



BNL-222708-2022-JAAM

Waveform Processing Using Neural Network Algorithms on the Front-end Electronics

S. Miryala

To be published in "JINST"

January 2021

Instrumentation Division
Brookhaven National Laboratory

U.S. Department of Energy
USDOE Office of Science (SC), Basic Energy Sciences (BES) (SC-22)

Notice: This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or any third party's use or the results of such use of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

1 PREPARED FOR SUBMISSION TO JINST
2 22ND INTERNATIONAL WORKSHOP ON RADIATION IMAGING DETECTORS
3 JUNE 27 - JULY 1, 2021
4 ONLINE

5 **Waveform Processing Using Neural Network Algorithms** 6 **on the Front-end Electronics**

7 **S. Miryala,^{*,1} S. Mittal,* Y. Ren,* G. Carini, G. Deptuch, J. Fried, S. Yoo, S. Zohar**

8 **Equal Contribution*
9 *Brookhaven National Laboratory,*
10 *98 Rochester Street, USA*

11 *E-mail: smiryala@bnl.gov*

12 **ABSTRACT:** In a multi-channel radiation detector readout system, waveform sampling, digitization,
13 and raw data transmission to the data acquisition system constitute a conventional processing
14 chain. The deposited energy on the sensor is estimated by extracting peak amplitudes, area under
15 pulse envelopes from the raw data, and starting times of signals or time of arrivals. However, such
16 quantities can be estimated using machine learning algorithms on the front-end Application-Specific
17 Integrated Circuits (ASICs), often termed as “edge computing”. Edge computation offers enormous
18 benefits, especially when the analytical forms are not fully known or the registered waveform suffers
19 from noise and imperfections of practical implementations.

20 In this work, we aim to predict peak amplitude from a single waveform snippet whose rising
21 and falling edges containing only 3 to 4 samples. We thoroughly studied two well-accepted neural
22 network algorithms, Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN)
23 by varying their model sizes. To better fit front-end electronics, neural network model reduction
24 techniques, such as network pruning methods and variable-bit quantization approaches, were also
25 studied. By combining pruning and quantization, our best performing model has the size of 1.5KB,
26 reduced from 16.6KB of its full model counterpart. It can reach mean absolute error of 0.034
27 comparing to that of a naive baseline of 0.135. Such parameter-efficient and predictive neural
28 network models established feasibility and practicality of their deployment on front-end ASICs.

29 **KEYWORDS:** Edge Computing, Machine Learning, Front-End Electronics, Neural Networks, AI
30 ASICs, Network pruning and Quantization

¹Corresponding author.

31 **Contents**

32	1 Introduction	1
33	2 Signal Modeling and Data Preparation	2
34	3 Neural Network Models and Experiment Results	3
35	3.1 Neural Network Model Baseline Performance	4
36	3.2 Neural Network Model Pruning	4
37	3.3 Quantization-aware Training	5
38	3.4 Pruning with Quantization-aware Training	6
39	4 Conclusion	8

