Computing and Software for Big Science manuscript No.
(will be inserted by the editor)

Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters

Georges Aad · Anne-Sophie Berthold · Thomas Calvet · Nemer Chiedde · Etienne Marie Fortin · Nick Fritzsche · Rainer Hentges · Lauri Antti Olavi Laatu · Emmanuel Monnier · Arno Straessner · Johann Christoph Voigt

Received: date / Accepted: date

Abstract The ATLAS experiment at the Large Hadron Collider (LHC) is operated at CERN and measures proton-proton collisions at multi-TeV energies with a repetition frequency of 40 MHz. Within the Phase-II upgrade of the LHC, the readout electronics of the Liquid-Argon Calorimeters of ATLAS are being prepared for high luminosity operation expecting a pile-up of up to 200 simultaneous proton-proton interactions. Moreover, the calorimeter signals of up to 25 subsequent collisions are overlapping, which increases the difficulty of energy reconstruction by the calorimeter detector. Real-time processing of digitized pulses sampled at 40 MHz is thus performed using Field Programmable Gate Arrays (FPGAs).

To cope with the signal pile-up, new machine learning approaches are explored: convolutional and recurrent neural networks outperform the optimal signal filter currently used, both in assignment of the reconstructed energy to the correct proton bunch crossing and in energy resolution. Since the implementation of the neural networks targets a FPGA, the number of parameters and the mathematical operations need to be well controlled. The trained neural network structures are converted into FPGA firmware using automated VHDL implementations and high-level synthesis tools.

Very good agreement between neural network implementations in FPGA and software based calculations is observed. The FPGA resource usage, the latency, and the operation frequency are analysed. Latest performance results and experience with prototype implementations are reported.

G. Aad · T. Calvet · N. Chiedde · E.M. Fortin · L.A.O. Laatu
CPPM, Aix-Marseille Université, CNRS/IN2P3, Marseille, France,

A.-S. Berthold · N. Fritzsche · R. Hentges · A. Straessner · J.C. Voigt
Institut für Kern- und Teilchenphysik, Technische Universität Dresden, Dresden, Germany
E-mail: Arno.Straessner@cern.ch

Keywords machine learning · convolutional neural network · recurrent neural network · FPGA · real-time processing · high-energy physics

1 Introduction

The ATLAS detector [1] is installed at the Large Hadron Collider [2] (LHC) in order to detect the particles produced in high-energy proton-proton collisions, and measure their properties. The proton bunches collide every 25 ns corresponding to a frequency of 40 MHz. During the future high-luminosity phase of LHC (HL-LHC) the machine is expected to produce instantaneous luminosities of $5-7\times10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ starting with Run-4 in 2027. This corresponds to 140-200 simultaneous proton-proton interactions. The Liquid-Argon (LAr) Calorimeters of ATLAS mainly measure the energy of electromagnetic showers of photons, electrons and positrons using their ionisation signal. The LAr Calorimeters are challenged by the large in-time pile-up and because up to 25 signal pulses created in subsequent LHC bunch crossings (BC) can overlap leading to out-of-time pile-up. Moreover, a new trigger scheme is foreseen [3] which allows the selection of collision events in subsequent bunch crossings. Thus, an assignment of the reconstructed energy to the correct bunch crossing with best possible energy resolution is necessary for each of the 182,000 calorimeter cells. For the trigger data path, a latency of about 150 ns is allocated to the reconstruction of the energy, based on a preliminary analysis of the full data processing chain [3,4].

Within the Phase-II upgrade of the LAr Calorimeter electronics [4], processing of the LAr pulses sampled at 40 MHz is foreseen using Field Programmable Gate Arrays (FPGAs). In the current design options, 384 or 512 LAr calorimeter cells shall be processed by one Intel Stratix-10 FPGA [5], which corresponds to the data measured by three or four

so-called Front-End Boards (FEBs), respectively. The full system needs to capture the data of 1524 FEBs in total.

To meet the challenging task of real-time energy reconstruction new machine learning methods are explored. The application of Artificial Neural Networks (ANNs) on FPGAs, however, is constrained by the limited digital signal processing (DSP) resources, logic and memory available in the FPGA devices. This, in turn, limits the number and type of mathematical operations that can be used by the machine learning application. In addition, software tools for converting trained neural networks into FPGA firmware are needed. In the following, first results and experience aiming at real-time reconstruction of LAr calorimeter energies are presented.

2 LAr cell energy reconstruction by Artificial Neural Networks

2.1 Simulation of LAr pulse sequences and legacy energy reconstruction

The first step in the development of the FPGA-based ANNs is the training of the networks on simulated data sequences. The AREUS [6] tool is used to convert the series of energy deposits in the LAr Calorimeter cells into a sequence of overlaid and digitized pulses taking into account analog and digital electronics noise. The software also allows a simulation of LHC bunch patterns, i.e. regular interruptions in the series of proton-proton collisions. An example sequence for one cell in the barrel section (EMB) of the electromagnetic LAr calorimeter, which is selected for the study presented here, is displayed in the top row of figure 1 for a mean number of pile-up events, $\langle\mu\rangle$, of 140.

The current readout electronics of the LAr Calorimeters applies an optimal filter [7] (OF) to determine the energy in each cell. By linear combination of up to five digitized pulse samples electronic noise and signal pile-up are suppressed. The coefficients of the OF are determined using the analog pulse shape and the total noise auto-correlation. In order to further identify true energy deposits and assign them to a certain LHC bunch crossing (BC), a peak finder is applied to the output sequence of the OF by selecting the maximum value in each group of three consecutive BCs. The OF results are used to compare with the neural network solutions.

