TECHNISCHE UNIVERSITÄT DRESDEN

Bitwise Optimization of Artificial Neural Networks for the Energy Reconstruction of ATLAS Liquid Argon Calorimeter Signals

Master-Arbeit
zur Erlangung des Hochschulgrades
Master of Science
im Master-Studiengang Physik

vorgelegt von

Alexander Lettau
geboren am 20.07.1997 in Frankenberg

Institut für Kern- und Teilchenphysik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
2023

Eingereicht am 20. November 2023

1. Gutachter: Prof. Dr. Arno Straessner
2. Gutachter: Dr. Frank Siegert

Summary

Abstract
English:
The Large Hadron Collider will be upgraded to become the High-Luminosity LHC. This upgrade will need improvements for the readout systems of the ATLAS detector to deal with the increased amount of data. The energy reconstruction of the ATLAS Liquid Argon Calorimeter, in particular, will need improvements because the currently used Optimal Filter has been shown to drastically lose performance in a high pile-up scenario, as expected after the upgrade. Artificial neural networks are an alternative because they can do non-linear operations. These networks are supposed to be installed on FPGA near ATLAS. This implementation will take relatively much computational resources compared to the Optimal Filter. Quantization is a technique to reduce the bit widths of neural networks. Reducing the bits used for FPGA implementation decreases resource consumption and simplifies the implementation. For this reason were convolutional neural networks limited to 100 parameters trained quantization aware to low bit widths in this work. With the saved resources, larger networks were optimized, and the increased performance was evaluated.

Abstract
Deutsch:
Der Large Hadron Collider wird momentan zum High-Luminosity LHC ausgebaut. Dadurch wird es zu stark erhöhten Datenflüssen in der elektronischen Auslese kommen. Um dies zu kompensieren, müssen mehrere Systeme des ATLAS-Detektors verbessert werden. Insbesondere die Energierekonstruktion des Flüssig-Argon-Kalorimeters ist davon betroffen, weil gezeigt wurde, dass momentan genutzte Optimalfilter durch erhöhtes pile-up an Leistung verlieren werden.
Künstliche neuronale Netze sind eine Alternative zu diesem, da sie nichtlineare Funktionen darstellen können. Für die praktische Anwendung am ATLAS Detektor müssen diese Netze auf FPGA implementiert werden. Neuronale Netze verbrauchen auf FPGA relativ viele Rechenressourcen im Vergleich zum Optimalfilter. Quantisierung der neuronalen Netze ist eine Möglichkeit die Bitbreite in den Netzen zu reduzieren, was den Verbrauch an Ressourcen verringert. In dieser Arbeit wurden faltende neuronale Netzwerke, die auf 100 Parameter limitiert waren, auf niedrige Bitbreiten quantisiert, um die FPGA-Implementierung zu vereinfachen und Rechenressourcen zu sparen. Der reduzierte Verbrauch wurde genutzt, um größere neuronale Netze zu optimieren und die gewonnene Leistung wurde untersucht.

Contents

Acronyms

ADC	Analog-to-Digital-Converter
ALM	Adaptive Logic Module
AREUS	ATLAS Readout Electronics Upgrade Simulation
ASIC	Application-Specific-Integrated Circuit
BC	Bunch Crossing
BCE	Binary Cross Entropy
Cern	Conseil européen pour la Recherche nucléaire
CMS	Compact Muon Solenoid
CNN	Convolutional Neural Network
DSP	Digital Signal Processor
EMB	Electromagnetic Barrel
EMEC	Electromagnetic End-Caps
FCAL	Forward Calorimeter
FPGA	Field Programmable Gate Array
HEC	Hadronic End-Caps
HE-LHC	High Energy Large Hadron Collider
HL-LHC	High Luminosity Large Hadron Collider
LAr	Liquid Argon Calorimeter
LHC	Large Hadron Collider
LHC-b	Large Hadron Collider beauty
MAE	Mean Absolute Error

OF	Optimal Filter
QReLU	Quantized Rectified Linear Unit
ReLU	Rectified Linear Unit
SM	Standard Model of particle physics
TCNN	Tagging Convolutional Neural Network

1 Introduction

1.1 Motivation

Modern physics knows four fundamental interactions on which all known physical processes are based. These interactions are gravity, electromagnetism, the weak and the strong interaction. All interactions except gravity can be described theoretically with the Standard Model of Particle Physics (SM) [1]. Testing and falsifying the Standard Model requires particle accelerators, with access to high energies, to find particles or interactions predicted by the theory. The biggest accelerator currently is the Large Hadron Collider (LHC) at CERN. It accelerates particles and collides them, with a center of mass energy of up to 13.6 TeV. At this energy, observing and studying rare physical processes from the early universe is possible. One highlight of the LHC was the discovery of the Higgs boson, the last unobserved particle predicted by the Standard Model [2].

However, it is known that the Standard Model is not complete. The last interaction, gravity, is described in general relativity, which is incompatible with the Standard Model. Because of this, there must be a more general theory that combines both models. Also, observations from astrophysics predict the existence of dark matter and dark energy, which cannot be described with Standard Model physics at the moment. Recent measurements of the anomalous magnetic dipole moment of muons also show deviations between predictions of the standard model and experimental data [3]. Other open questions are, for example, the neutrino masses, motivated by neutrino oscillations, or the observed dominance of matter over antimatter in the universe. Answering these questions makes it necessary to extend the Standard Model.

Most standard model extensions predict the existence of new particles to explain these observations. Searching for these particles with particle colliders enables falsifying the models. Until now, the LHC has found no new fundamental particle after the Higgs boson. This, in turn, allows the rejection of Standard Model extensions that predict particles in the range of the LHC.

Many remaining extensions and new alternatives predict particles that are very hard to detect. To discover them regardless, further improvements on the performance of the LHC are needed [4].

1.2 The Standard Model of Particle Physics

The SM describes particles and the interactions between them. It was developed as a theory in the 1970s based on the three known interactions of electromagnetism, the weak and the strong interaction. Gravity is not included because it cannot be fully described with known quantum physics.

The SM knows two types of particles: bosons with an integer spin and fermions with a half-integer spin. Interactions in it are based on the symmetry groups $SU(3)_C \times SU(2)_L \times U(1)_Y$. Every symmetry group corresponds to one conserved quantity, the so-called charge. These charges are the color charge C for the strong interaction, the hypercharge Y for electromagnetic interactions, and the weak isospin T_3 for weak interactions. Assuming invariance under local gauge transformation requires the introduction of additional gauge fields whose excitations can be described as bosonic particles. These act as carriers of forces. They are the photon for electromagnetic interactions, gluons for strong interactions, and the W and Z bosons for weak interactions. Particles with a charge can then interact with each other by exchanging these gauge bosons. For example, two quarks can interact via strong interaction because they have color charge by exchanging a gluon. All known existing particles are thereby predicted by the SM. The particles of the SM can be seen in Fig. 1.1.

During its creation, some particles predicted by the SM were not observed yet (most famously the Higgs boson, but also the Tau neutrino or Top quark). With the discovery of the Higgs boson in 2012, all particles of the SM have been observed. Now, particle physics is trying to find new physics beyond the SM.

Extensions of the SM usually predict new particles. For example, a natural way to quantize gravity would be to predict a new gauge boson, the graviton. Explaining dark matter with new particles is possible by assuming the existence of a weakly interacting particle with a mass in the range of around 100 GeV (called WIMP) [6]. Many extensions of the SM predict such particles.

Direct observation of new particles to test and falsify those extensions requires particle accelerators.

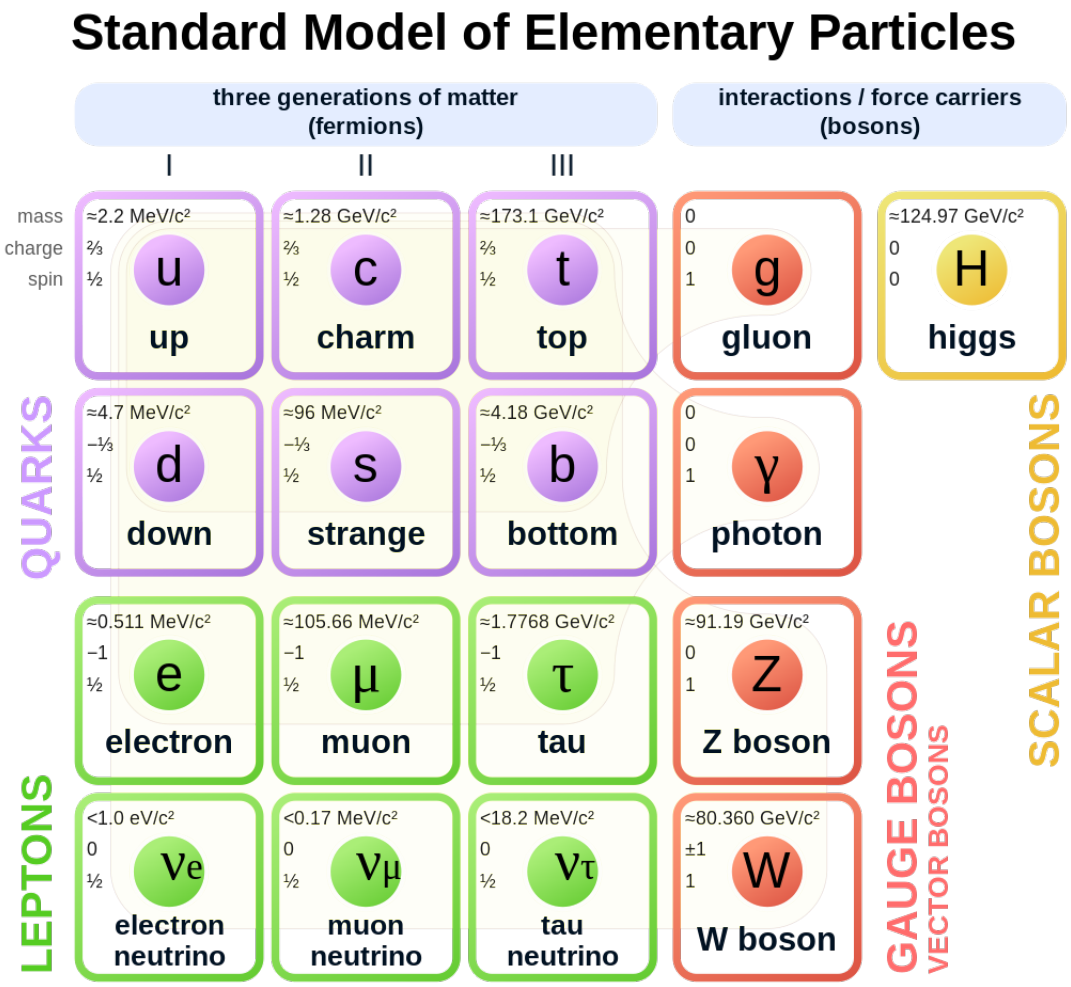

Figure 1.1: The particles of the standard model. On the left, the fermions are shown. These are made up of quarks, which are strongly interacting, and of leptons. Leptons are again made up of electrically charged leptons and neutrinos. Both types of fermions appear in three generations of particles. Bosons can be seen on the right. They are made up of vector bosons, which are the carriers of forces, and scalar bosons, of which only the Higgs boson is predicted [5].

1.3 The Large Hardron Collider

With a circumference of 26.7 km, the LHC at CERN is currently the largest particle accelerator in the world [7]. In it, two opposing proton beams are accelerated via electric and magnetic fields to energies up to 6.8 TeV and brought into collision. During a collision of two protons, this energy results in an invariant mass of 13.6 TeV, which leads to the creation of heavy, short-lived particles and enables the study of them by observing their decay. The LHC can also accelerate beams of heavy nuclei, mostly lead, to energies of up to 5.36 TeV. This reduces the accessible mass but enables observing processes that need very high densities, primarily to observe a quark-gluon plasma.

Inside the beams, the protons are packed into packages with a Gaussian distribution called

bunches. Every bunch includes around 10^{11} protons after its injection into the LHC. The standard deviation of the bunches is in the order of cm in longitudinal direction and μm in radial direction. Collisions between these beams happen with a frequency of once every 25 ns. This time between two collisions defines a bunch crossing (BC), which is a commonly used unit of time at the LHC. At every collision, around 20 proton-proton collisions are expected.
Two necessary units for particle accelerators are the luminosity and the cross section. The cross section (σ) of a process is defined as:

$$\sigma = \frac{\overline{N}}{\phi} \tag{1.1}$$

With the mean number of particle interactions ($\overline{N}$) and the particle fluency (ϕ). The fluency is defined as:

$$\phi = \frac{\Delta N}{\Delta A} \tag{1.2}$$

With the number of particles N and the area of the target A. A cross section can be interpreted as a unit for the probability of a specific process that can be measured.
Knowing the cross section, the luminosity can be defined as:

$$L = \frac{1}{\sigma}\frac{dN}{dt} \tag{1.3}$$

Again, with the number of particles N and the time t. Integrating the luminosity over time gives the integrated luminosity:

$$L_{\text{int}} = \int L dt \tag{1.4}$$

The integrated luminosity is proportional to the number of collisions at an accelerator. It is, therefore, specific to every accelerator and serves as a measurement for the gathered data during operation.
The cross section can also be calculated theoretically via Fermi's golden rule:

$$\sigma = \frac{2\pi}{\hbar}\frac{1}{L}|M_{fi}|^2\rho \tag{1.5}$$

With the probability amplitudes M_{fi} and the phase space factor (ρ) whose values depend on the model that is assumed. Measuring cross sections and comparing them with the theoretical value of a model is a way to falsify the model.
At the end of Run 2, the integrated luminosity of the LHC reached 190fb^{-1}. For run 3, this is expected to go up to 390fb^{-1}, which gives more data for analysis. These data are created and gathered at the LHC interaction points.