40 **1 Introduction**

41 Our capability to develop increasingly complex detectors capable of providing more precise infor-
42 mation of the phenomena under observation, from accelerator-based science, particle physics and
43 photon science, to large or distributed telescopes in different frequency domains, is pushing the
44 limits of the standard streaming-storing-analyzing paradigm. The search for new fundamental par-
45 ticles, for example, has driven the development of new colliders with data rates exceeding petabytes
46 per second [1]. Such computational loads burden experiments with increased data infrastructure
47 requirements. Early efforts to address this challenge have investigated porting neural networks
48 tasked with classifying jets to Field-Programmable Gate Array (FPGA) architectures [2]. This
49 work highlighted the potential for incorporating FPGA-based neural networks in the L1 trigger
50 system and evaluated the trade-offs between quantization and pruning on resource consumption
51 and performance. One promising approach for further decreasing data rates is porting the analysis
52 from FPGAs to Application-Specific Integrated Circuits (ASICs). Front-End ASICs, also called
53 ReadOut Integrated Circuits, Fig. 1, condition, digitize, and further process incoming signals before
54 streaming data to the data acquisition system. Historically, the algorithms in the ASIC’s digital
55 processor have been conventional signal processing algorithms such as impulse response filters,
56 peak detectors, and digitizers. Recent advances in machine learning have demonstrated that neural
57 networks can be trained to work as best estimators, i.e. to regress ground truth parameters from data
58 generated with *ab-initio* detector system simulations, Fig. 1. This allows for estimating particle’s de-
59 posited energy without calibration and computationally expensive global fit methods. Incorporating
60 this computational approach in an ASIC would reduce data throughput by having a more intelligent
61 interface with, for example the L1 trigger system. In light of this promising potential, efforts to
62 realize ASIC-based neural networks have recently begun attracting significant interest. In [3], the
63 authors propose a data compression algorithm using deep neural networks for the Phase-II upgrades
64 of the CMS Collaboration’s HGCALE (High-Granularity CALorimeter) experiment. In that study,

65 a Convolutional Neural Network (CNN) -based autoencoder was realized using automatic layout
 66 design (i.e, place and route) tools with more than half a million digital logic gates dissipating few
 67 100's of mW of power and consuming significant real-estate. Area and power metrics are critical
 68 design parameters, which define the boundaries for the design and usable implementation of ASICs.

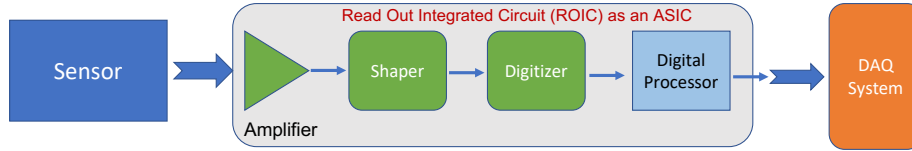


Figure 1: Introducing a neural processor for real-time data processing on the readout integrated circuit.

69 2 Signal Modeling and Data Preparation

70 Ground truth training data, of which the generation process is shown in Fig. 2, was prepared as
 71 sampled waveform snippets resulting from simulating the readout analog circuitry and modelling
 72 the sensor as a 50 μm thick low-gain avalanche diode sensor. Charge pulses are simulated using a
 73 model based on straggling functions [4] for energy loss of minimum ionizing particles traversing
 74 a silicon sensor and generation of charge due to drift and multiplication. The charge pulses are
 75 represented as 25-point piece-wise linear functions and converted into the s-domain. The sensor
 76 currents and distribution of charge pulses are shown in Fig. 2 a, b. The analog processing chain,
 77 described as a CR-RC3 filter, is characterized by the peaking time of the impulse response chosen
 78 to be approximately two times longer than the duration of the charge pulses and is shown in Fig. 2
 79 c. The time domain response of the analog chain stimulated by the charge signals is calculated as
 80 an inverse Laplace transform of the product of s-domain representation of the sensor charge pulses
 81 and the transfer function of the filter. Examples of these operations are shown in Fig. 2 d. The time
 82 domain response of the analog chain are recorded as waveform snippets allowing some padding
 83 before each pulse. Then, noise is added in time domain, by generation noise sequences, chopping
 84 their length to match the lengths of the waveform snippets and adding both as shown in Fig. 2 e.
 85 Noise is generating as bandwidth limited time sequences obtained by a superposition of individual
 86 impulse responses to a sequence of delta pulses of randomized polarity and amplitude. A typical
 87 noise power frequency spectrum, for which the dominating noise source is the first stage of the
 88 processing chain is modeled. The variance of the noise time sequence is calculated and the noise
 89 process is scaled to result in the planned signal-to-noise ratio, for example, 30 or 15. A total of
 90 about 10k waveform snippets, such as the one shown in Fig. 2 f, with noise have been generated.
 91 The original values of charge magnitude and time of arrival have been written together with each
 92 waveform snippet to allow testing of the investigated processing methods. The subsampling, used
 93 in the later experiments, is depicted Fig. 2 f by red circles.