Supervised learning is applied during the network training. The true energies deposited serve as target values which are also indicated in the top row of figure 1. The network training utilizes the Keras [8] API to the TensorFlow [9] platform.

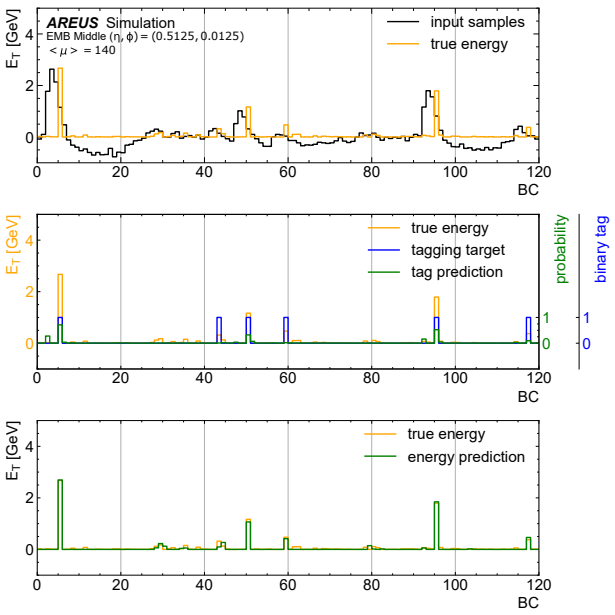

Fig. 1 Top: Sample sequence (black) of an EMB middle-layer cell located at a pseudo-rapidity, η, of 0.5125 and an azimuthal angle, ϕ, of of 0.0125 within the ATLAS coordinate system. The sequence is simulated by AREUS, together with the true transverse energy deposits (yellow), at $\langle\mu\rangle = 140$ as a function of the bunch crossing (BC) counter. The true deposits are shifted by five BC to improve the plot visibility. Middle: The Convolutional Neural Network (CNN) for pulse tagging provides a hit probability (green) for each BC. Its training is based on a binary input sequence (blue) with values of unity for energy deposits 3σ above noise threshold. Bottom: The transverse energy reconstruction CNN makes its predictions (green) based on the probability of the tagging layer and the input samples.

2.2 Convolutional Neural Networks

Alternatively to the OF method, Convolutional Neural Networks (CNNs) [10] are developed. The networks analyse the input data sequence in a sliding-window approach. Linear combinations of data values in a given window of subsequent bunch-crossings, also called receptive field, are fed into parallel layers of artificial neurons, called feature maps. Different neuron activation functions are used. The maps are combined to a multi-layer structure.

The underlying resource restrictions of the FPGA are central when developing CNNs for the LAr energy reconstruction. The large number of cells treated by one FPGA allows at most a few hundred multiplier-accumulator units, respectively parameters, per network. A two-layered network architecture was found to yield the best performance. The first "tagging" network structure identifies significant energy deposits above a threshold of 3σ of the electronic noise, corresponding to 240 MeV. Together with the sample sequence a detection probability is passed to a second structure which is trained to reconstruct the deposited energy in

each calorimeter cell. An example of the architecture is presented in figure 2.

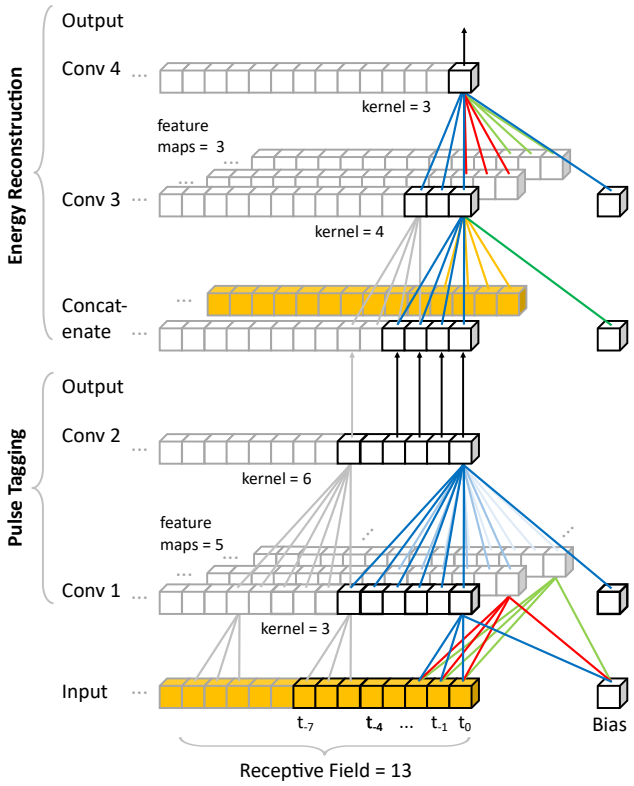

Fig. 2 Architecture of an Artificial Neural Network (ANN) with four convolutional layers. The dataflow goes from bottom to top. The input sequence is first processed by the tagging part of the network in the bottom part of the figure. After a concatenation layer, the tag output and the input sequence are processed by the transverse energy reconstruction part of the ANN. The total receptive field of this network incorporates 13 bunch crossings.