1.4 The ATLAS Detector

The LHC has four main interaction points in which the beams are brought into collision. All of these have a detector that is designed around them. These are ATLAS [8], CMS [9], ALICE [10] and LHCb [11]. This thesis focuses on the ATLAS detector. A representation of ATLAS is shown in Fig. 1.2.

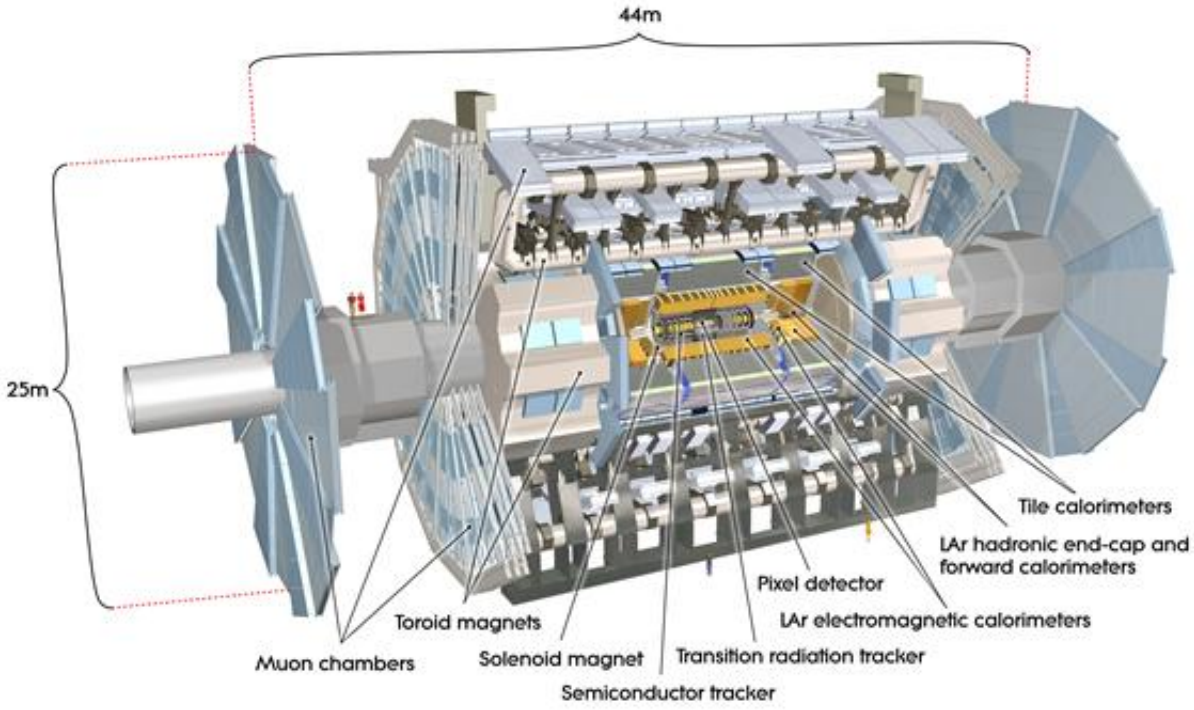

Figure 1.2: Representation of the ATLAS detector and its systems [12].

The innermost part of ATLAS is used for tracking. This means that the trajectory of charged particles moving through it is reconstructed [13]. A pixel detector, a semiconductor tracker, and a transition radiation tracker do measurements in this part. Since the entire tracking part is built in a magnetic field that is created through solenoid electromagnets, electrically charged particles will move in a curved trajectory through it. The curvature of the trajectory is proportional to the charge and momentum of the particles. Measuring the curvature makes it possible to get information on the momentum and possible identity of particles.
After the tracking are the ATLAS calorimeters. These measure the energy of particles traveling through them. At ATLAS, there are two calorimeter systems:

- First, the Liquid Argon Calorimeters (LAr). These are mainly for measurements of electrons and photons [14].
- Second, the Tile Calorimeters (TileCal). These measure the energy of mainly strong interacting particles (e.g., protons, neutrons, and long-lived baryons or mesons) [15].

After the calorimeters, most particles have lost all their energy and are therefore stopped and measured in the detector. Exceptions are particles that cannot interact via electromagnetic or strong interaction (like neutrinos and possible unknown particles). These particles cannot be directly measured in ATLAS. Another exception are muons, which only deposit small amounts of energy because they do not interact via strong interaction and have a relatively high mass. For measurements of muons, the last part of ATLAS is the muon spectrometer, which does measurements on the muons inside a magnetic field [16].
This layered detector set-up enables the identification of particles by observing their visibility in the different detector parts, see Fig. 1.3. For example, electrons will move through the tracker in a curved trajectory, which creates a signal in the tracker and lose all energy in the LAr. Neutrons, on the other hand, as neutral particles, will move through the tracker without being influenced much by the magnetic field and, therefore, create no signal. After the tracker, they will create a small signal in the LAr and lose all energy in the TileCal.

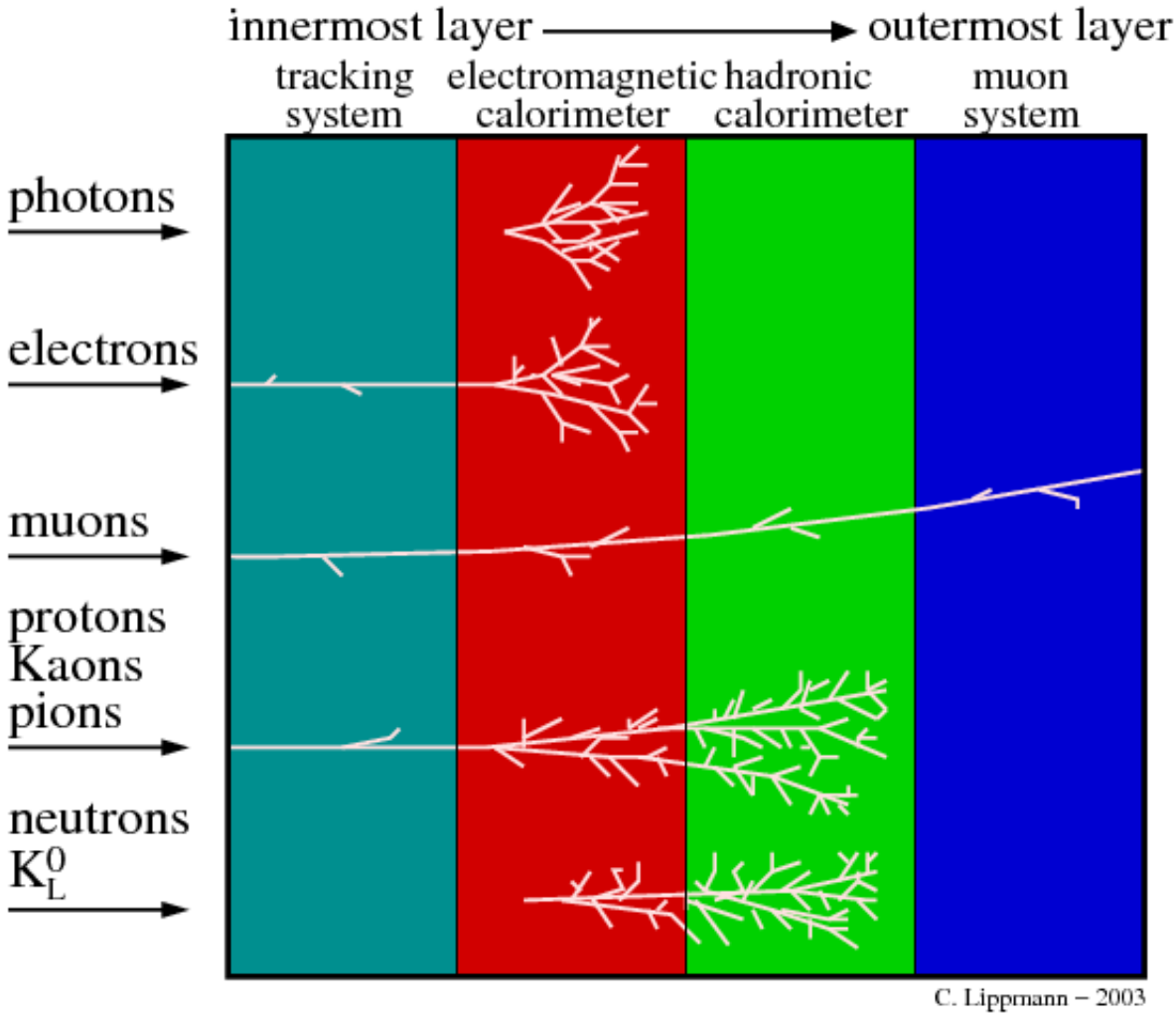

Figure 1.3: Example for the visibility of different particles in a layered detector [17].

The ATLAS detector is radially symmetric around the beam line, and mirror symmetrical around the xy-plane. To describe the position of detector parts in ATLAS, the so-called

pseudorapidity (η) is used. It is defined as:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) \tag{1.6}$$

With the polar angle θ.

1.4.1 The ATLAS Liquid Argon Calorimeter

Energy measurements are of vital importance for the analysis of particle collisions. Precise knowledge of the energy allows the reconstruction of the inverse mass of short-lived particles created during the collision and to identify processes.
Also important is to detect so-called missing transversal energy. This is a unit of energy, defined over the momenta in transversal plane to the collision. Because of conservation of momentum in a head-on collision, all the momenta should sum up to 0. Leftover momentum is called the missing transversal energy. It comes mainly from long-lived particles that cannot react via strong or electromagnetic interactions. Inside the standard model, the only particles that interact that way are neutrinos. They are created in large amounts through the decay of other particles (e.g., Tauons or W-bosons). Since they only interact via weak interactions, they will deposit no relevant energy in the calorimeters and create some missing energy.
Measuring the missing energy makes it possible to spot neutrinos and, therefore, identify some processes. Some theories of physics beyond the standard model also predict other long-lived particles that cannot be detected directly in calorimeters. If those exist, they could be detected via missing energy, which requires precise measurement in the calorimeters.
As an electromagnetic calorimeter, the LAr measures the energy of primarily electrons and photons, which lose all their energy inside it. Processes involving these particles are very interesting because many particles decay into electrons or create photons. Even the Higgs boson was discovered in 2012 by searching for $H \to \gamma\gamma$ or $H \to ZZ \to 4l$ decays. Other ionizing particles, like protons or muons, will only deposit small amounts of energy inside the LAr calorimeter.
Particles deposit energy in a calorimeter through electromagnetic interactions of the particle with the calorimeter cells. To enable these interactions, lead is used as an absorber material to initiate showers of highly energetic photons through Bremsstrahlung. These photons and energetic electrons can then ionize liquid argon in the calorimeter cells. Ionizations in these create a drop in voltage that is proportional to the deposited energy.
The LAr calorimeter system is made up of different parts, depending on the coverage of the detector. First is the electromagnetic barrel (EMB) in for the $|\eta| \leq 1.475$ region. This one is built in accordion geometry for optimal coverage. Next are two Electromagnetic EndCao (HEC) calorimeter for the $1.375 \leq |\eta| \leq 3.2$ region. Another pair of HadronicEndCap (HEC) calorimeter cover the $1.5 \leq |\eta| \leq 3.2$. Last is the Forward calorimeter (FCal) for $|\eta| = 1.475$
The voltages measured in the calorimeter are converted to a digital sequence with an Analog-to-Digital-Converter (ADC). Reconstructing the energy out of this sequence needs algorithms for the electronical readout of the calorimeter.

1.4.2 Electronical Readout of the ATLAS Liquid Argon Calorimeter

At ATLAS, proton beams collide with a frequency of 40 MHz and with up to 20 proton-proton-collisions per Bunch Crossing (BC). This results in a high amount of collected data every second. In the LHC it can reach 40 Tbs^{-1}. These data cannot be saved. Most of them are not interesting to begin with. Some parts of it are background, while others are from processes that are not interesting for analysis. For example, when searching for Higgs bosons decaying into photons, processes involving Neutrons are not interesting and act as background. Because of this, the ATLAS data are filtered and triggered so that only interesting data are recorded [18].

The Particles created in the collision will arrive nearly at the speed of light. Along their path, they will ionize atoms in the calorimeter and create charges, which leads to a drop in voltage. These charges will then slowly recombine, which reduces the voltage. This means for the electronics that triangular pulses are created in theory. These pulses have a timely length of around 25 BC. For practical reasons, the pulses are filtered into a bipolar form. An example of the pulse shaping is shown in Fig. 1.4 [19]. This shaping has the advantage that the integral over the pulse is finite, which proves beneficial for the electronics and analysis.