94 The original simulation waveform timing window of 8ns is sampled at the rate of $\sim 3\text{ps}$,
 95 containing 3401 samples. To reflect waveform digitization in real applications, all waveforms are
 96 downsampled by about 190-fold, reducing its dimension (1D) from 3401 to 18 i.e sampling rate
 97 of $\sim 437\text{ps}$, resulting in 0 to 1 sample on the rising transition and 1 to 2 samples on the falling
 98 transition of a waveform. The subsampling is a challenge as the maximum value in a downsampled

99 waveform likely deviates much further from its ground truth amplitude. If we naively treat the
 100 maximum values in downsampled waveforms as our predictions, the average absolute error is
 101 0.135. Therefore, our models aim to outperform this naive method. We randomly separate the
 102 waveforms into train-validation and test subsets. The test dataset has never been exposed to the
 103 model training or validation to reflect the model performance on the test dataset on unseen data.

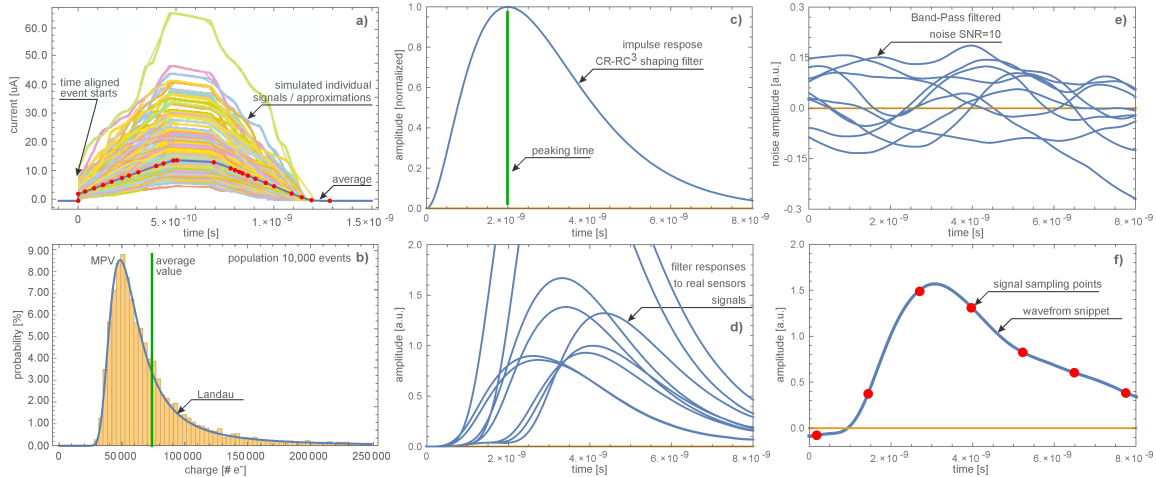


Figure 2: Generation of sampled waveform snippets, a) sensor currents with average current waveform, b) distribution of signal charges, c) impulse response of electronic analog filter, d) convolutions of sensor currents with filter impulse responses, e) electronic noise waveforms, f) waveform snippet with sampling

104 3 Neural Network Models and Experiment Results

105 Neural-network-based machine learning approaches have demonstrated tremendous success in complex
 106 recognition tasks that previously were prohibitive. Here, we consider two well established
 107 network architectures: MLP and CNN [5]. MLP consists of a sequence of fully connected (fc)
 108 layers with non-linear activation functions. A fc layer is a linear model that takes an input vector,
 109 multiplies it by a weight matrix, and adds a bias vector (Fig. 3). Despite their apparent simplicity,
 110 such layers, when stacked upon each other, possess universal approximation capabilities [6] and are
 111 competitive with state of the art convolutional and attention based models [7]. A CNN consists of a
 112 sequence of convolutional layers followed by non-linear activations. A convolutional layer (conv)
 113 consists of multiple small matrices, called “kernels” (colored in orange in Fig. 3). Each kernel scans
 114 through the input signal by a fixed stride and computes the inner product with the aligned portion
 115 of the input.