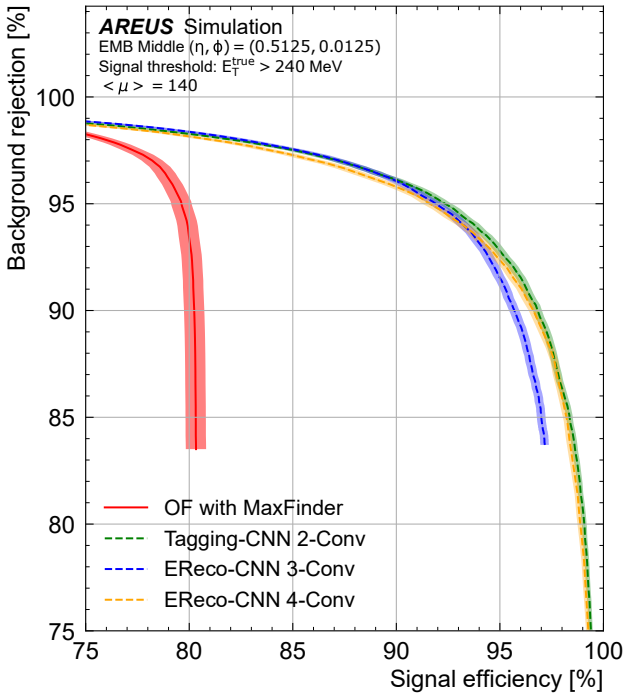

Fig. 3 Signal efficiency and background rejection ROC curves of the two presented Artificial Neural Networks (yellow, purple) and their tagging part (green), compared to the Optimal Filtering (OF) with a maximum finder (red). Signal refers to deposits with $E_{\mathrm{T}}^{\mathrm{true}}$ above 240 MeV (3σ above noise threshold), background those below. Efficiencies are calculated for an EMB Middle LAr cell ($\eta = 0.5125$ and $\phi = 0.0125$) simulated with AREUS assuming $\langle\mu\rangle = 140$. Approaching the upper right corner of the plot indicates signal efficiencies of 100% and a background rejection of 100% and would therefore be optimal. For better visibility, the results are shown only in the range above 75%. Filled bands represent the statistical uncertainty.

An improvement was achieved by pre-training the tagging part of the network before embedding it into the entire architecture. Middle and bottom rows of figure 1 display the processing steps for both the tagging and the energy reconstruction parts.

The ability of the tagging CNN to detect true signals and reject background is illustrated in figure 3 for a tagging network with 2 convolutional layers (”2-Conv”) and kernel sizes of 3 and 6. The signal efficiency and background rejection are compared to the performance of the OF algorithm and a subsequent maximum finder. The receiver operating characteristic (ROC) curves indicate the performance when varying the tag probability threshold, respectively the threshold on the energy calculated by the OF. The OF achieves a maximum signal efficiency of about 80%, while the tagging CNN reaches efficiencies well above 90%.

In the following, two CNNs named ”3-Conv” and ”4-Conv” will be presented. While their tagging part has the same configuration, the energy reconstruction consists of one, respectively two, convolutional layers, as listed in table 1. Layers, kernel sizes, dilation rate and the number of feature maps per layer were chosen such that the performance regarding signal detection and energy reconstruction under conditions like the occurrence of signals in quick succession and realistic LHC bunch train patterns was best. Dilation, i.e. a regular removal of dense connections between network nodes, would allow an enlargement of the receptive field without increase in number of network parameters. However, not applying dilation was found optimal during hyper-parameter optimisation. The resource restrictions are well satisfied for both networks.

A sigmoid function is used as activation function for the tagging layers because it obtained best results for the binary tag classification. A Rectified Linear Unit (ReLU) activation function was chosen for the energy network motivated by the ReLU property that negative input values are set to zero and only positive values are forwarded. The ROC curves for both complete CNN networks are shown in figure 3, again compared to the OF and to the tagging network only. The

maximum efficiencies are only slightly reduced when the energy calculation is included and both CNNs clearly outperform the OF algorithm.

Table 1 CNN configurations with one and two energy reconstruction layers and identical tagging layer.

	"3-Conv"			"4-Conv"			
	Tagging		Energy Reconstruction	Tagging		Energy Reconstruction	
Layer index	1	2	3	1	2	3	4
Kernel Size	3	6	21	3	6	4	3
Dilation Rate	1	1	1	1	1	1	1
Feature Maps	5	1	1	5	1	3	1
Activation Function	sigmoid		ReLU	sigmoid		ReLU	
Number of Parameters	51		43	51		37	
Receptive Field	28			13			

2.3 Recurrent Neural Networks

Recurrent Neural Network (RNN) algorithms are designed for the inference of time series and extraction of the underlying parameters. They are natural candidates for the inference of deposited energies from time-ordered digitized LAr signals. Two RNN architectures are considered: Vanilla-RNN [11] and Long Short-Term Memory (LSTM) [12].

LSTM based algorithms: LSTM based networks demonstrate upmost management of information through long sequences, allowing the constraint of long-lived dependencies in data. LSTM cells are composed of four internal neural networks, three learn to open and close access to the data flow through time, the last acting directly on the data to extract the desired features at a given time. However, their complexity scales rapidly with the dimension of the internal networks, while the application of intelligent algorithms in the LAr calorimeter read-out system sets tight limits on the network size. In order to limit the parameter count to a few hundred, only one layer of LSTM cells, with 10 internal dimensions, is allowed. A decoder, consisting of a network with a single neuron and ReLU activation, is placed at the exit of the LSTM to concatenate the output in a single energy measurement.