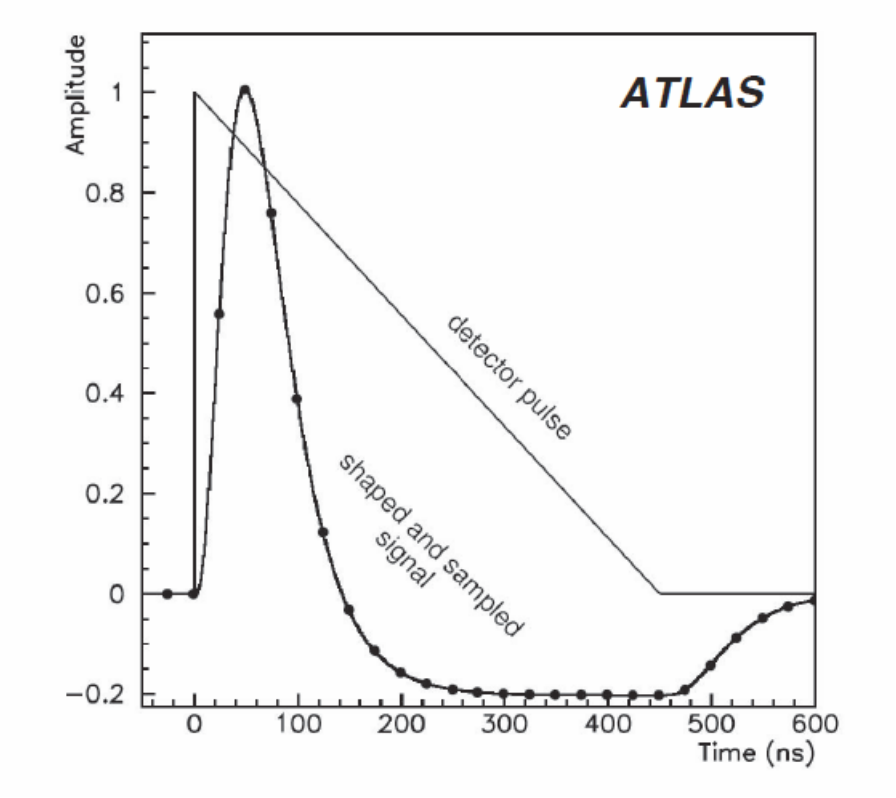

Figure 1.4: Pulse shaping for LAr calorimeter signals [14].

After the pulse is filtered, an algorithm can calculate its energy. Currently used is an Optimal Filter (OF), which consists of a pair of coefficients that are multiplied on the ADC sequence

[20]. It is defined as:

$$\mathrm{OF}(t) = \sum_{i=0}^{n-1} a_i \cdot x(t-i) \tag{1.7}$$

With the time (t) in BC and the OF coefficients a_i. Its coefficients can be calculated analytically for every detector cell, depending on the pulse shape and the expected noise. The number of coefficients can be varied. At the LAr calorimeters used are four coefficients since Run 2. Using more coefficients would be better for noise suppression, but the capabilities of the detector are limited at the moment.
It can be shown mathematically that this filter is the best linear operation to reconstruct the energy for the specific pulse form that the pulses are filtered into [20].

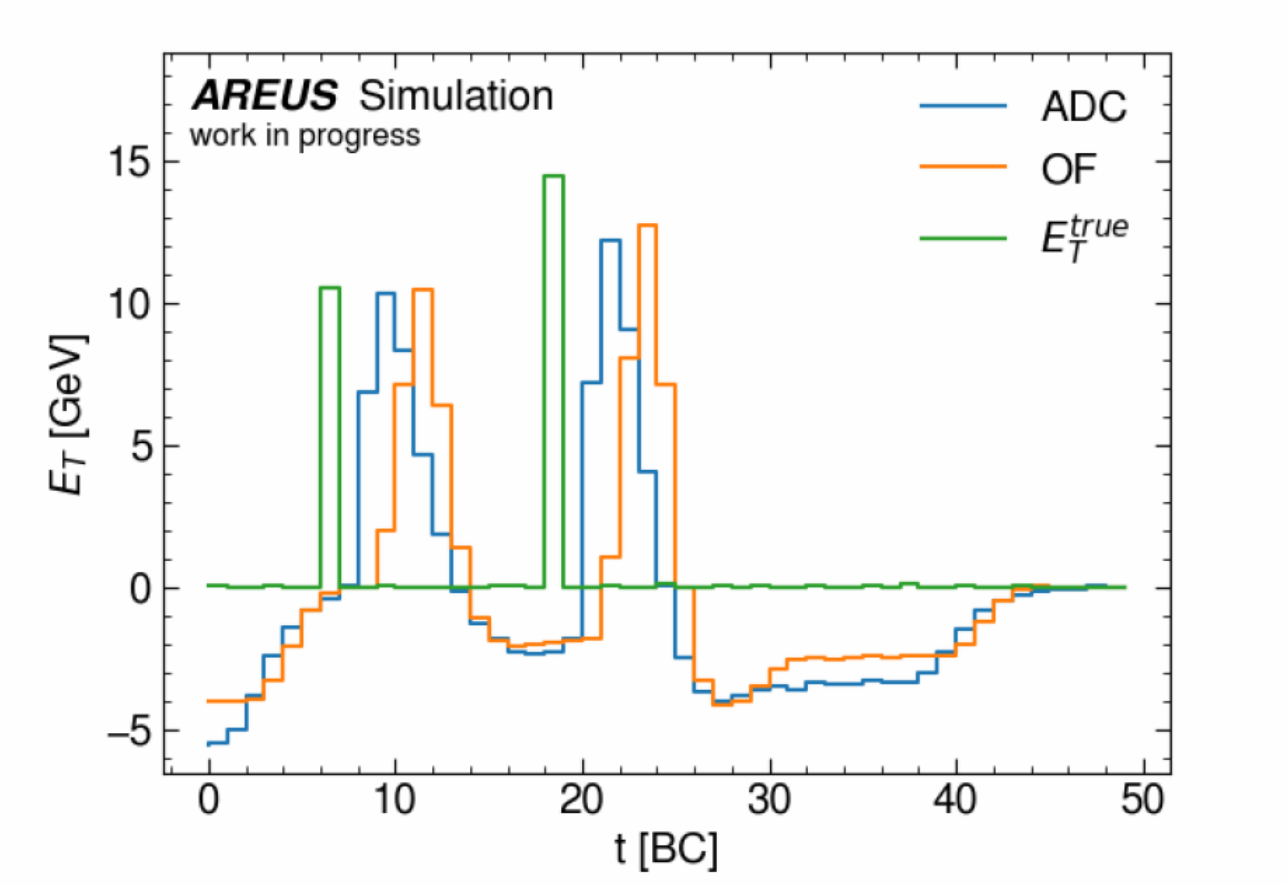

Figure 1.5: Example of an ADC sequence, the corresponding OF output and the true energies $E_{\mathrm{T}}^{\mathrm{true}}$. Hits come in the detector and deposit energy. After some BCs, the ADC has filtered a pulse, and the OF will use the pulse to calculate energy after more BCs. An underestimation of the second peak can be seen.

1.5 The High Luminosity Large Hadron Collider

During Run 2, the LHC created an integrated luminosity of 158 fb^{-1}, with a center of mass energy of 13 TeV [21]. At the end of Run-3 in 2024, this is expected to increase to 350 fb^{-1}, which results in an enormous amount of data for analysis. However, even with this amount of data, no new physics has been found at ATLAS yet. Because of this, it might be worth looking for some far-reaching improvements to the performance of the LHC.

One possibility that was discussed was to increase the energy of the proton beams. The High-Energy-LHC (HE-LHC) was a proposed upgrade to the LHC that would increase the centre of mass energy to up to 27 TeV [22]. Increasing the energy would potentially enable the discovery of new processes that are out of reach for the normal LHC because they need a higher energy to emerge, like particles with a higher mass. One downside of this upgrade would be that the cross section for processes with a lower energy would decrease. This means that rare processes accessible for the LHC are potentially missed. Also, there is no guarantee that new processes will happen at the new energy range.

The alternative was to increase the number of proton-proton collisions per bunch crossing. This upgrade is called High Luminosity LHC (HL-LHC) and was chosen [23]. It will result in a higher luminosity, which gives more data for analysis. Increasing the luminosity while keeping the energy constant means that more data of the same processes are collected. This enables the detection of very rare processes that might have gone unnoticed through the current detection. It offers a better statistics on already observed rare processes, like interactions involving the Higgs boson. This would result in a better accuracy on these and enables to study them with better precision.

The HL-LHC upgrade is currently planned to be finished in 2028, see Fig. 1.6 for the timeline. It is expected to reach an integrated luminosity of up to 3000 fb^{-1} per year.

Reaching this luminosity needs optimizations on the LHC. It requires to increase the number of accelerated protons per bunch from 115 billion to 220 billion and decreasing the phase space area of the protons.

Figure 1.6: Planed timeline for the HL-LHC upgrade [24].

1.5.1 Pileup at the High Luminosity Large Hadron Collider

Besides improvements to the accelerator itself, the HL-LHC also requires upgrades to the existing detectors of the LHC to enable them to deal with the increased number of events that will be detected. More events also mean more uninteresting events and background, resulting in many small energy deposits in the detector. For example, pions are created in large quantities during a pp collision.

One particular difficulty that will arise at the HL-LHC will be increased pileup [25]. This is a special kind of background that consists of collisions outside the main hard scatter vertex. A sketch of pileup can be seen in Fig. 1.7. There are two types of pileup. First so-called in-time pileup. This consists of many small energy deposits. The other type is out-of-time pileup. This is created when two energetic events arrive at the detector, with a short, timely distance between them. Distances between events are called gaps and are measured in BC, the time between two collisions, or 25 ns. The result of this will be an overlap of the pulses. This overlap distorts the pulse shape, which can lead to underestimated energies for events with few BC between them.

In particular, the OF has problems dealing with out-of-time pileup. This is because the OF can only correct pileup by averaging over all BC.

The amount of pileup is expected to increase drastically at the HL-LHC. Using the OF would result in a worsened energy resolution of the LAr calorimeters.

Solving this problem needs more advanced algorithms. This work uses Artificial Neural Networks.

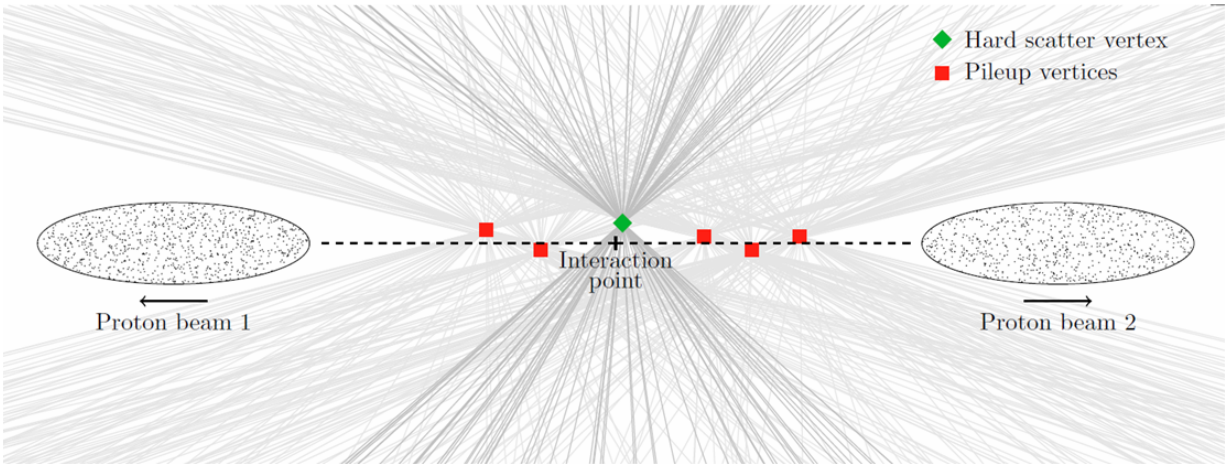

Figure 1.7: Pileup during a particle collision [26].

1.6 Artificial Neural Networks

Artificial neural networks (ANN) are connections of so-called artificial neurons that can be trained to learn functions. They are a part of machine learning. The idea behind ANNs is going back to 1943 [27]. Using them for practical applications only started in the 2000s because the computational requirements are high. Since then, ANNs have been used with great success in many applications, like language translation and speech recognition, and for scientific work, like protein folding in biology [28] or in particle physics, like at the LHC.

An artificial neuron is a structure that can take data as input, do calculations, and give some output. Fig. 1.8 shows an example of an artificial neuron. Arranging multiple neurons parallel to each other creates a layer of neurons that can do a certain task. Multiple layers of neurons

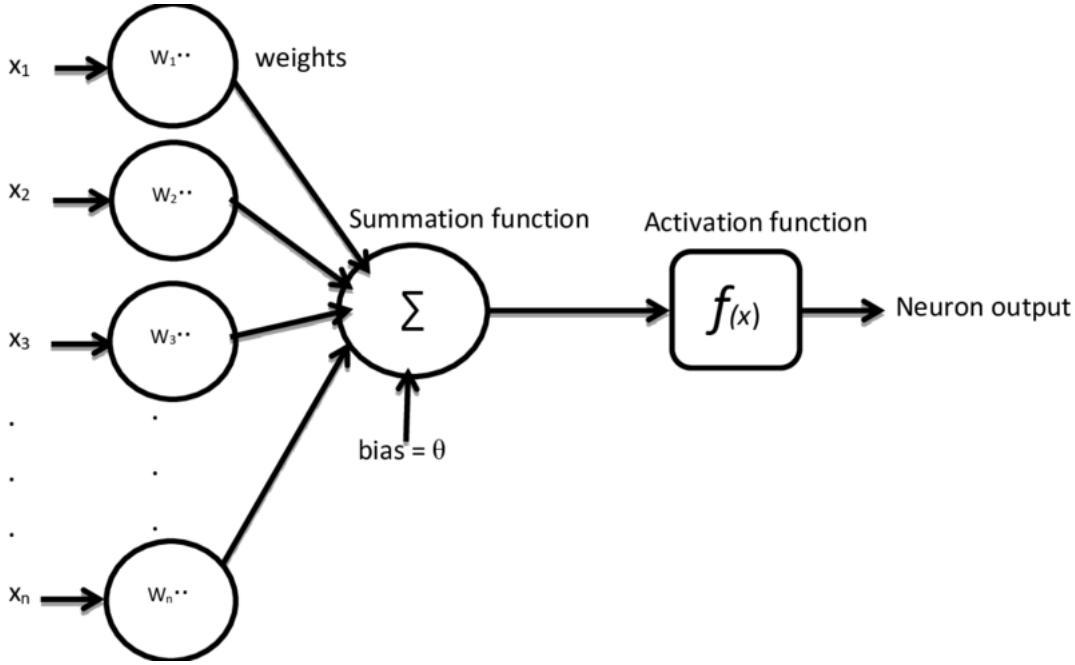

Figure 1.8: Sketch of an artificial neuron [29].

arranged after each other create a deep neural network. Each of the layers can learn to work on different successive tasks. The first layer is called the input layer, and the last is the output layer. All the layers between are called hidden layers because they are not "visible" from the outside and because it can be challenging to understand them in detail.

