivado HLS C-simulation.

5 Limitations & future work

One limitation encountered in this project was that the design choices were necessarily those that resulted in correctly functioning firmware[7] when deployed to the FPGA, rather than resulting from exploration of a much larger hyperparameter space, which could involve other optimisations such as sparsity. Future

[7]More specifically, incorrect behaviour was observed when monitoring the TX ports of the FPGA, ranging from missing trigger predictions to abnormal port states. This could have been due, in part, to the limited number of clock cycles (1024) over which EMP monitors the buffers, since the results from triggers operating with longer latency would not be present in the TX output.

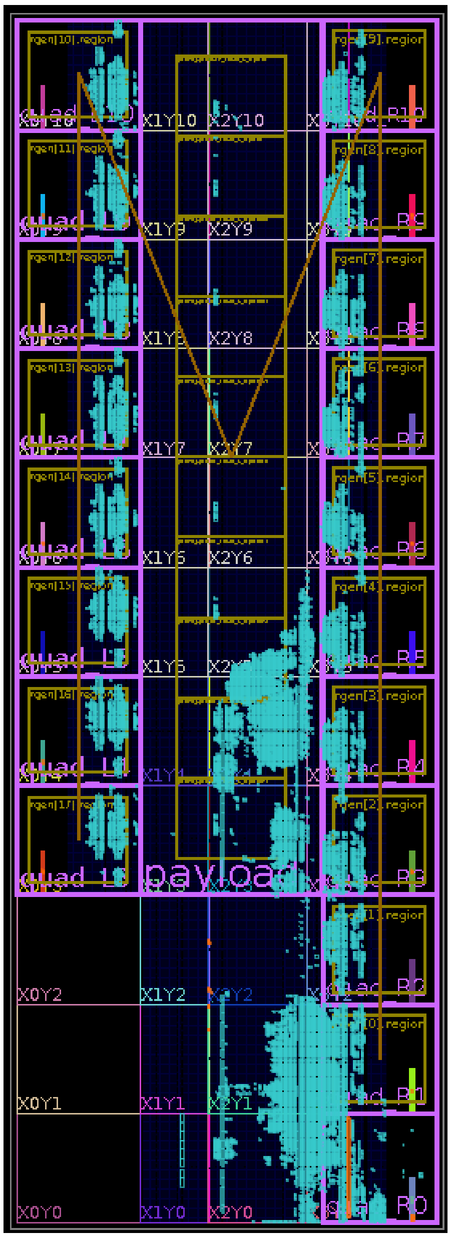

Figure 17: A graphic representation of the 5-bit QAT $HH \to b\bar{b}b\bar{b}$ trigger implemented on the target KU15P FPGA. The light blue shows areas of the device where components are used, and the trigger algorithm itself constitutes part of the area marked 'payload'. The remaining areas are components of the EMP framework, with I/O ports in the areas marked 'quad', for example.

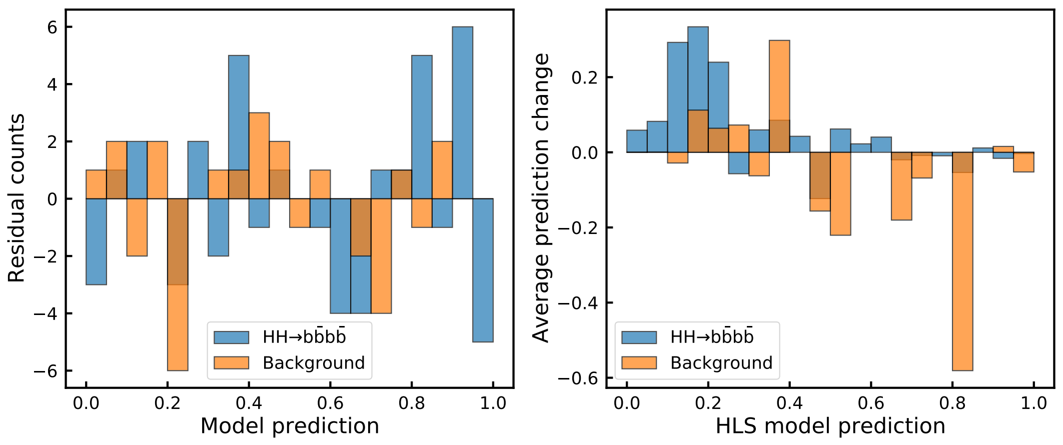

Figure 18: Hardware results for the QAT $\langle 5,1\rangle$ $HH \to b\bar{b}b\bar{b}$ trigger. *Left*: The excess counts in each bin when events were processed by FPGA hardware compared to those recorded using the same events but via C-simulation of HLS. *Right*: The average change in event classification on hardware, per bin, relative to the prediction achieved via C-simulation of HLS.