Two LSTM based networks for real-time energy measurements are presented. The single-cell design derives from a many-to-many RNN evaluation, and is illustrated in figure 4. At each bunch crossing, a LSTM cell analyses the LAr signal amplitude and the output of the previous cell to predict an energy. The same operation with the same LSTM object is repeated until the end of data. To allow the RNN to accumulate enough information a delay of five bunch crossings is imposed in the training process. This delay also avoids the RNN to learn from yet to happen collisions in the training phase. The second design uses a sliding-window algorithm and is illustrated in figure 5. At each bunch crossing a LSTM network is instantiated. This network is trained as a many-to-one RNN targeting an energy prediction with five ADC samples as input. The target energy corresponds to potential pulses starting on the second BC, allowing the network to read one BC before the deposit, and four on the pulse. This is found to be the best compromise between the correction for past events, the energy inference on the pulse, and short sequences meeting FPGA constraints. The sliding-window algorithm applies the network to subsequent bunch crossings allowing a prediction in real time. The final dense operation corresponds to the single neuron decoder which reads the LSTM output and calculates the energy.

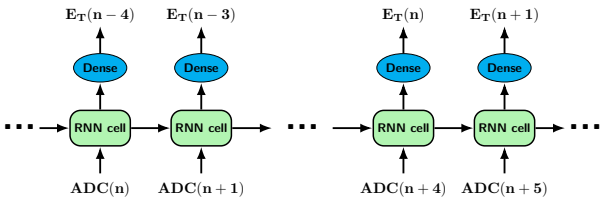

Fig. 4 Single-cell application of Long Short-Term Memory (LSTM) based recurrent networks. The LSTM cell and its dense decoder are computed at every bunch crossing (BC). They analyse the present signal amplitude and output of the past cell, accumulating long range information through a recurrent application. By design, the network predicts the deposited transverse energy with a delay of six BC.

Vanilla-RNN based algorithm: The Vanilla-RNN cell is the most compact RNN architecture. It is composed of a single internal neural network trained both to forward the relevant information in time, and to infer the energy at a given bunch-crossing. In order to fulfill constraints from the LAr calorimeter system, the size of the Vanilla-RNN internal network is reduced as much as possible. Only 8 internal dimensions are allowed. To avoid the usage of look-up-tables in the FPGA, a ReLU activation is used. As for LSTM networks, a single neuron decoder with ReLU activation function concatenates the output in a single energy measurements. In total, the network comprises only 89 parameters.

With limited internal capabilities, Vanilla-RNN networks are not capable of managing the information over long periods of time. Therefore, only a sliding window application

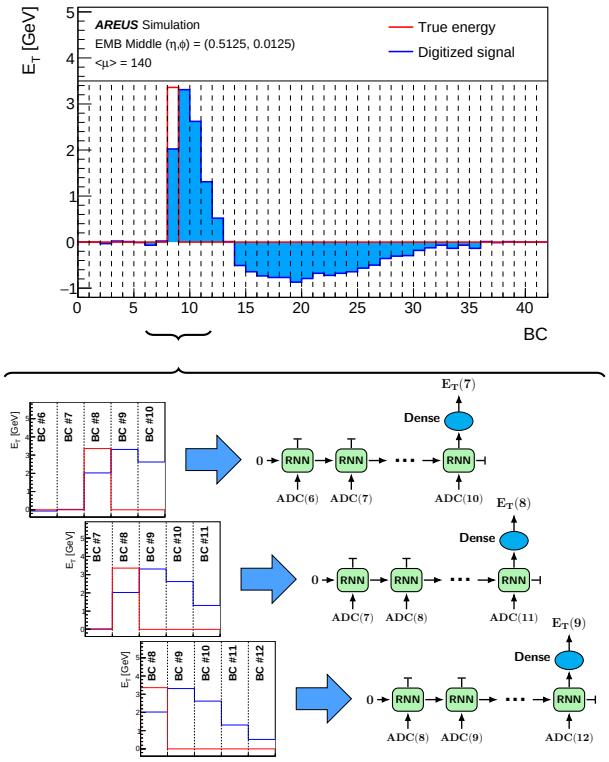

Fig. 5 Sliding window application of LSTM based recurrent networks. At each instant, the signal amplitude of the four past and present bunch crossings are input into an LSTM layer. The last cell output is concatenated with a dense operation consisting of a single neuron, and providing the transverse energy prediction.

is considered. It is defined in the same way as for LSTM networks.

Discussion: The final structure and parameter choices of the three RNN networks are shown in table 2. For the same number of parameters, the single-cell and sliding-window applications are expected to provide different insights into the features of the data. In particular, the sliding-window algorithm focuses only on a few inputs around the BC of interest: four on the pulse and one in the past. It is thus expected to be more robust when regressing the energy value of isolated data pulses. On the other hand, the single-cell design concatenates the present data with all past measurements. While this could limit the robustness of the measurement in consecutive but isolated pulses, it better alleviates remnants of past events. Out-of-time pile-up and recurrent LHC bunch patterns are typically expected to impact measurements in tens of subsequent bunch crossings. High performance in these cases requires a correction of long-lived patterns that can only be achieved with efficient management of the information through time. The single-cell design is particularly robust in situations where subsequent pulses overlap as described in section 2.4. On the other hand, the Vanilla-RNN network demonstrates performance competitive with LSTM networks. This, added to its compact design, makes the Vanilla-RNN network the most suited among the RNN based algorithms for treating individual channels of the ATLAS LAr calorimeter system.

Table 2 Configurable key parameters of the single-cell and sliding-window algorithms.