The main idea behind ANNs is to let optimization algorithms modify the neurons in a network. This enables the network to "learn" a function that gives a wanted output.

Artificial neural networks are very loosely inspired by real neurons and neural networks from biology. Inspired means that some techniques for neural networks, like the connections between neurons, are based on discoveries from real neurons. Real neurons can work very differently from artificial neurons, though, and it is not necessarily the goal of artificial neural networks to copy them.

1.6.1 Artificial Neurons

In order to understand neural networks and the calculations inside, a more detailed look into artificial neurons is needed. A single neuron consists of weights (w). These are numbers which are multiplied with the input (x). The product is summed up with the bias (b) and put into the activation function (φ):

$$f(x) = \varphi\left(w_i x_i + b\right) \tag{1.8}$$

An activation functions is an additional manipulation of the data to calculate the output. Without the activation function, a neuron is just a linear function of the input.

Since an ANN is a composition of neurons, using no or linear activation functions would mean that the network itself will be a linear function.

Because this would be quite limiting, most activation functions are non-linear. Using non-linear activation functions enables the network to approximate more complicated functions. In Fig. 1.9 are two common activation functions shown. The most common activation function at the moment is the rectified linear unit (ReLU), which is defined as:

$$f(x) = \max(0, x) \tag{1.9}$$

ReLU removes all negative outputs and keeps only positive ones. The advantages of ReLU compared to other functions are that it and its derivative are fast to compute, it is non-linear, and infinitely many times differentiable almost everywhere.

Mathematically, there is the difficulty that ReLU is not differentiable at 0, which can lead to problems when using gradient descent optimization on it. In practice, this is negligible because the derivative is calculated numerically. More problematic is the so-called dying ReLU problem. This arises from the fact that ReLU sets outputs smaller than 0 to 0. Since

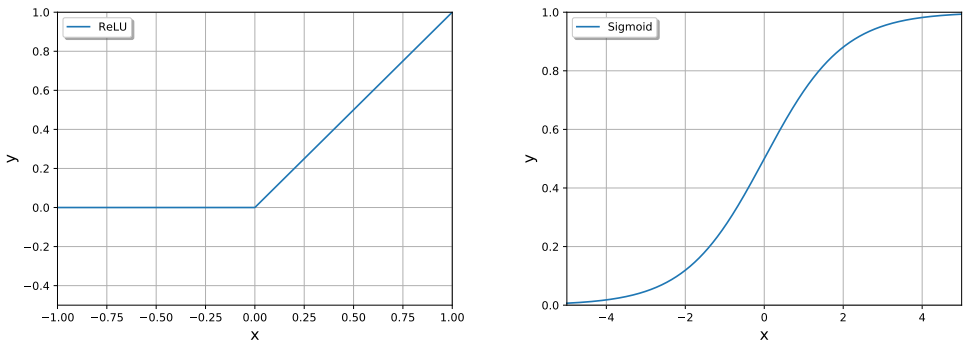

Figure 1.9: Common Activation functions for artificial neurons.

the derivative will now also be 0, neurons can get stuck at 0 as output, leading to them dying and being ignored during training. Some attempts to solve this problem include using so-called "leaky ReLU" functions [30]. These are variations of ReLU with a non-zero slope everywhere. Before ReLU, the most common activation function was a Sigmoid function:

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{1.10}$$

This function benefits classification networks, which must decide between distinct choices as output. Sigmoid functions will tend to saturate between these choices (mathematically between 0 and 1) and give a probability of these as output. The main issue of this function is that it is computationally expensive compared to ReLU. Also, the saturation, which is beneficial for classification, might be unwanted or interfering with regression networks, that have to choose between continuous outputs.
In practice, there are many other activation functions. Some of these are variations of ReLU or Sigmoid, while some are entirely different. For this work, only ReLU and Sigmoid were used.
Once artificial neurons are understood, multiple neurons can be connected. This creates an artificial neural network; a sketch is shown in Fig. 1.10.

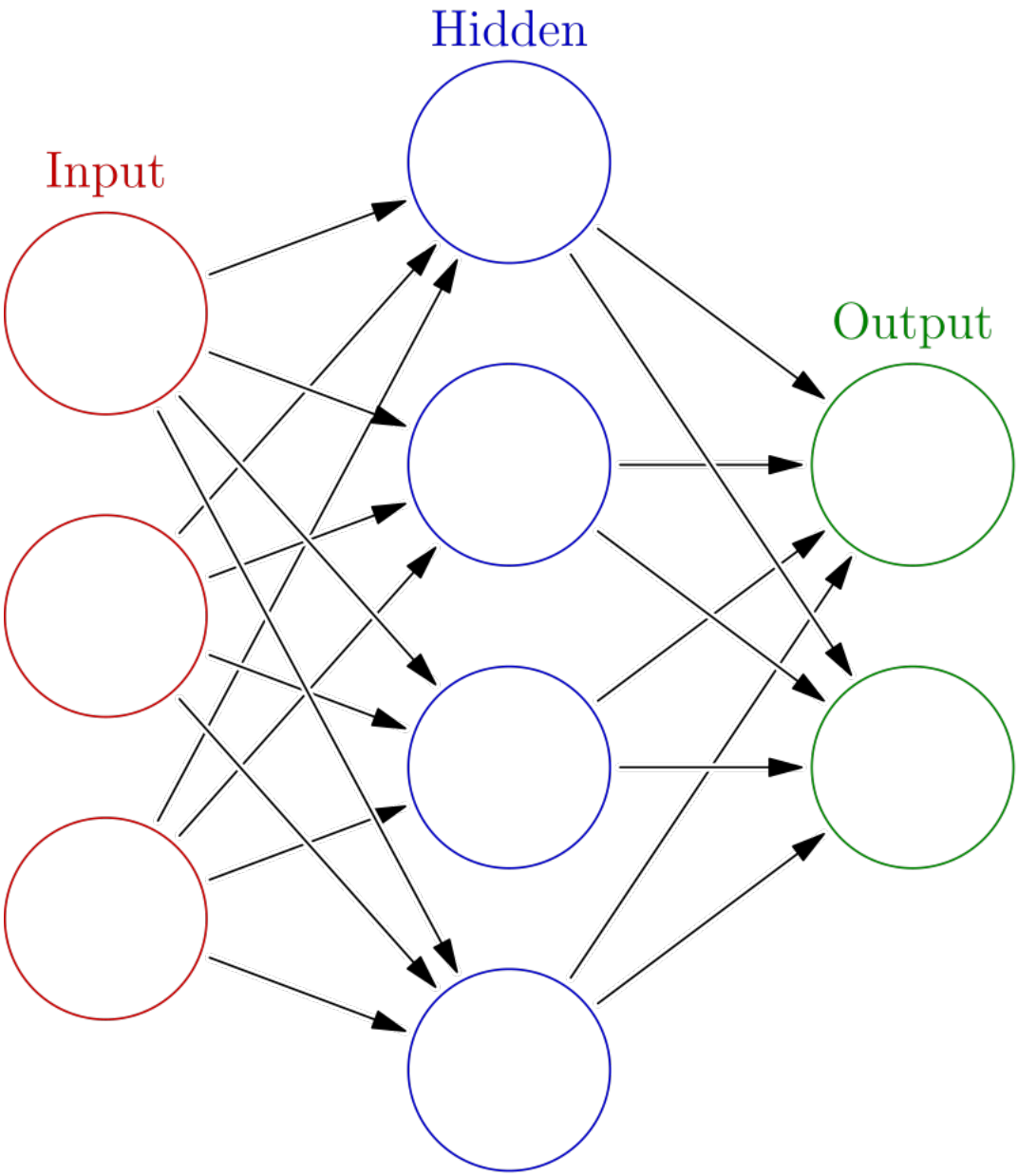

Figure 1.10: Sketch of an artificial neural network. The circles are representations of artificial neurons [31].

1.6.2 Convolutional Neural Networks

There are many ways to connect neurons. Biological neurons and their connections inspire some of these connections. It has been shown that different connections can work well on certain classes of problems. This work is focused on so-called Convolutional Neural Networks (CNN).
These networks apply a convolution operation with so-called filters on the data. Usually, they are applied on structured data and have been shown to work well on these [32]. They are most commonly applied on pictures, translating into matrices for greyscale pictures or tensors of third order if the color is taken into the network.
For the following explanation, it is assumed that the data is a matrix. The filters are typically also a matrix but much smaller than the input. Its size is called kernel and defines how much of the input every filter can see and analyze at a time. Inside the network, the filters are applied stepwise on the data and do a convolution operation (matrix multiplication in case of matrices). This operation then creates a new matrix that is called a feature map. An example can be seen in Fig. 1.11. This feature map holds information the network thinks is important for further calculations. Interpreting the feature map is a way to understand what the network is doing. In the filters are numbers, which are the weights of the neural network, that can be manipulated during training. This enables the network to adjust the filters to create feature maps that are specialized on certain parts of the data.
In this work, networks are using the LAr calorimeter's ADC sequences as input. This translates to one-dimensional convolution, which means vector multiplication. Output is then a sequence of the predicted energies deposited in the detector.

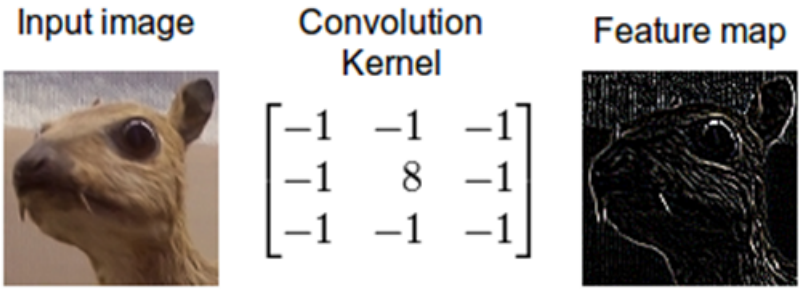

Figure 1.11: Example of a convolution operation for pictures [33].

1.6.3 Training of Artificial Neural Networks

Training a neural network means letting an optimization algorithm adjust the network to perform a given task. The architecture of the network, which means the number of neurons, layers, and the connections between them, is given and cannot be changed in training for the networks used in this thesis. This gives the algorithm the weights to adjust. In general, there are two common ways of training.

Supervised training means the network gets large amounts of input data with a known true output and tries to learn the true output. There are different ways how this can work; this explanation will focus on so-called gradient descent optimization. At the start, all the weights are set to a random value, and the network uses them on the input to calculate some output. After this is done once, the algorithm checks the predicted values of the network and the true values of the data. It will then calculate a so-called loss function to evaluate the training. The loss function is supposed to be minimized during the training. A common loss for regression networks is the mean absolute error (MAE):

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{1.11}$$

With the predicted value of the network x_i, the true value y_i and the number of samples n. This loss usually works well for a regression network that must choose between a continuum of possible values. Alternatively, the amount of wrongly tagged values can be used for a tagging network that must decide between a few distinct values (like 0 and 1). This value is called the accuracy. In practice, the loss can be problem-dependent, yet it should always be a measure of how well the network performs. It has been shown that complicated loss functions will not always lead to better performances but might even slow the training or create some bias [34].

After determining the loss, the network will change the weights by a random value and do a next epoch, which means going through all the data and calculating the new loss. Looking at the difference in loss, it can calculate the derivative of the loss in dependence on the weights. Using the derivative, the algorithm can now look for a decrease in slope and check for the location of a minimum of the loss function; see Fig. 1.12 for an example. Doing this for many epochs, the network will lower its loss progressively and end in or near a minimum of the loss function. It has "learnt" to solve the problem, according to the wanted output. This way of learning with known true outputs is also called backpropagation.