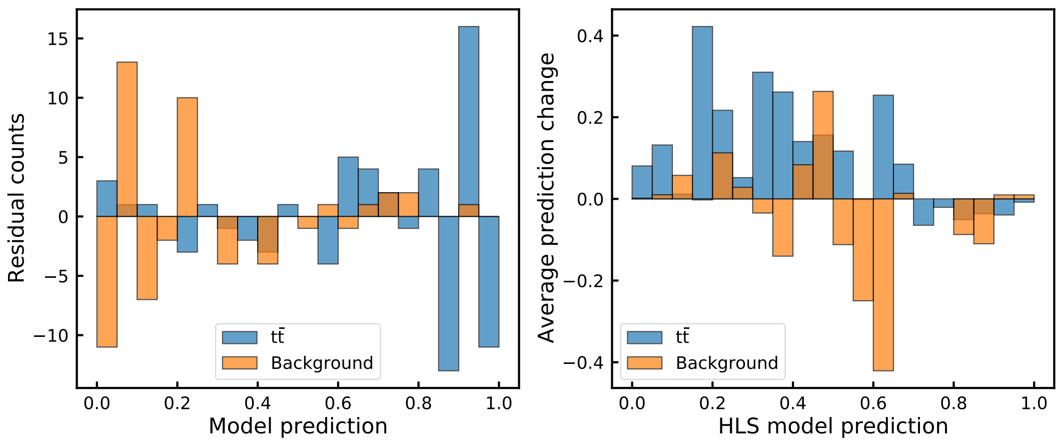

Figure 19: Hardware results for the QAT $\langle 6,1\rangle$ $t\bar{t}$ trigger. *Left*: The excess counts in each bin when events were processed by FPGA hardware compared to those recorded using the same events but via C-simulation of HLS. *Right*: The average change in event classification on hardware, per bin, relative to the prediction achieved via C-simulation of HLS.

work could explore implementation restrictions, beyond the resource and timing constraints, and therefore a larger range of triggers.

Another potential drawback of this work is that the neural networks were trained using supervised methods; when searching for rare events such as the 'boosted' [47] $HH \to b\bar{b}b\bar{b}$ decay, which may not be represented in the training data, unsupervised models such as autoencoders could be beneficial. An autoencoder developed in Ref. [4] has been shown to be sensitive to events like $HH \to b\bar{b}b\bar{b}$ and to have minimal resource usage when quantised post-training. The use of `QKeras` to build a QAT autoencoder could therefore provide performance benefits and further resource reduction, as seen in this report.

Quantisation, especially when considering the full range of methods offered via `QKeras` [23], produces a large space of hyperparameters, making it challenging to find an optimal algorithm without considering a reduced space as in this work. `QKeras` offers an extension to the `Keras Tuner` – `AutoQKeras` – which handles the optimisation problem created by the trade-off between performance and resource consumption. This has been shown to produce neural networks with greater performance, and lower resource utilisation, than homogeneously quantised models [23, 42]. This tooling could be used to make further improvements to architectures like those studied in this work.

Furthermore, the aim of machine learning triggers for use in the L1T is to improve upon traditional cut-based methods for signal discrimination. A limitation of this work is that the performance measures used leave the question: *what have the machine learning algorithms actually learned?* Gaining insight into this could better characterise the triggers and make clearer the events for which they have improved selectivity beyond traditional algorithms. A possible approach to this problem would be to determine the sensitivity and specificity as a function of high-level quantities such as jet p_T, which is particularly important for $HH \rightarrow b\bar{b}b\bar{b}$ measurements, as described in Section 2.4.1. Beyond this, additions to the methodology of this project such as k-fold cross-validation or data augmentation, alongside performance evaluation using FPGA data as opposed to simulation, would produce more robust results.