		Single-cell	Sliding-window	
		LSTM	LSTM	Vanilla-RNN
Time inference	Receptive Field	∞	5	5
	Samples after deposit	5	4	4
RNN layer	Dimension	10	10	8
	Activation	tanh	tanh	ReLU
	Recurrent Activation	sigmoid	sigmoid	N/A
Dense layer	Dimension	1	1	1
	Activation	ReLU	ReLU	ReLU
Number of Parameters		491	491	89

2.4 Results

Performance of the aforementioned ANN methods and the OF mith maximum finder are estimated in an AREUS simulation of energy deposits in one selected calorimeter cell at ($\eta = 0.5125$, $\phi = 0.0125$) in the middle layer of the barrel (labelled EMB Middle) and for long bunch crossing sequences. An average pile-up $\langle\mu\rangle = 140$ is assumed. Furthermore, only energy deposits 3σ above the noise threshold (corresponding to $E_{\mathrm{T}}^{\mathrm{true}} > 240\,\mathrm{MeV}$) are retained in what follows. Figure 6 shows a comparison of the energy resolution between the legacy OF and five ANN algorithms. The CNN and RNN networks outperform the OF both in terms of bias in the mean and of resolution. The smallest range that contains 98% of the entries is also shown to exhibit non-Gaussian behaviour present in far tails of the resolution, and particularly at low energies. The OF tends to underestimate low deposited energies while the ANNs largely recover this effect. The single-cell implementation of the LSTM network has the best performance although it has the same number of parameters as the sliding-window implementation. Even though the Vanilla-RNN has fewer parameters than the LSTM, its performance is similar in the sliding-window implementation. The CNN networks both have a comparable number of parameters. Nevertheless, the 3-Conv architecture outperforms the 4-Conv architecture. Overall, the LSTM networks reach a better performance than the CNNs

and Vanilla-RNN. However, the LSTM implementations require 5 times more parameters than the compact CNNs and the Vanilla-RNN.

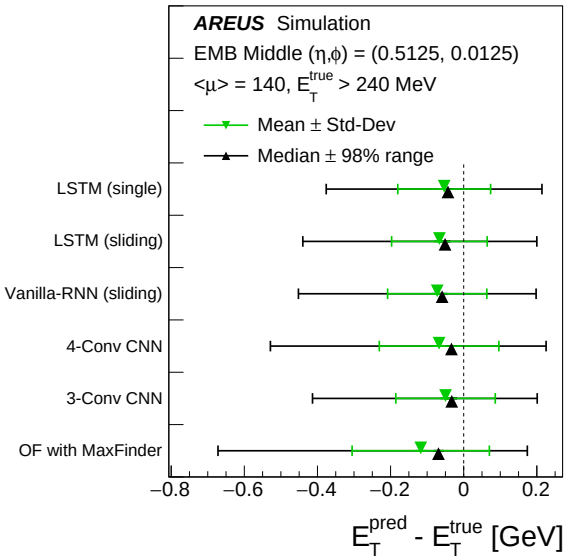

Fig. 6 Transverse energy reconstruction performance for the optimal filtering and the various ANN algorithms. The performance is assessed by comparing the true transverse energy deposited in an EMB Middle LAr cell ($\eta = 0.5125$ and $\phi = 0.0125$) to the ANN prediction after simulating the sampled pulse with AREUS assuming $\langle\mu\rangle = 140$. Only energies 3σ above the noise threshold are considered. The mean, the median, the standard deviation, and the smallest range that contains 98% of the events are shown.

One of the challenges of the energy reconstruction algorithms is the capacity to correctly predict two subsequent deposited energies with overlapping pulses. Figures 7, 8, 9 and 10 show the energy resolution as a function of the time-gap between two deposited energies. Only deposited energies above 240 MeV are considered. This ensures that the pulse amplitude is large enough to distort the pulse shape of the subsequent event. With a time-gap smaller than 20 bunch crossings the computed energy is underestimated by the OF algorithm and the resolution is significantly degraded. ANN algorithms are robust against pulse shape distortion by overlapping events and allow for an improved energy reconstruction also at small time gaps. LSTM based algorithms in the single-cell application are particularly stable along the time gap as they can access as many BC in the past as found necessary in the training phase. With 28 receptive fields for the 3-Conv, and 13 for the 4-Conv, CNN algorithms have similar performance as function of the gap. On the other hand, the sliding-window Vanilla-RNN is only using one bunch-crossing prior the deposit. Therefore, it is the least capable of correcting for overlapping pulses at short gaps.

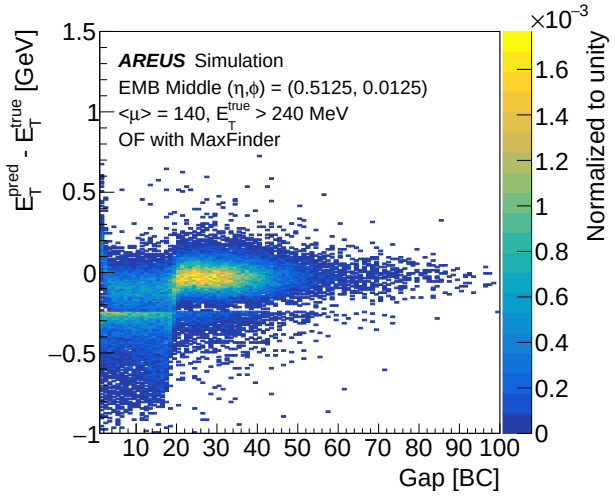

Fig. 7 Resolution of the transverse energy reconstruction as a function of the gap, i.e. the distance in units of bunch crossings (BC), between two consecutive energy deposits for the optimal filtering (OF) algorithm and a subsequent maximum finder.