In practice, gradient descent has some difficulties. The most obvious is that it cannot differentiate between a local minimum, global minimum or a saddle point. Because of this, gradient descent will not always reach an optimal loss and tend to get stuck or jump over them. Some training algorithms combine gradient descent with some techniques to spot saddle points or local minima and escape them. For example, making bigger jumps in the weight values when at a low derivative or setting them to new random values has proven efficient. In this work,

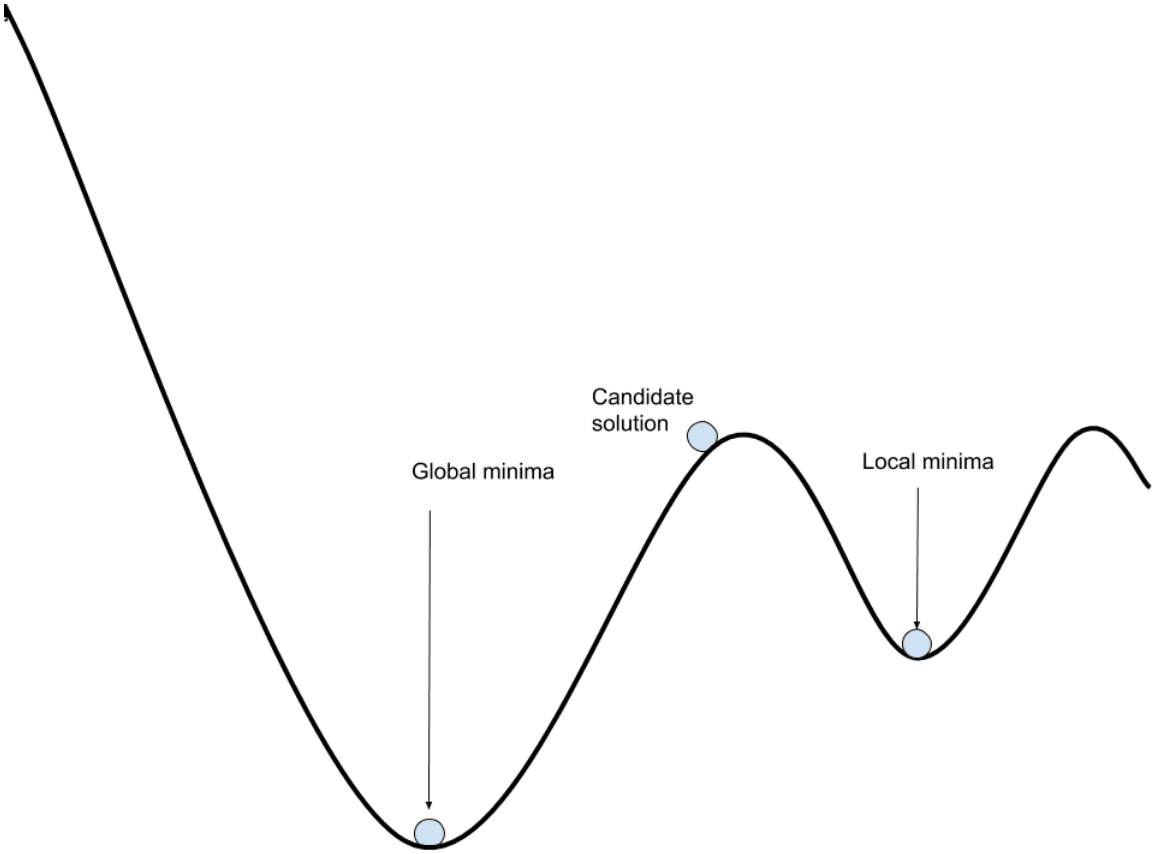

Figure 1.12: Example of a one-dimensional loss function, depending on the parameter value, with one global and one local minimum. The weights start at a random position, the candidate solution. After going through all the inputs, the network will search for a decrease in slope and either end in the global or local minimum. Loss functions from real networks with many parameters are usually highly multi-dimensional. [35].

an optimizer called ADAM was used that combines gradient descent with stochastic methods. [36]

One common alternative to supervised learning is unsupervised learning. In this case, the network does not know the true outputs (or there are no known true outputs to begin with) and will try by itself to order the input.

Now, the question is how many layers are needed for a neural network and which activation function to use. Mathematically, it has been shown that a neural network with one hidden layer and sigmoid activation can approximate every continuous function. [37] In practice, how well a network works depends on the architecture and training data. Because of this, most neural networks use more than one hidden layer. The limitation for the network size is technically only the availability of computational resources.

Even when not limited in the network size, it is not always useful to make very big neural networks. Larger networks technically have a better capacity for learning, but they also require more time for learning and are more prone to overfitting. This is a challenge in machine learning that occurs when the network has learned the training data very well but fails to generalize

to new data. It means the network has specialized too much on a specific data set and does not know other data.

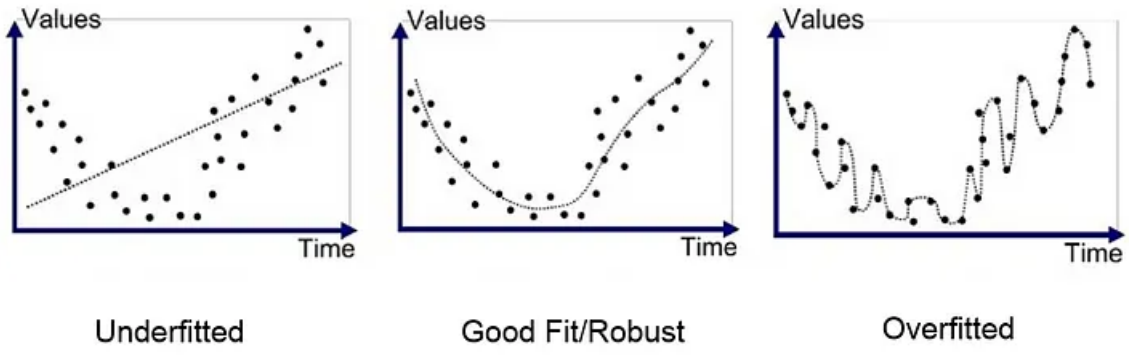

Figure 1.13: Examples of over and underfitting for a regression [38].

One example would be a tagging network that has to detect cars in a picture. If it is overfitting, it might have been trained only on data with red cars or cars seen from a certain angle. Then it can only detect exactly these cars and no others. Dealing with overfitting can be done in different ways. First, choosing training data with a distribution that includes every possible situation is one way to deal with it, but it is difficult to include every possibility.
Another way is to split the data into two parts. Common is a split into training data for training and calculating the loss as described and the validation data. These data are not used for the learning algorithm but to calculate an additional validation loss. If the network tends to overfit, the normal loss decreases while the validation loss, on which the network is not trained, increases. This method does not prevent overfitting but enables spotting an overfitted network early on. Also, the training time has to be kept in mind. Long trainings will likely keep decreasing the loss function and let the network learn the training data but hold the danger of overfitting because the network has too much time on the same data. In this case, it might learn some specific noise of the training data.
The opposite problem of overfitting is underfitting. This occurs when the network is not even able to learn the training data. It usually means that the network is too small, does not have an architecture that can solve the problem or was trained not long enough.

1.7 Implementation of Artifical Neural Networks on FPGA

The ANNs trained in this work will be used at the ATLAS LAr calorimeters. For this application, they must be implemented on hardware close to the detector. The used hardware will need to be able to run neural networks and withstand the great flow of data at ATLAS during operation. Field Programmable Gate Arrays (FPGA) were chosen for this. These are integrated circuits that can be reprogrammed after manufacture.

FPGA are made up of an array of programmable logic blocks. An example of a logic block is shown in Fig. 1.14 for a sketch. Connections between these blocks can be reconfigured by using switches between them. The FPGA as a whole and every block inside have an input and an output port (I/O-port). Using logical functions and connections of logic blocks makes it possible to manipulate the input signal and create complex functions. Implementation of the networks is done in VHDL, which is a hardware description language. Letting the networks run on FPGA creates a logical function that takes the ADC sequence as input and gives the energy as output. A more detailed description of the technical implementation of the networks can be seen in [39].

The main advantages of FPGA over other systems are their ability for good parallelization and that they can work well with very high input and output rates of data. These abilities are needed at the HL-LHC, which makes FPGA suited for this work. Also, the fact that they can be reprogrammed at will offers great flexibility for designing them and reduces the risk of software errors during operation. If the numbers for production are low, as for this work, they are also cheaper than ASICs (application-specific integrated circuits). Disadvantages of FPGA are, e.g. that they are slower than ASICs and usually have a higher power consumption.

The LAr calorimeter will use the Intel Agilex 7, which can be seen in Fig. 1.15.

One challenge when letting neural networks run on FPGA is resource consumption. FPGA are quite limited in their computational resources.

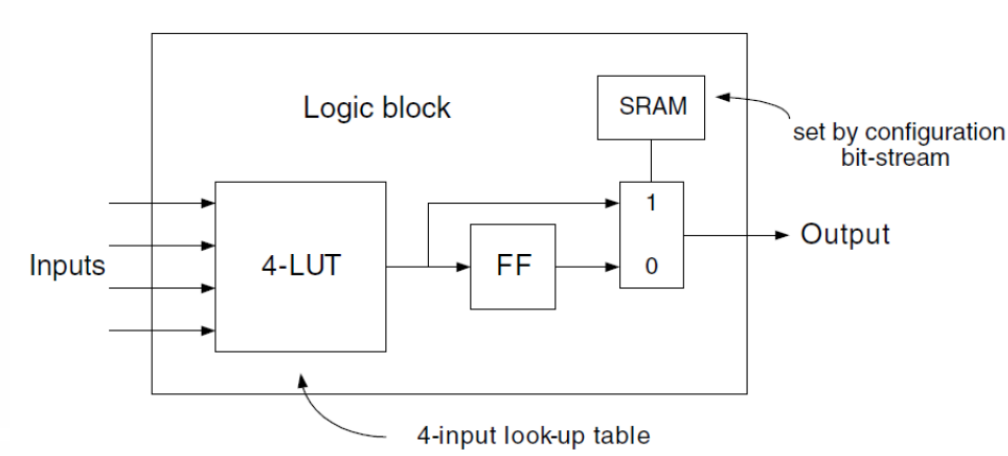

Figure 1.14: Sketch of a programmable logic block [40].

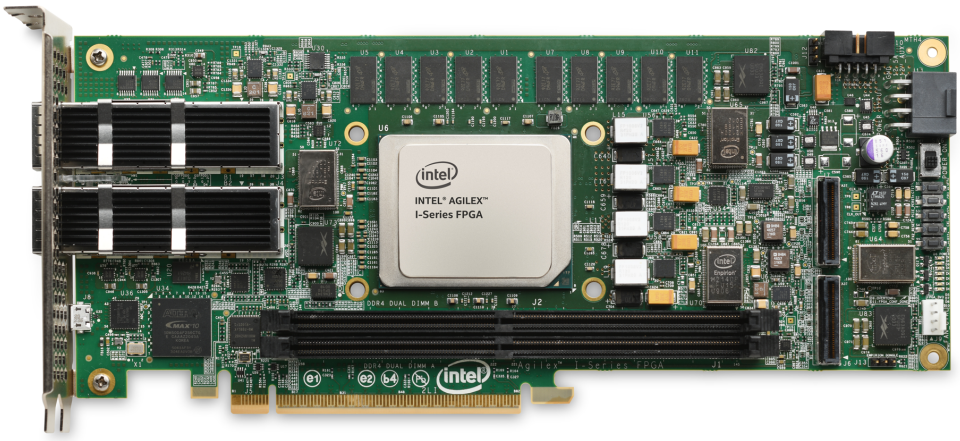

Figure 1.15: The Intel Agilex 7 FPGA which will be used at the ATLAS detector [41].

The most important resources for this work are the adaptive logic units (ALMs) and digital signal processors (DSPs). DSPs are needed for the multiplication of numbers. This resource depends on the number of parameters used in the network. ALMs are logic units that are used to represent the circuit layouts. Their usage depends on many factors, including the bit width of numbers in the FPGA. The power consumption is also limited because of cooling during operation.
The exact numbers of consumption depend mainly on the following:

- The number of parameters. These are the weights and biases in the network. More parameters mean that more ALMs and DSPs are needed for calculations. It also increases the power consumption. This number depends on the architecture of the CNN and cannot be influenced after training. The number of parameters must, therefore, be constrained during the optimization of the architecture. It was estimated in the beginning that 100 parameters should satisfy the consumption.

- Quantization of the network. The fewer bits the network uses for calculations, the less ALM is needed. This can be influenced by trainiing the network at a given bit range.

- Technical implementation of the FPGA. There are different ways to implement a network on the FPGA. Some methods are more efficient for resource consumption. This is a task for the software development.

- The number of layers. Technically, this is not a hard limitation, but the framework will need to be optimized for a certain number of layers.

In 1.16, can an early estimation for the consumption of one neural network on one FPGA be seen. This prediction was one motivation for the quantization but is outdated because of

optimized implementation in parallel to this work. An updated estimation can be seen in Fig. 3.1. During operation, 33 neural networks will work on one FPGA, so it can be expected to multiply these numbers with around 33.
It is important that consumption should not go over 80% of the available resources. If it exceeds this limit, the compiler will run into problems. Less consumption also makes the software implementation easier and leaves more space for other software. Also, full resource consumption is not wanted because other software will be implemented parallel to the CNNs on the FPGA.
A set distribution of the resources for the different software that will be implemented would be desirable, but this was not finally decided at the time of this work.
One factor that is influencing the consumption of ALMs and power is the bit width of numbers in the neural network. Keras uses 32-bit floating point numbers for calculations when training the networks. The FPGA could technically be programmed to work with those too, but is optimized for 18-bit fixed point operations. Also, using full 32-bits is not necessarily needed for a good performance and might waste resources that could be saved. Reducing the resource usage of a network would be beneficial for implementation and would enable the saving of resources on the FPGA that can used for other software or larger networks.
One possibility to reduce the bits in the network is the quantization of the networks.