116 Computationally expensive operations such as exponentiation and division have been not used
 117 by restricting activation functions to ReLU and avoiding batch normalizations to meet power, area,
 118 performance and latency constraints. Recent studies have shown that a norm-free neural network
 119 can perform well [8]. To promote low latency, the investigated networks were of not more than 6
 120 layers. The resulting network architectures are suitable for deployment on most non-GPU computing
 121 hardware, such as FPGAs, intelligence processing units (IPUs), microcontrollers, or in ASICs.

122 Section 3.1 details the performance of various MLP and CNN configurations as the baseline.
 123 Section 3.2 shows the experiments with different pruning methods on these networks. Section 3.3
 124 applies variable-bit quantization-aware training to reduce model sizes, and Section 3.4 details how
 125 combining pruning with quantization further reduces model sizes.

126 3.1 Neural Network Model Baseline Performance

127 Neural architectures and suitable hyperparameters are explored and determined by experimenting
 128 with various network configurations. The layer widths and kernel sizes in consideration are tabulated
 129 in Table 1. To establish a fair comparison between MLPs and CNNs, models of different sizes are
 130 denoted as ‘‘Tiny’’ (T), ‘‘Small’’ (S), ‘‘Medium’’ (M), and ‘‘Large’’ (L) for the number of parameters
 131 in the proximities of 400, 1300, 4000 and 9700, respectively. The number of layers of both MLP
 132 and CNN are fixed at five: one input layer, three hidden layers, and one output layer. Using the
 133 tiny MLP as an example, the input layer has a dimension of $\text{input-dim} \times 8$, the three hidden layers
 134 have 8×8 , and the output layer has 8×1 , where the input-dim is 18. Each number in an MLP
 135 configuration indicates the width of a fully connected layer. For the CNN configurations, the first
 136 three numbers indicate the number of output channels in convolutional layers with the last number
 137 noting the width of the fully connected output layer.

Table 1: Model configurations of multi-layer perceptron (MLP) and convolutional neural network (CNN) with the corresponding number of parameters.

(a) Model Configurations of MLP			(b) Model Configurations of CNN		
MLP	Config.	# Param.	CNN	Config.	# Param.
Tiny (T)	8-8-8-8	377	Tiny (T)	2-2-2-16	453
Small (S)	16-16-16-16	1137	Small (S)	3-3-3-32	1289
Medium (M)	32-32-32-32	3809	Medium (M)	5-5-5-64	4149
Large (L)	52-52-52-52	9309	Large (L)	6-6-6-128	9725

138 Models are trained to regress the waveform ground truth amplitude using the MSE as the target
 139 loss function and the AdamW optimizer. The optimal number of epochs required for training is
 140 determined by using the 90/10 train-valid split and finding the minimum validation error epoch.
 141 The model then is retrained with the entire training data for the optimal number of epochs.

142 All models were evaluated on a held-out data set using the Mean-Absolute-Error, $\text{MAE} =$
 143 $1/N \sum_{i=1}^N |y_i - \hat{y}_i|$. Lower MAEs between predicted signal amplitudes and ground truth values are
 144 better. Violin graphs shown in Fig. 3 capture the highest (violin bottom), lowest (violin top) and
 145 median performing models sampled from 10 runs for the specific model configuration.

146 Model sizes also play an important role. In general, the tiny models do not perform as well as
 147 other models. The large models have similar performance as the medium models but are prone to
 148 over-fitting, resulting in a greater variance in model performance. Therefore, the remaining sections
 149 about network pruning and quantization primarily focus on the medium and large models.

150 3.2 Neural Network Model Pruning

151 Network pruning reduces a neural network’s size. Through pruning, model weights with the lowest
 152 magnitudes are removed, as smaller values in a weight matrix will have lesser impact on the output.