Finally, the most obvious limitation of this project is the extremely long initiation intervals of the implemented triggers. This is inherent in the stream-based CNN implementation via `hls4ml`, and could only be reduced by using a much higher FPGA clock frequency or extremely small input images, both of which are not viable. Possible solutions to this problem include: *time-multiplexing* of a given trigger, which would dramatically increase total resource usage, but with percent-level consumption on a VU9P is at least feasible; a different firmware implementation of CNNs, which would greatly restrict the size of the model architecture [42]; or, other types of neural network, such as the MLP.

6 Conclusion

Convolutional neural networks were produced for selection of two physics events – the decay $HH \rightarrow b\bar{b}b\bar{b}$ and $t\bar{t}$ production – that are of interest for the CMS Phase-2 physics program at the HL-LHC. Using quantisation-aware training, low-resource algorithms were produced with greater than 50% sensitivity to the $HH \rightarrow b\bar{b}b\bar{b}$ signal for a 10kHz fake rate, retaining most of the floating-point baseline performance. These models were successfully deployed to an FPGA using a stream-based CNN implementation via `hls4ml`, with results showing slight discrepancies when compared to firmware simulation. Furthermore, whilst the triggers had latencies $\mathcal{O}(1\mu\text{s})$, suitable for L1T application, their initiation intervals were found to be too long. Finally, some suggestions were made for investigating a wider range of trigger algorithms, and further optimising performance and resource utilisation.

References

[1] G Apollinari et al. "High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report". Technical Report CERN-2015-005, CERN, Geneva, 2015.

[2] CMS Collaboration. "The CMS experiment at the CERN LHC". *JINST*, 3(08):S08004–S08004, 2008.

[3] CMS Collaboration. "Technical Proposal for the Phase-II Upgrade of the CMS Detector". Technical Report CMS-TDR-15-02, CERN, Geneva, 2015.

[4] CMS Collaboration. "The Phase-2 Upgrade of the CMS Level-1 Trigger". Technical Report CMS-TDR-021, CERN, Geneva, 2020.

[5] ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". *JINST*, 3:S08003, 2008.

[6] CMS Collaboration. "Pileup mitigation at CMS in 13 TeV data". *JINST*, 15(9), 2020.

[7] CMS Collaboration. "Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC". *Phys. Lett. B*, 716:30–61, 2012.

[8] CMS Collaboration and LHCb Collaboration. "Observation of the rare $B_s^0 \to \mu^+\mu^-$ decay from the combined analysis of CMS and LHCb data". *Nature*, 522:68–72, 2015.

[9] CMS Collaboration. "$B \to \mu^+\mu^-$ rare decays at CMS". Technical Report CMS-CR-2018-137, CERN, Geneva, 2018.

[10] CMS Collaboration. "Study of $\mathrm{W}^\pm\mathrm{W}^\pm$ production via vector boson scattering at the HL-LHC with the upgraded CMS detector". Technical Report CMS-PAS-FTR-18-005, CERN, Geneva, 2018.

[11] CMS Collaboration. "Measurement of rare $B \to \mu^+\mu^-$ decays with the Phase-2 upgraded CMS detector at the HL-LHC". Technical Report CMS-PAS-FTR-18-013, CERN, Geneva, 2018.

[12] CMS Collaboration. "Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s} =$ 13 TeV". *JINST*, 15:P10017, 2020.

[13] Darin Acosta et al. "Boosted Decision Trees in the Level-1 Muon Endcap Trigger at CMS". *Journal of Physics: Conference Series*, 1085:042042, 2018.

[14] Tommaso Diotalevi et al. "Development of Machine Learning based muon trigger algorithms for the Phase-2 upgrade of the CMS detector". In *PoS – LHCP2018*, volume 321, page 092, 2018.

[15] Alexander Radovic et al. "Machine learning at the energy and intensity frontiers of particle physics". *Nature*, 560:41–48, 2018.

[16] T. Q. Nguyen et al. "Topology Classification with Deep Learning to Improve Real-Time Event Selection at the LHC". *Computing and Software for Big Science*, 3(12), 2019.

[17] Celia Fernández Madrazo et al. "Application of a Convolutional Neural Network for image classification for the analysis of collisions in High Energy Physics". *EPJ Web Conf.*, 214:06017, 2019.

[18] ATLAS Collaboration. "Fast and resource-efficient Deep Neural Network on FPGA for the Phase-II Level-0 muon barrel trigger of the ATLAS experiment". Technical Report ATL-DAQ-PROC-2020-008, CERN, Geneva, 2020.