Network	f_{max}	ALMs	DSPs	Latency
4-Conv	487 MHz	16698 (1.8 %)	42 (0.7 %)	150 ns
3-Conv	423 MHz	21256 (2.3 %)	46 (0.8 %)	125 ns

Figure 1.16: Early prediction for the resource consumption of one CNN with 100 parameters on an Agilex FPGA. [40].

1.8 Quantization of Neural Networks

Quantization of neural networks is a technique to reduce the bit width of numbers in the network. Reducing the bit width of networks decreases the memory needed to store the weights, lowers the number of operations for a single calculation and therefore reduces the consumption of resources, reduces the time needed for calculations and decreases power consumption.
In principle, there are two different ways to achieve this:

- Post-training quantization rounds the weights or data to a given bit width after the training is finished. This is easy to implement and usually works well for light quantizations. However, cutting off too many bits from an already-trained network will likely result in a drop in performance. This technique was used before on networks for testing and showed deviations between the FPGA and training. These deviations would result in errors during operation.

- Quantization aware training is an alternative that reduces the bit width during the training. It works by rounding weights or data down to the given bit width after each epoch. This forces the network to work with the given bit width and lets it learn to use the bit range while keeping a good performance. A disadvantage of this is that it requires additional work for the training and can lead to unstable trainings since the reduced bit range can interfere with calculating the loss function. It will also increase the time needed for the trainings because more calculations are needed for the quantization.

Implementing the networks for ATLAS on the FPGA requires a reduction from 32 bits from Keras to 18 bits in the FPGA. Since this is a relatively big decrease in the bits, post-training quantization is not suitable for this work. For this reason, quantization-aware training was tested in this thesis.

1.8.1 Quantization Aware Training

In this work used was QKeras as a framework for quantization-aware trainings [42]. This is a tool to reduce the bit width of numbers in neural networks.
Numbers for computations that are not constrained to integers are floating-point numbers. These are a subset of the rational numbers. They consist of 1 sign bit, bits for the integer (called exponent) and fractional bits (called mantissa). Floating point numbers take a given amount of total bits and can vary the different parts in their bit sizes. Keras uses 32-bit floating point numbers to calculations and save the weights.
On the other side, fixed point numbers use a fixed amount of bits for the integer and fractional. This can be computationally beneficial if the numbers are known to be in a certain range or limited in size because fewer bits are needed. They have the disadvantage of less precision for calculations because more rounding can be needed.

Reducing the bit width of a floating point number to convert it into a fixed point number is called quantization. In QKeras it works by applying the quantized bits function:

$$\text{quantized\_bits}(x) = 2^{\text{int}-\text{b}+1}\text{cliround}(x) \cdot 2^{\text{b}-\text{int}-1}, -2^{\text{b}-1} - 1) \tag{1.12}$$

This function takes a given amount of total bits (b) and integer bits (int). The function's input will then be quantized by rounding its integer and decimal bits to the given range. This operation is applied during training on each weight, bias and activation function after each epoch.

QKeras can alternatively use other quantizations involving stochastic rounding or integer-only quantization, which were not tested in this work.

To give an example of how quantization works, the quantized ReLU activation is shown in Fig. 1.17. As a reminder, ReLU was defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{1.13}$$

Quantization of ReLU turns it into QReLU function; see Fig. 1.17. This is technically a step function with a cutoff. Its cutoff is depending on the integer bits. Values above it are rounded back to the maximal possible value. Values under 0 are set to 0. The quantization is for values between 0 and 1. Here, every value is rounded to the closest floating point number in the given range, which creates a step function.

Mathematically, the derivative of QReLU is 0 almost everywhere, except for the jump points where it is not defined. For quantization to very low bits, this can lead to problems with the backpropagation. On large bit widths, this problem is negligible because the derivative is calculated numerically. (Even without quantization, the networks are technically quantized to 32 bits.)

As said before, going down to 18 bits is needed for FPGA implementation. However, there is no reason to stay at 18 bit. If the network can achieve a good performance with fewer bits, then the saved bits would decrease the consumption of computational resources and power if the software is modified to work with fewer bits. This, in turn, might allow the implementation of bigger neural networks with a better performance. Alternatively, the saved resources could be used to implement very efficient networks with more space for other programs on the FPGA.

On the downside are also some challenges that occur when using quantization.

- The performance can degrade. Having fewer bits available for calculations can lead to a drop in performance because numbers will be less precise inside the network and need to be rounded. This drop will usually become worse the less bits are available. Compensating this with an architecture that is optimised for the wanted bit width would be ideal, but it was not available for this work.

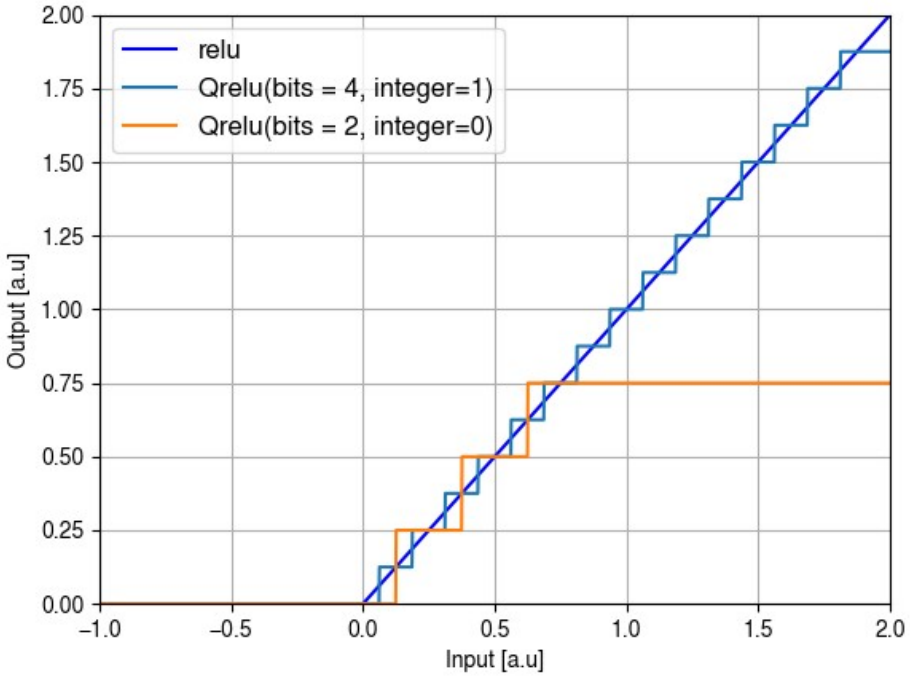

Figure 1.17: Quantization of the ReLU function for different bit widths

- Trainings will need more time. First, because every epoch of the training needs more operations for the quantization, but also because quantized networks tend to get stuck during training and not converge their loss function. This, in turn, means more trainings are needed for the best results.

- Oscillations of parameters. This means that the weights do not converge against the optimal value but jump around it because they are outside the bit range. These oscillations are a common problem for quantization-aware trainings. They technically also occur in normal trainings, but they become worse the fewer bits are available. An example would be a binary network (binary means that all weights only use 1 bit). The global minimum of the ls function may be at a parameter value of 0.5. This value is not reachable for a binary network. What would follow during training is that the network will try to get a value of 0.5. The quantization will, however, force the network back to 0 or 1. This leads to an oscillation between both values, while neither leads to optimal performance. A possible solution would be to use another architecture optimized for lower bits. Other ways to deal with oscillations during the training are currently under research [43].

2 Convolutional Neural Networks for the ATLAS Liquid Argon Calorimeter

2.1 Training of Neural Networks

In the following part, neural networks will be trained quantization aware to reconstruct the energy from LAr calorimeter signals. Very important for the training of neural networks are the data. As data for the trainings and evaluation, low gain ADC sequences of the LAr were used that were simulated with AREUS [44]. AREUS is a simulation tool for the electrical readout of the LAr calorimeter. The low gain data sequences are made up of 16-bit integer numbers. There are also high gain data from the ADC, but those were not tested at the moment because they have a different pulse shape and must be treated separately. Networks in this thesis were trained and evaluated on the EMB $\eta = 0.5125$x0.0125 cell. Training networks on other cells will be done in the future. First evaluations on the performance for other cells were done for not quantized networks.
This thesis uses convolutional neural networks (CNN). There is also research on recurrent neural networks for the LAr calorimeter [45]. These are networks that simulate memory for the neurons, using the output of the last epoch as an additional input for the current epoch.
Since resources on the FPGA are limited, there is a need to constrain the consumption. At the moment, the FPGA framework is optimized for networks with 100 parameters and up to four layers.
Different architectures within this architecture were optimized with KerasTuner. ([46]). This is a stochastic optimization algorithm that tests different combinations of parameters within a given set of layers. For the general architecture, two different types of networks were tested.

- First normal convolutional neural networks. These are convolution layers that calculate the energy out of the ADC sequence. They are called nCNN_m, where n is the number of layers and m is the number of parameters. The activation function on all these networks is ReLU.

- As an alternative, tagging convolutional neural networks (TCNN) were also trained. These are a combination of two convolutional networks. First is a network for tagging with one or two layers. These networks are pretrained and supposed to learn the position of energies in the input sequence. Since the first part is a tagging network, whose output is probability between 0 and 1, they use a sigmoid function for activation. This probability, together with the original input, is then given to the second network, which is again a convolutional network with two layers that learn the energy. They are called nTCNN_m, where n is again the total amount of layers and m the parameters. The idea behind this network was that knowing the position of energies might enable the network to easier differentiate between close hits. One problem for this network is that the limited

number of parameters means that the energy reconstruction might suffer because these parameters must be shared between two networks. Using a Sigmoid function is also computationally expensive.

Both Network principles were tested with different architectures. An example of a TCNN architecture can be seen in Fig. 2.1.
The best performance was found in 2CNN networks. These networks could outperform the OF in terms of energy reconstruction and achieve an increased performance on overlapping hits. It could also be seen that CNNs generally performed better than TCNNs. Tagging networks had trouble outperforming the OF in the overall energy resolution. They still performed well on overlapping hits, however. Deeper CNN architectures, which means 3CNN and 4CNN, also performed worse than the 2CNN architecture. Later, new networks will be tested with these architectures and increased numbers of parameters. More information on the different networks and training can be seen in [39].

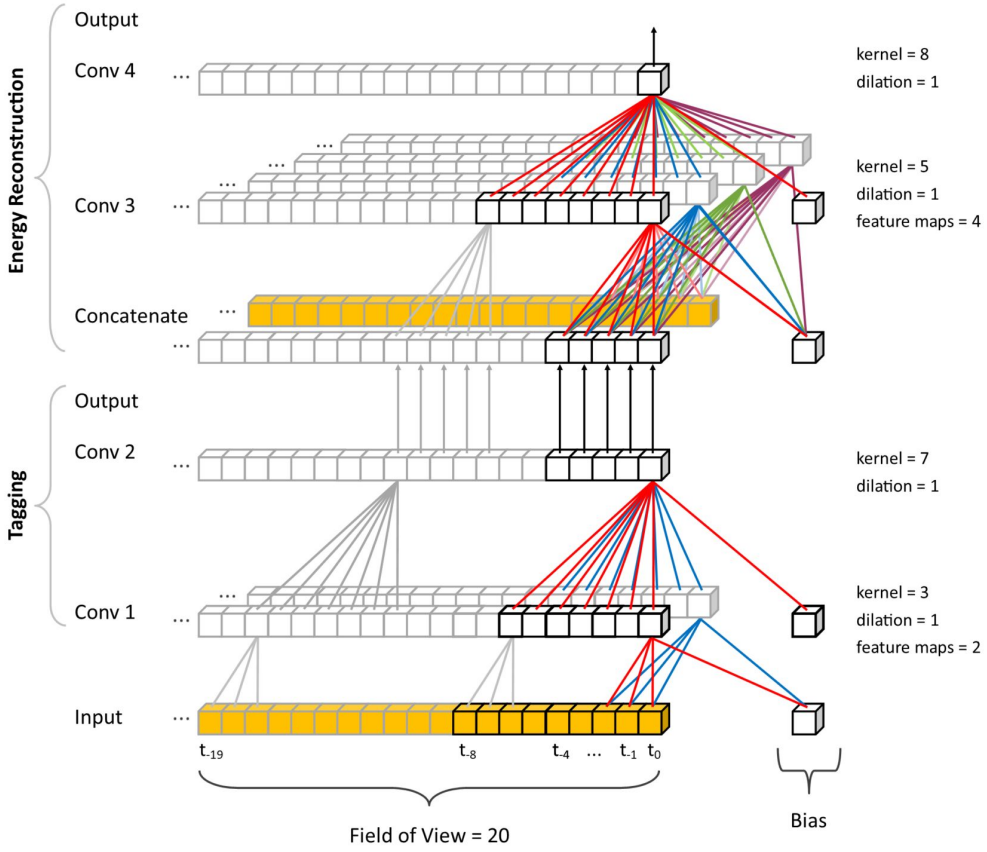

Figure 2.1: Sketch of the layers of a 4TCNN. The input ADC sequence first goes into the two-layered tagging. Its first layer creates two different feature maps. The tagging has a field of view of 20, which means that it can see 20 ADC values at any time. After the tagging, every ADC value has a probability if it shows a hit. These probabilities and the sequence are combined in the concatenate part for the energy reconstruction. In this part, two layers will use the new input to calculate the energy. A normal CNN network would skip the tagging and only use layers for the calculation of the energy.