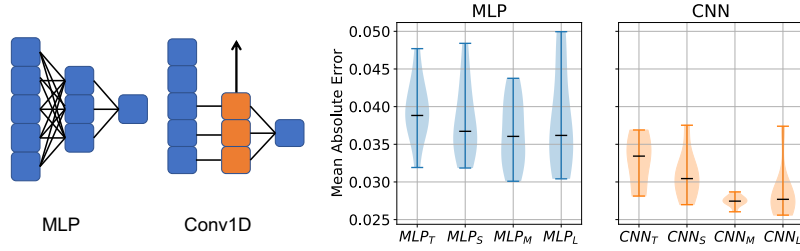


Figure 3: Schematics of MLP and Conv1D. Performances of MLP and CNN in various model sizes.

153 Iterative pruning combines the pruning and training steps. This allows the weights of the remaining
 154 network to adapt to the network architecture change, resulting in a better model. Interestingly,
 155 Frankle and Carbin [9] have proposed that after each pruning step, the remaining weights should be
 156 reset to the model initial values. By doing so, they found the pruned model can even outperform the
 157 original. This suggests that at least one subnetwork exists with the particular initial values and can be
 158 trained to perform very well. This is the so-called, “Lottery Ticket Hypothesis (LTH)”. We consider
 159 additional two parameter resetting strategies – continuous pruning (CP) not changing the values of
 160 remaining weights and random reinitialization (RR) resampling the remaining weights randomly.
 161 For parameter ranking, global unstructured L1 pruning was used for its superior performance [10].

162 Pruning was scheduled, Fig. 4c, to start after the 100th epoch and repeat every 20 epochs until
 163 training completion. Each pruning cycle removes 10% of the remaining weights and leaving 10 %
 164 of the original weights in the end ($0.9^{21} \approx 10\%$). The total number of epochs was 520. As only
 165 weights were pruned and not biases, CNN and MLP result in the different sparsity values.

166 Ten runs for each combination of four different network models and three pruning methods
 167 were performed. Then, we smooth each run using a uniform kernel length of 5 averaged over 10
 168 runs. Figures 4a, b, d, and e show the pruning results of different methods on four respective
 169 models: MLP_S , CNN_S , MLP_M , and CNN_M . The color bands indicate two standard deviations.
 170 Surprisingly, all pruning methods perform similarly on all network models in the experiments. All
 171 models can reach the same or similar MAE of their non-pruned full model counterparts. This
 172 suggests that our proposed pruning technique is a viable approach to reduce the model parameters
 173 under hardware resource constraints or to decrease hardware processing cycles [11, 12].

174 3.3 Quantization-aware Training

175 Model quantization aims to reduce model sizes by replacing 32-bit floating-point parameters with
 176 smaller 8 or 16 bit fixed-point representations. Quantization-aware training (QAT) simulates noises
 177 induced by low-precision representations during network training, which improves the quantized
 178 model performance comparing the post-training quantizations [13]. We furthered the QAT by
 179 considering variable bit QAT that different layers have different precision bits using QKeras.

180 The input and output are fixed to 16-bit for not losing information at beginning and not inducing
 181 error by low precision for regression. To reduce the search space, four hidden layers are grouped into
 182 “first two” and “last two”. Each layer group has three choices (4-bit, 6-bit or 8-bit) for fixed-point
 183 representations, resulting in $3^2 = 9$ total bit-precision configurations for both MLP and CNN.

184 Ten runs were performed for each bit-precision configuration on each MLP and CNN models,
 185 Fig. 5. Quantized models are ordered by their model sizes after quantization. As the number of