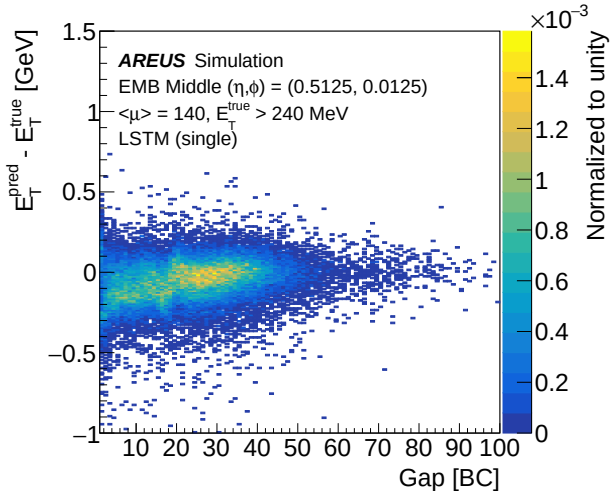

Fig. 8 Resolution of the transverse energy reconstruction as a function of the gap, i.e. the distance in units of bunch crossings (BC), between two consecutive energy deposits for the Long Short-Term Memory single-cell algorithm.

3 Network application on FPGA

3.1 Conversion of CNN to VHDL

For CNNs a direct implementation in VHDL was chosen because the network structure maps well to the multiplication-accumulation units of the DSPs available on the FPGA. A modular firmware design adapts to the specific architecture by configuration constants, which are read from a file during the synthesis or compilation stage. A Python script generates the configuration file from the Keras model files. The script also performs the transition from floating point to fixed point

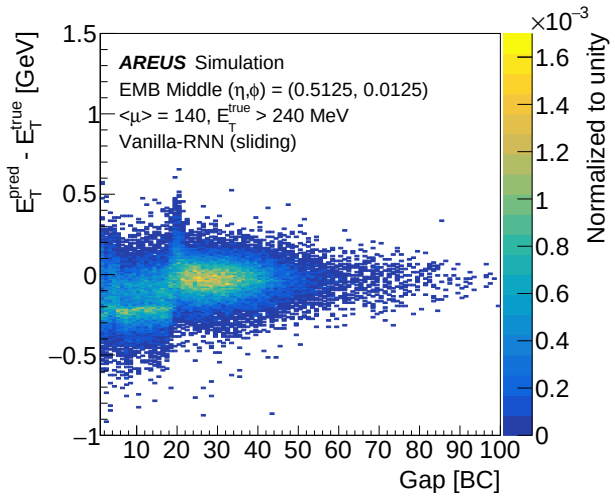

Fig. 9 Resolution of the transverse energy reconstruction as a function of the gap, i.e. the distance in units of bunch crossings (BC), between two consecutive energy deposits for the Vanilla-RNN sliding-window algorithm.

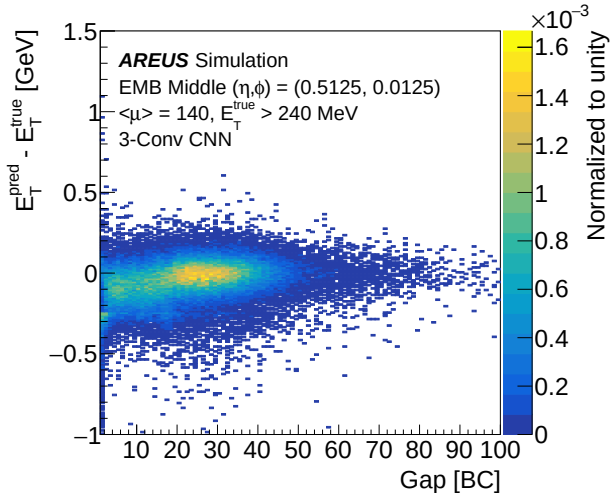

Fig. 10 Resolution of the transverse energy reconstruction as a function of the gap, i.e. the distance in units of bunch crossings (BC), between two consecutive energy deposits for the Convolutional Neural Network algorithm.

representation with a configurable total and fractional bit width. A bit width of 18 is chosen because it matches the DSP precision of the Stratix-10 FPGA. Of those 18 bits, 10 bits are used for the decimal part of the fixed-point representation. For the sigmoid activation function two implementations are available. A piece-wise linear approximation saves resources, while a look-up table (LUT) with discrete integer values allows a higher precision.

The VHDL implementation is designed in a modular way. A dedicated component realises the connections between one feature map and all feature maps of the previous layer. In this way a multi-layer CNN can be constructed, and each layer is configured independently. To make use of the high processing frequency of the FPGA, time division multiplexing is used to allow one CNN instance to process the data of multiple channels. Intermediate pipelining stages are added to the design to control the relative signal transition time between two processing steps. This, in turn, allows a high maximum frequency at which the CNN core can be executed. Moreover, processing of the input sequence is started when the first sample of each sequence arrives. This is cascaded through all layers of the network in a continuous way and minimizes the latency until the final result is available.

For all layers, the input values need to be multiplied by their respective weights. These multiplications are best performed by DSPs on the FPGA, which are dedicated for high speed arithmetic operations. In the case of the Stratix-10 FPGA, they have a special structure with two multiplication-accumulation units in one DSP. To make optimal use of the available DSPs, the serialised streams of data that are input to the FPGA, are rearranged into pairs of two in order to exploit both streams per processor. The DSPs are chained up according to the kernel size to process and accumulate the input from different time steps. The results are synchronised and summed with the first calculation path afterwards. With this approach, the DSPs can be utilized most efficiently.