2.2 Performance of Convolutional Neural Networks with 100 Parameters

The architecture of the best network can be seen in Fig. 2.2. It consists of two 1-dimensional convolution layers and two ReLU activations, with 99 parameters.

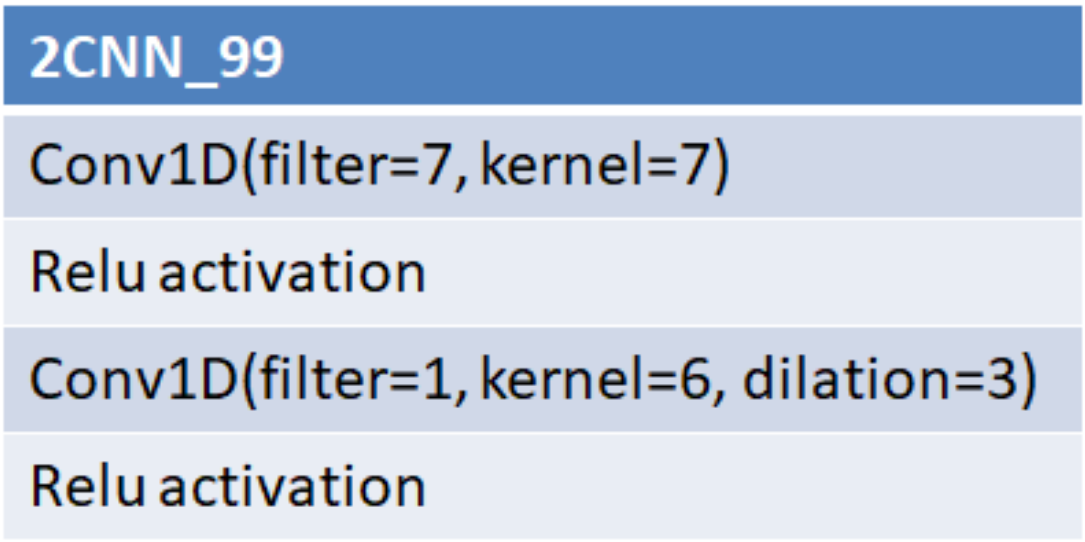

Figure 2.2: Architecture of the best performing 2CNN network limited to 100 parameters, the 2CNN_99.

This network uses a technique called dilation in its second convolutional layer. Dilation is a method to artificially increase the field of view of the network. The reason for this is the limited number of parameters. 25 BC is the total available field of view. If the network wants to use this fully, the kernels in the layers would need to sum up to 25. While limited to 100 parameters, large filters would reduce the available number of filters.
Dilation can now be used to increase the field of view without adding more parameters. It works by delaying a layer for a given number of inputs, in this case for some BC. This increases the field of view but desynchronizes the different layers. During the optimization of the architecture, the network had the choice to use dilation for its layers. Many of the good working networks used dilation in the last layer. This shows that having a full field of view benefits the performance.
Before this thesis, the networks were trained in normal Keras, meaning that 32-bit floating point numbers were used. The following will show the performance of the not quantized networks.
To evaluate this, first, the training will be explained. The data for the training used were simulated LAr calorimeter ADC sequences, normalized to the range of $[-1, 1]$ as input and output. This normalization was done because it has been shown that networks can perform better when using normalized data [47]. Normalization was done by dividing every input

through the maximal possible ADC value before the training. Output was then again a normalized ADC value. This output is now directly proportional to an energy, which can be directly calculated from the LAr ADC design. Letting the network use real energies in GeV as output would also be possible. However, using only ADC values in every layer is easier for FPGA implementation.
The simulated data can be grouped into different sets depending on the distance between two hits:

- Random gap data have a Gaussian distribution centered around 30 BC.
- Uniform gap data have a uniform distribution between the hits from 1 to 80 BC.
- Constant Gap data are simulated with exactly 30 BC between all hits
- Only pileup data include no real hits but only pileup. They serve as background to let the network learn to differentiate between true hits and pile up.

Training was done with a mixture of all data sets. The training data were also randomly split before every training. This split used 80% of the data for training and 20% as validation data to spot overfitting.
The loss function used for the energy reconstruction was the mean absolute error (MAE) of the predicted output with the true output. This loss was modified by weighting hits with an energy over 1 GeV with a factor of 30. Those highly energetic hits are more interesting for analysis, and it is therefore important that the network can reconstruct them well. The relative amount of hits with hose energy is also low compared to input with low or 0 energy. With the modified loss, the network is forced to concentrate on the important hits. The cut of 1GeV was chosen arbitrarily at the moment and kept because it gave good results. In the future, taking a different cut for the energies might be useful, based on the mean energy value of pileup, background, or electronic noise. Other losses, e.g., more complicated ones, have been tested before, but the MAE gave the best results.
Trainings for this thesis were done for 100 epochs with a technique called early stopping. This means that if the loss function does not decrease for a certain amount of epochs (15 in this thesis) the training stops and uses the weights before the loss stopped decreasing. The goal is to prevent overfitting and reduce the time for training by eliminating time that would not improve the performance further. One possible disadvantage was that training's that would have needed more time to improve, or some improvements that needed more than 15 epochs, were also disregarded. To cope with that and with statistical fluctuations that influence every training, the not quantized network was trained 100 times, and the best performing results were evaluated.
The best performance was chosen based on the lowest reached loss value. Lowest loss, which means the lowest mean absolute error, does not necessarily mean the best performance since

other measurements, like the performance on close hits, are not taken into consideration. The loss was still taken as a measurement because it is an easy and single value to measure performance and because the networks are optimized during training at this value. A low MAE also means that the network will have an overall good performance.
Different performances were looked into for further evaluation, shown in the following plots for the best-trained network. The plots are calculated on ADC sequences using only random Gap data. A performance comparison on different data sets can be seen in section 3.6.
As a baseline and for comparison, first is the performance of the OF shown. In 2.3, the distribution of the reconstructed energy residuals can be seen. On the left are energies below 1 GeV, and on the right are energies above 1 GeV. Energies shown in this thesis are transversal energies unless said otherwise. These plots show two areas. The bright and dense line at the center shows the resolution of the network. It shows a Gaussian distribution of the energy residuals. Outside of it are outliers. These mostly come from overlapping hits, but singular hits can also be falsely reconstructed.

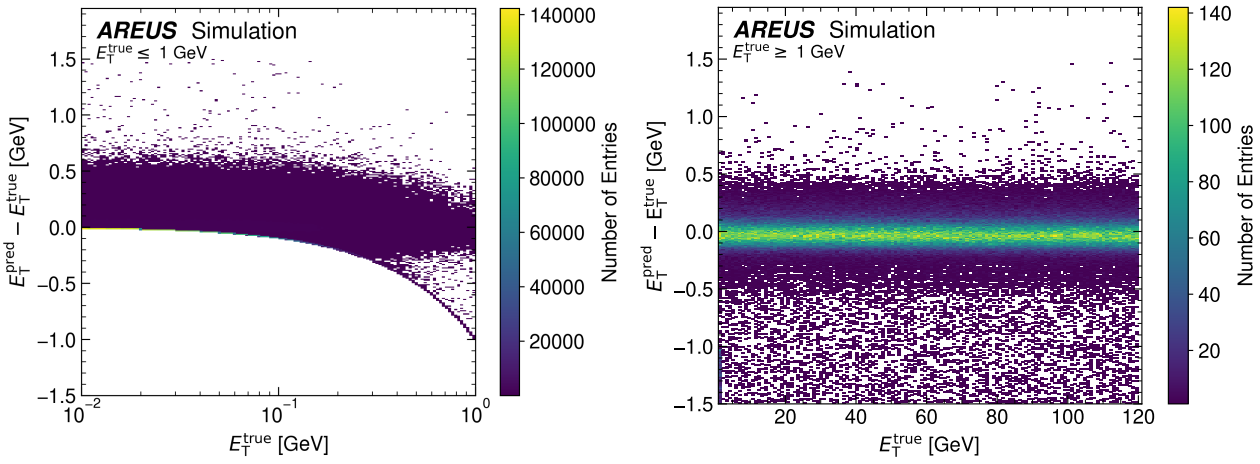

Figure 2.3: Differences of the energies predicted by the Optimal Filter with the true energies.

In the low energy range, the performance of the OF is very good. Its distribution here is without any regions of error, and the resolution is very high. The cut-off for negative residuals that can be seen comes from the fact that ADC values cannot be negative. Because of this, the OF does not give out negative values. Instead, such values are set to 0, creating the line in this plot. For high energies, the OF performs well; a linear distribution that is constant for all energies can be seen. The amount of outliers of the OF is also constant for all energies. One problem that the OF has is the large amount of underestimations, which means outliers in the negative y-axis. These arise from the high amount of pileup in the HL-LHC. This increased pileup leads to an increased amount of undershoots that the OF cannot deal with. It will, in turn, underestimate the energy of those subsequent hits. Even ignoring the outliers, the OF already has some bias towards underestimating, that can be seen. Its distribution is not

centered around 0 but in the negative, and the outliers are asymmetrical, which means that underestimations are more common than overestimations. This is another point that could be improved with neural networks.
Based on this, the mean value of these energies with standard deviation and the median with 98% range was calculated for different energy ranges and is shown in Fig. 2.4. Mean value and standard deviation are always calculated with a Gaussian fit over the energy in this thesis. The 98% range is calculated by ordering the residuals and removing the largest and lowest 1% of the distribution. This means that 98% of all reconstructed energies are inside this region. It serves as a measurement for the number of outliers outside of the Gaussian center.
This plot shows again the large amount of underestimations by the network and that the performance is constant for all energies. The upper range of the OF is at a constant value of around 0.4 GeV and the lower at -18 GeV for all energies above 1 GeV. Its standard deviation is ± 0.083. Variations for different energies are not expected for the OF when using uniform distributions of energies.
Another performance measure that needs to be checked is the performance of two overlapping hits. The overlap distorts the pulse shape and can make the reconstruction harder. This performance is seen in Fig. 2.5. Here are the energy residuals plotted against the timely distance between the hits measured in BC. The Optimal Filter performs well for hits with a higher distance than 30 BC. It has a dense distribution with a low amount of outliers and no large errors. This is the region for which the OF was optimized.
The OF is losing performance for hits between 20 BC and 30 BC. Here, the hits start overlapping, which does not decrease the density of the distribution much, but it creates error structures and leads to overestimations of energies. The OF can still reconstruct the hits in this region but with some errors. One problem that the OF has in this region is that while the density in total stays relatively constant, the OF gets a bias toward overestimating the energies. This bias changes fast with the energy, creating a non-linear error structure at around 25 BC. The OF mostly creates undershoots for closer hits under 20 BC, and the resolution drops drastically.

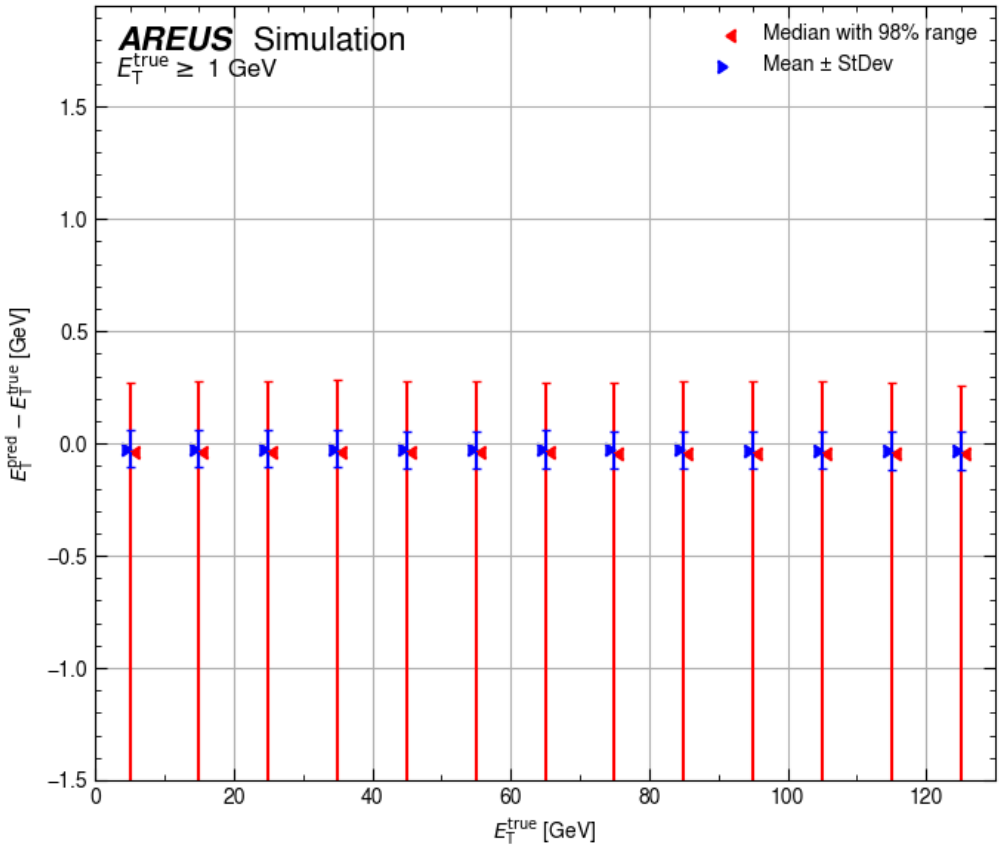

Figure 2.4: Mean value with standard deviation and median with 98% range for differences of energies reconstructed with the OF with the true energies. The full lower 98% range goes down to around -18 GeV and was not shown for better visibility.