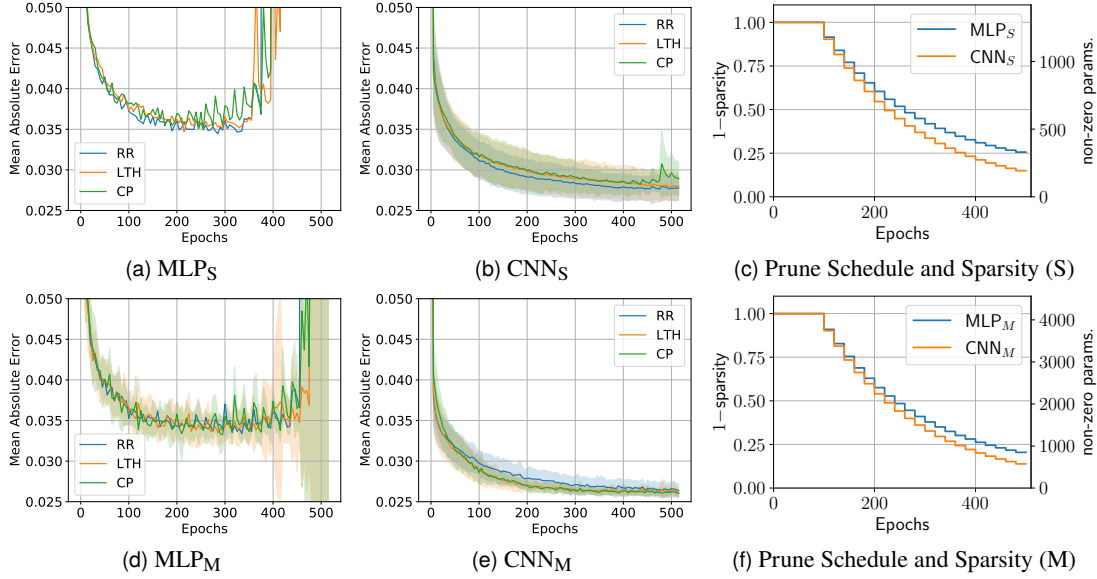


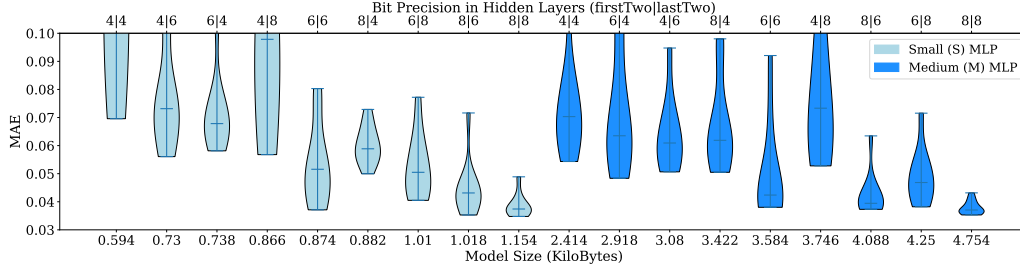
Figure 4: Performance of the RR, LTH and CP pruning methods on S and M - sized MLP and CNN models.

186 parameters of CNN concentrate more in the last hidden layer of fully-connected, same bit-precision
 187 configuration of MLP and CNN may end up in different ranks. Results of QAT in the MLP model,
 188 Fig. 5a, show that when the precision of all four hidden layers is 8 bits, the MLP_M (4.75KB) can
 189 reach the test MAE of 0.0373 (averaged over 10 runs), whereas MLP_S (1.15KB) with all hidden
 190 layer of 8 bits of 0.0375, which is close to that of MLP_M . When the precision of the first two hidden
 191 layers is less than the precision of the last two hidden layers, a loss of information is evident as the
 192 evaluation MAE in such cases is relatively higher than others. In terms of the CNN models, Fig. 5b,
 193 when the precision of hidden layers is homogeneous 8-bits, the MAE is lowest, around 0.036. Even
 194 though CNN_S (0.75KB) has only two-thirds of the parameters comparing to its MLP_S (1.154KB),
 195 it has better performance. Therefore, CNN models deliver better performance at lower precision
 196 using QAT with a small memory footprint.