[19] Christopher M Bishop. *"Pattern recognition and machine learning"*. Springer, 2006.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *"Deep Learning"*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[21] Davis Blalock et al. "What is the State of Neural Network Pruning?". 2020. Preprint arXiv:2003.03033.

[22] National Instruments. `https://www.ni.com/en-gb/innovations/white-papers/08/fpga-fundamentals.html`, 2020.

[23] Claudionor N. Coelho et al. “Automatic deep heterogeneous quantization of Deep Neural Networks for ultra low-area, low-latency inference on the edge at particle colliders”. 2020. Preprint arXiv:2006.10159.

[24] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. “BinaryConnect: Training Deep Neural Networks with binary weights during propagations”. In *Advances in Neural Information Processing Systems*, volume 28, pages 3123–3131. Curran Associates, Inc., 2015.

[25] Jennifer Ngadiuba et al. “Compressing deep neural networks on FPGAs to binary and ternary precision with `hls4ml`”. *Mach. Learn.: Sci. Technol.*, 2(1):015001, 2021.

[26] Xiaofan Lin, Cong Zhao, and Wei Pan. “Towards Accurate Binary Convolutional Neural Network”. In *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[27] Mohammad Rastegari et al. “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks”. In *Computer Vision – ECCV 2016*, pages 525–542. Springer International Publishing, 2016.

[28] Farah Fahim et al. “`hls4ml`: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices”. 2021. Preprint arXiv:2103.05579.

[29] M. Tanabashi et al. “Review of Particle Physics”. *Phys. Rev. D*, 98:030001, 2018.

[30] CMS Collaboration. “A measurement of the Higgs boson mass in the diphoton decay channel”. *Phys. Lett. B*, 805:135425, 2020.

[31] Andrea Dainese et al. “Report on the Physics at the HL-LHC, and Perspectives for the HE-LHC”. Technical Report CERN-2019-007, CERN, Geneva, 2019.

[32] J. Baglio et al. “The measurement of the Higgs self-coupling at the LHC: theoretical status”. *JHEP*, 04:151, 2013.

[33] M. Cepeda et al. “Report from Working Group 2: Higgs Physics at the HL-LHC and HE-LHC”. *CERN Yellow Rep. Monogr.*, 7:221–584, 2018.

[34] CMS Collaboration. “Search for Non-Resonant Higgs Pair-Production in the $b\bar{b}b\bar{b}$ Final State with the CMS detector”. Technical Report CMS-PAS-HIG-17-017, CERN, Geneva, 2018.

[35] CMS Collaboration. “Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV”. *JINST*, 13:P05011. 114 p, 2017.

[36] François Chollet et al. “Keras”. `https://keras.io`, 2015.

[37] Martín Abadi et al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems”. `https://www.tensorflow.org/`, 2015.

[38] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[39] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML’10, page 807–814, USA, 2010. Omnipress.

[40] T. O’Malley et al. “Keras Tuner”. `https://github.com/keras-team/keras-tuner`, 2019.

[41] Javier Duarte et al. “Fast inference of deep neural networks in FPGAs for particle physics”. *JINST*, 13, 2018.

[42] Thea Aarrestad et al. “Fast convolutional neural networks on FPGAs with `hls4ml`”. 2021. Preprint arXiv:2101.05108.

[43] Tom Fawcett. “An Introduction to ROC Analysis”. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[44] Chaim Baskin et al. “UNIQ: Uniform Noise Injection for Non-Uniform Quantization of Neural Networks”. *ACM Trans. Comput. Syst.*, 37(1–4), 2021.

[45] Frederic Shirley. “Novel Trigger Algorithms for CMS: Using Machine Learning to Maximise Particle Detection in the Level-1 Trigger”. Master’s thesis, University of Bristol, 2020.

[46] Kai Amin-Wetzel. “Developing a Neural CMS Trigger”. Master’s thesis, University of Bristol, 2020.

[47] CMS Collaboration. “Combination of Searches for Higgs Boson Pair Production in Proton-Proton Collisions at $\sqrt{s} = 13$ TeV”. *Phys. Rev. Lett.*, 122:121803, 2019.

Certification of ownership

Project Report presented as part of, and in accordance with, the requirements for the Final Degree of MSci at the University of Bristol, Faculty of Science.

Author: Kyle Pidgeon
Title: Neural network triggers for the CMS detector at the HL-LHC
Date of submission: 11/05/2021

Signed:
K. Pidgeon

Full name: Kyle Pidgeon
Date: 11/05/2021