Figure 11 compares the output of the VHDL implementation, simulated with Quartus 20.4 [13] and Questa Sim 10.7c [14], with the Keras CNN output. The small differences observed are on the one hand caused by discretization and the chosen bit precision, and on the other hand by the LUT-based realisation of the activation function.

3.2 Conversion of RNN to HLS

For RNN algorithms an implementation in Intel High Level Synthesis (HLS) [13] is chosen to allow additional flexibility in the design. The networks are based on two different functions, the first being the implementation of a single RNN cell, the second one handling the recursive aspect of the network architecture.

The LSTM or Vanilla-RNN cells are coded as template functions. The template is used to pass on the weights and the internal architecture of the cell. The weights and architecture parameters are automatically generated by Python scripts from the Keras model. The precision of the fixed-point value is a configurable parameter. The activation functions and the recurrent activation functions other than the ReLU are implemented as LUT. The LUTs are generated with Python scripts. A configurable parameter allows using full precision mathematical functions instead of LUTs.

Two variants of the recursive functions are implemented to support the single-cell and the sliding-window architectures. The single-cell function uses one instance of the LSTM

cell implementation and allows linking the output of this cell at a given BC to its input at the subsequent BC. A continuous output flow is achieved with data entering through recursive calls of the logic, however requiring an input frequency no larger than the cell computation time. In the sliding-window, the function invokes for each window five instances of either the LSTM cell or the Vanilla-RNN cell, one for each BC. The output of each cell serves as an input to the next. The algorithm requires one such chain of five RNN cells for each BC in order to predict the deposited energy. To be able to process data in real time without using multiple RNN chains for multiple BC, a fully pipelined design is needed. The implemented design ensures that the Initiation Interval (number of clock cycles between two inputs in HLS) is equal to one. Every loop is fully unrolled: each of the loop iterations has its own logic resources. The memory needed is implemented as registers to optimize the latency.

A comparison of the energy computation in software, as given by Keras, and in firmware simulation with Quartus 21.1 and Questa Sim 10.7c is shown in figure 11. The fixed point values are chosen to ensure a resolution of the order of 1%. For the sliding-window LSTM 18 bits are used including 13 bits for the decimal points. For the single-cell 22 bits are used including 14 bits for the decimal part. For the sliding-window Vanilla-RNN, data paths in the cell and RNN weights use different representations. Data paths use 19 bits with 16 bits for the decimal part. Weights are implemented using 16 bits out of which 13 are for the decimal part. The LUT implementation is optimized using logic to account for symmetries in the sigmoid and tanh functions. The LUT size is reduced by a factor 4 compared to the naive linear range. Their granularity is also optimized and 1024 words are found sufficient.

3.3 FPGA implementation results

In a first stage, the neural networks were implemented for a single data input channel in order to compare their basic properties. Performance results of these implementations on a Stratix-10 FPGA are shown in table 3, comparing maximum execution frequency, F_{max}, latency, and resource usage in terms of number of DSPs and Adaptive Logic Modules (ALM). The maximum achievable processing frequency for all implementations is in the range of 480 - 600 MHz. In this way up to fifteen-fold multiplexing of the input data, which is received at the LHC bunch crossing frequency of 40 MHz, is possible. In the baseline scenario imposed by the ATLAS trigger system, 150 ns can be allocated to the energy reconstruction with the optimal filtering algorithm. Only the CNN algorithms currently meet the latency constraints of the baseline scenario. However, scenarios with relaxed latency constraints are considered and could allow the usage of RNN algorithms.

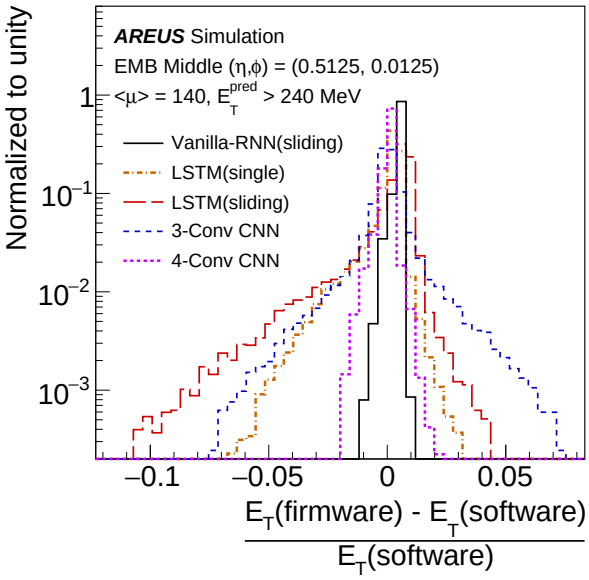

Fig. 11 Relative deviation of the firmware implementations from the software results for the different transverse energy reconstruction Artificial Neural Networks (ANN). Only bunch crossings with predictions different from zero and true transverse energies larger than 240 MeV are considered. Inputs to the ANNs are sampled pulses obtained from the simulation of an EMB Middle LAr cell ($\eta = 0.5125$ and $\phi = 0.0125$) with AREUS assuming $\langle\mu\rangle = 140$.

Table 3 Performance of the VHDL implementation of CNNs and the HLS implementation of RNNs compiled with Quartus 20.4 [13] for a Stratix-10 FPGA (reference 1SG280HU1F50E2VG) and single data input channel.