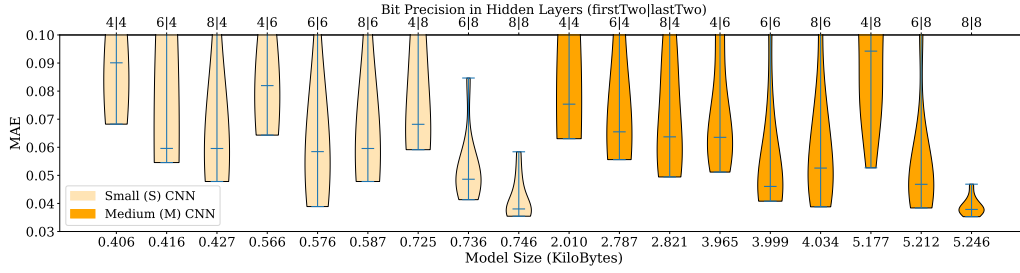
197 3.4 Pruning with Quantization-aware Training

198 Here, we explore the effects of applying both pruning and quantization during training for our
 199 task. In another study, the combined approach can reduce the model size significantly without
 200 sacrificing accuracy [14]. In our implementation, we use TensorFlow’s Model Optimization Toolkit
 201 to implement pruning with QAT (PQAT).

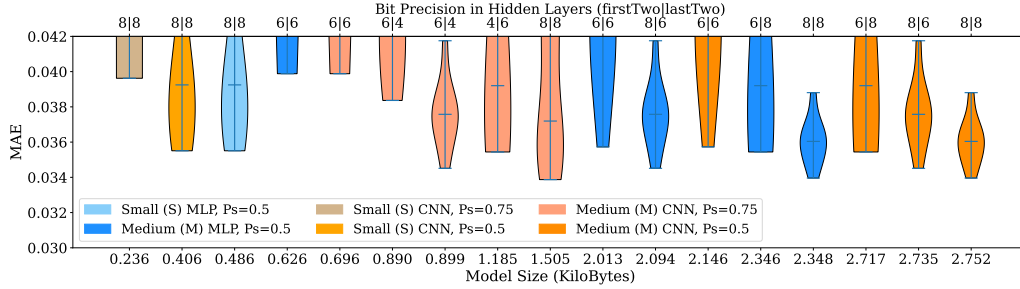
202 The choices of pruning the sparsity (P_s) targets, 50% and 75%, were guided by the results in
 203 Sec. 3.2. Fig. 5c shows the evaluation results of applying PQAT on the small- and medium-sized
 204 MLP and CNN models for a given pruning sparsity. Configurations with MAEs greater than 0.04
 205 are filtered out. Overall, the results agree with our intuition: larger models can be quantized and
 206 pruned further. When MLP_S is trained with 75% sparsity, its MAE is even larger because the model
 207 size becomes too small. However, in MLP_M for 50% P_s and with homogeneous 8-bit precision in
 208 hidden layers, the performance is as good as MLP_M in QAT (shown in Fig. 5a). With PQAT, MLP_M
 209 model size is reduced from 4.75 KB to 2.35 KB, and it delivers equally good performance (MLP_M



(a) Evaluation of QAT in MLP Model



(b) Evaluation of QAT in CNN Model



(c) Evaluation of PQAT in MLP and CNN (Ps = Pruning Sparsity).

Figure 5: Variable Bit Quantization-aware Training (QAT) and Pruning with QAT in CNN and MLP.

210 in QAT). In CNN, using PQAT with Ps 50% and 75% results in many bit-precision configurations
 211 that have low MAEs (less than 0.04). For instance, CNN_S 8|8 (homogeneous 8-bit precision in
 212 hidden layers) with Ps 75% results in the smallest model size of only 0.236 KB. In some trials (out
 213 of 10 total), it has an MAE under 0.04 as shown in Fig. 5c. This still performs as good as its larger
 214 counterpart CNN_S (0.746 KB) in QAT experiments (Fig. 5b). Other bit-precision configurations
 215 in CNN_M have performed even better. CNN_M 6|4 with Ps 75% (0.899 KB) results in a good MAE
 216 of 0.0372, which has significantly improved from its larger counterpart (2.786 KB) in QAT that
 217 results in MAE of 0.0665 without pruning. In addition, the best performing CNN_M 8|8 in QAT has
 218 a size of 5.246 KB, and its size is reduced to almost by half in PQAT with a Ps 50% to 2.75 KB
 219 while delivering a similar performance. Using PQAT in CNN and MLP shows that CNN delivers
 220 better performance even when the pruning sparsity is 75%, retaining only 25% of model parameters
 221 in memory. Models trained using PQAT have a lower memory footprint and MAE compared to
 222 models with separate QAT and pruning stages.

223 4 Conclusion

224 We have demonstrated that pruned and quantized MLPs and CNNs can recover peak values from
225 downsampled waveforms generated by interactions of particles in sensor. In the absence of pruning,
226 MLPs and CNNs outperform the simple model MAE of 0.135 by over 4x and achieve MAE
227 evaluation metrics of 0.03 and 0.026, respectively. Pruning and quantization reduce the model size
228 even by 90%, modestly increasing the evaluation error from 0.026 to 0.034. These findings show
229 that ASIC-based neural networks are promising candidates for real-time edge-computation. Future
230 work will continue the co-design approach by designing an ASIC which will implement such neural
231 network models and automate architecture search and model reduction to better fit the hardware.

232 References

- 233 [1] by MéliSSa Gaillard and S. Pandolfi, *CERN Data Centre passes the 200-petabyte milestone*, .
- 234 [2] A. Heintz, V. Razavimaleki, J. Duarte, G. DeZoort, I. Ojalvo, S. Thais et al., *Accelerated charged*
235 *particle tracking with graph neural networks on fpgas*, 2020.
- 236 [3] G. D. Guglielmo, F. Fahim, C. Herwig, M. B. Valentin, J. Duarte, C. Gingu et al., *A reconfigurable*
237 *neural network asic for detector front-end data compression at the hl-lhc*, *IEEE Transactions on*
238 *Nuclear Science* **68** (2021) 2179.
- 239 [4] F. Cenna, N. Cartiglia, M. Friedl, B. Kolbinger, H. F. W. Sadrozinski, A. Seiden et al., *Weightfield2:*
240 *A fast simulator for silicon and diamond solid state detector*, .
- 241 [5] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*. MIT Press.
- 242 [6] K. Hornik, M. Stinchcombe and H. White, *Multilayer feedforward networks are universal*
243 *approximators*, *Neural Networks* **2** (1989) 359.
- 244 [7] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner et al., *MLP-mixer: An*
245 *all-MLP architecture for vision*, 2105.01601.
- 246 [8] A. Brock, S. De, S. L. Smith and K. Simonyan, *High-performance large-scale image recognition*
247 *without normalization*, 2102.06171.
- 248 [9] J. Frankle and M. Carbin, *The lottery ticket hypothesis: Finding sparse, trainable neural networks*,
249 <https://openreview.net/forum?id=rJl-b3RcF7>.
- 250 [10] D. Blalock, J. J. G. Ortiz, J. Frankle and J. Gutttag, *What is the state of neural network pruning?*,
251 2003.03033.
- 252 [11] M. Shimoda, Y. Sada and H. Nakahara, *FPGA-based inter-layer pipelined accelerators for filter-wise*
253 *weight-balanced sparse fully convolutional networks with overlapped tiling*, .
- 254 [12] M. Sun, P. Zhao, M. Gungor, M. Pedram, M. Leeser and X. Lin, *3d CNN acceleration on FPGA using*
255 *hardware-aware pruning*, in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6,
256 DOI.
- 257 [13] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard et al., *Quantization and training of neural*
258 *networks for efficient integer-arithmetic-only inference*, 1712.05877.
- 259 [14] B. Hawks, J. Duarte, N. J. Fraser, A. Pappalardo, N. Tran and Y. Umuroglu, *Ps and qs:*
260 *Quantization-aware pruning for efficient low latency neural network inference*, *Frontiers in Artificial*
261 *Intelligence* **4** (2021) .