	3-Conv CNN	4-Conv CNN	Vanilla RNN (sliding)	LSTM (single)	LSTM (sliding)
Frequency F_{max} [MHz]	493	480	641	560	517
Latency $\mathrm{clk}_{\mathrm{core}}$ cycles	62	58	206	220	363
Resource Usage					
#DSPs	46 0.8%	42 0.7%	34 0.6%	176 3.1%	738 12.8%
#ALMs	5684 0.6%	5702 0.6%	13115 1.4%	18079 1.9%	69892 7.5%

The large number of readout channels to be treated by one FPGA requires time-domain multiplexing of the data processing. The CNNs and the Vanilla-RNN networks are therefore also implemented in multiplexed versions. Their performance is presented in table 4. Only the firmware designs for which the core clock frequency reaches the required value for the corresponding multiplexing factor are shown, i.e. 600 MHz for fifteen-fold multiplexing of the Vanilla-RNN and 240 MHz for six-fold multiplexing of the CNNs.

Table 4 Multiplexing performance of the VHDL implementation of CNNs and the HLS implementation of RNNs compiled with Quartus 20.4 [13] for a Stratix-10 FPGA (reference 1SG280HU1F50E2VG).

	3-Conv CNN	4-Conv CNN	Vanilla RNN
Multiplicity	6	6	15
Frequency F_{max} [MHz]	344	334	640
Latency clk_{core} cycles	81	62	120
Max. Channels	390	352	576
Resource Usage			
#DSPs	46 0.8%	42 0.7%	152 2.6%
#ALMs	14235 1.5%	15627 1.7%	5782 0.6%

The multiplexed VHDL firmware design keeps the number of DSP units at the same value as the single-channel version and requires more ALMs. On the other hand, the HLS is re-optimized for the multiplexed design, allowing notably a significant reduction of the latency. The multiplexed HLS also increases the usage of DPS units compared to its single-channel counter part, but keeps the logic resource usage at a low level.

From the estimated resource usage the maximum number of channels is calculated which can be processed by one Stratix-10 FPGA of the selected type. Assuming that 100% of the FPGA resources can be dedicated to ANN algorithms, the CNN with three convolutional layers and the Vanilla-RNN network reach a value above 384, which corresponds to the design option where data are received from three Front-End Boards of the ATLAS LAr Calorimeters. Furthermore, the Vanilla-RNN could handle the 512 channels in the scenario where data are received from four Front-End Boards.

While the priority for optimizing the ANNs shown here was to decrease the resource usage in the FPGAs, the implementations in VHDL (for CNNs) and in HLS (for RNNs) focus on different aspects. The VHDL implementation targets mainly low latency for fast execution. The HLS implementation targets high frequency to allow higher multiplexing. This is clearly seen in table 4. By further exploiting the design tools from the VHDL and HLS implementations, the CNN and RNN realisations are expected to reach even smaller resource usage, shorter latency and higher clocking frequency. The best compromise between these three parameters is yet to be reached.

4 Conclusion

Artificial Neural Networks of CNN and RNN types targeting an FPGA implementation have been successfully trained to reconstruct LAr calorimeter cell energies. The ANNs outperform the OF algorithm and still meet the tight FPGA resource constraints. Short latency and high execution frequency of the implemented networks are adapted to the requirements of the LAr real-time processing. In future, the processing cores shall be integrated into the full data processing chain within the FPGA. An optimisation of the ANNs and the development of an automated conversion to FPGA firmware using VHDL and HLS tools will be further pursued.

5 Acknowledgements

This work was in part supported by the German Federal Ministry of Education and Research within the research infrastructure project 05H19ODCA9. The project leading to this publication has received funding from Excellence Initiative of Aix-Marseille Université - A*MIDEX, a French "Investissements d'Avenir" programme, AMX-18-INT-006.

References

1. ATLAS Collaboration (2008) The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3:S08003. https://doi.org/10.1088/1748-0221/3/08/S08003
2. Evans L, Bryant Ph, eds. (2008) LHC machine. JINST 3:S08001. https://doi.org/10.1088/1748-0221/3/08/S08001
3. ATLAS Collaboration (2017) Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System, CERN-LHCC-2017-020, ATLAS-TDR-029. https://cds.cern.ch/record/2285584
4. ATLAS Collaboration (2017) Technical Design Report for the Phase-II Upgrade of the ATLAS LAr Calorimeter, CERN-LHCC-2017-018, ATLAS-TDR-027. https://cds.cern.ch/record/2285582
5. Intel Corporation (2020) Intel Stratix-10 Device Datasheet, Version 2020.12.24
6. Madysa N (2019) AREUS: A Software Framework for ATLAS Readout Electronics Upgrade Simulation, EPJ Web Conf. 214:02006. https://doi.org/10.1051/epjconf/201921402006
7. Cleland W E, Stern E G (1994) Signal processing considerations for liquid ionization calorimeters in a high rate environment, NIM A Volume 338:467-497. https://doi.org/10.1016/0168-9002(94)91332-3
8. https://keras.io/; Accessed: 2021-02-18
9. https://www.tensorflow.org/; Accessed: 2021-02-18
10. LeCun Y, et al. (1989) Backpropagation applied to handwritten zip code recognition, Neural Computation 1(4):541–551. https://doi.org/10.1162/neco.1989.1.4.541
11. Sherstinky A (2020) Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network, Physica D: Nonlinear Phenomena 404:132306. https://doi.org/10.1016/j.physd.2019.132306
12. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory Neural Computation 9:1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

13. Quartus, ModelSim and HLS tools available from https://www.intel.com; Accessed: 2021-02-18
14. Questa Sim available from https://eda.sw.siemens.com/; Accessed: 2021-06-20
15. Duarte J, et al. (2018) Fast inference of deep neural networks in FPGAs for particle physics, JINST 13:P07027. https://doi.org/10.1088/1748-0221/13/07/P07027