Another performance measure that needs to be checked is the performance of two overlapping hits. The overlap distorts the pulse shape and can make the reconstruction harder. This performance is seen in Fig. 2.5. Here are the energy residuals plotted against the timely distance between the hits measured in BC. The Optimal Filter performs well for hits with a higher distance than 30 BC. It has a dense distribution with a low amount of outliers and no large errors. This is the region for which the OF was optimized.
The OF is losing performance for hits between 20 BC and 30 BC. Here, the hits start overlapping, which does not decrease the density of the distribution much, but it creates error structures and leads to overestimations of energies. The OF can still reconstruct the hits in this region, but with some errors. One problem that the OF has in this region is that while the density in total stays relatively constant, the OF gets a bias toward overestimating the energies. This bias changes fast with the energy, creating a non-linear error structure at around 25 BC. For closer hits under 20 BC, the OF mostly creates undershoots, and the resolution drops drastically.

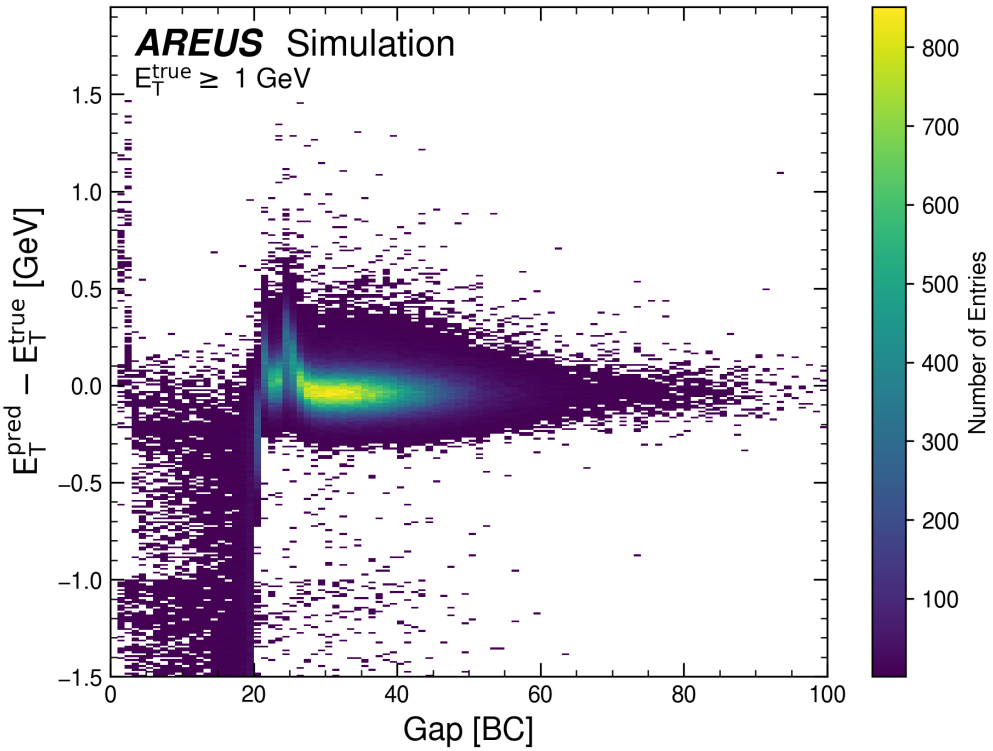

Figure 2.5: Differences of energies predicted by the OF and true energies, plotted against the distance between two hits in bunch crossings.

Now, a comparison with the best-trained neural network, the 2CNN_99, will be made. The main performance goal for the neural networks was to improve the performance for close hits and keep the OF performance for larger gaps.
Its performance can be seen in Fig. 2.6.
First, the general energy reconstruction performance will be rechecked. In the low energy range, the performance in superior to the OF. It can be seen that the distribution is more dense with fewer outliers, which means it has a better resolution and is still constant for all energies.
For higher energies, the bias is visibly reduced. The CNN is centered closer to 0, which is an improvement. It's resolution of the network is also similar to the OF. Overall, the distribution is more symmetric compared to the OF. While the OF had a tendency for undershoots, the 2CNN_99 has visibly fewer undershoots but more overestimations on the other side. This is still an improvement; a symmetrical distribution has advantages for analysis. One difficulty of the 2CNN_99 is that the distribution of outliers is not constant for all energies, There is an error structure seen at an $E_{\mathrm{T}}^{\mathrm{true}}$ of around 20 GeV. This structure was seen on some 2CNN networks but not on other architectures. For other energies, it is primarily constant with only slight variations. The resolution, on the other side, is constant for all energies.
Fig. 2.7 shows again calculated measurements based on these plots. The standard deviation of the network is constant for all energies at a value of ± 0.086 GeV, which is slightly worse compared to the OF. The upper range of the network is at a mean value of 0.49 GeV for energies above 30 GeV. This range is increasing at lower energies, with an upper limit of 1.2 GeV for energies between 10 and 20 GeV. These values are slightly larger compared to the OF. The lower range, on the other hand, is at a mean value of -1.3 GeV for all energies, but with slight variations. For lower energies, this range is increasing. Compared to the OF, this is a large increase in performance. The reason for this increase is the better reconstruction of overlapping hits.

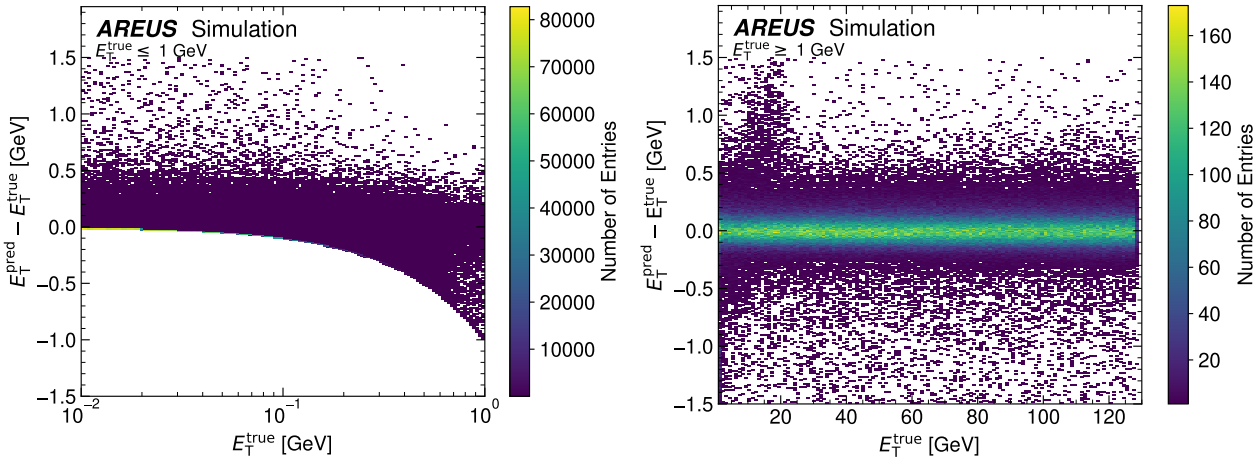

Figure 2.6: Differences of the energies predicted by the 2CNN_99 with the true energies.

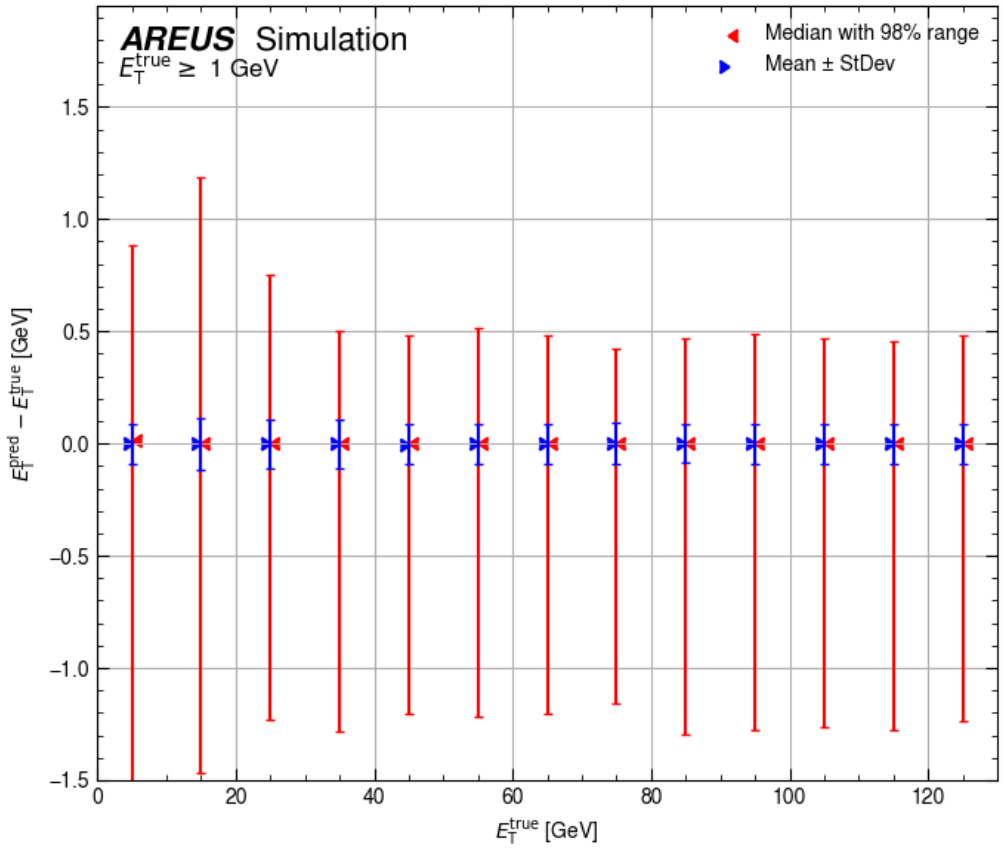

Figure 2.7: Mean value with standard deviation and median with 98% range for differences of energies reconstructed with the 2CNN_99 and the true energies.

The performance on these hits is shown in Fig. 2.8 For distances over 30BC, it is very similar with no visible difference to the OF. The density is equal, and the amount and distance of outliers are similar. It is interesting to note that the CNN shows no improvement in this area. This means that non-linearity does not give an advantage for high gaps.
Lower BC, up to around 10 BC, can see a big improvement. The resolution here is denser, and the amount of outliers is limited in range compared to the OF. A problem with the network is that it is not constant for low gaps and instead varies a bit, mostly between 15 and 25 BC. This is the same range where the OF was losing its performance.
Hits with even lower BC see a slight improvement. The OF could not handle them at all, while the CNN makes a symmetric distribution centred around 0. The range of the outliers is also more limited compared to the OF. Keep in mind that the evaluated data show a random gap distribution, that is, Gaussian distributed around 30 BC.

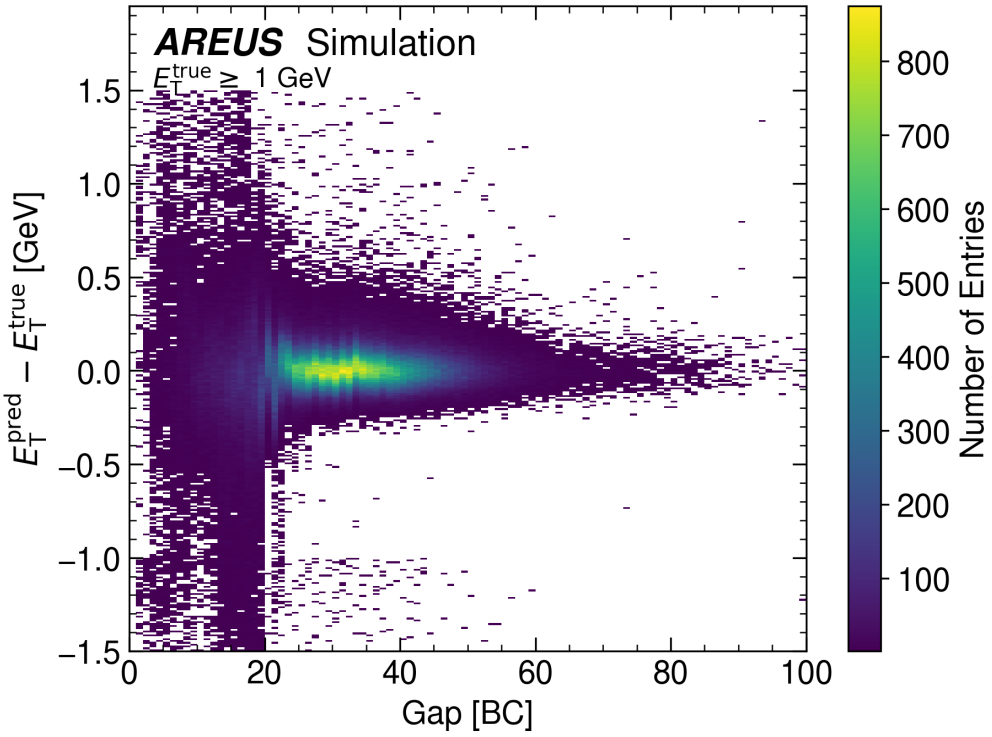

Figure 2.8: Differences of energies reconstructed with the 2CNN_99 and true energies plotted against the distance between two hits in BC.

2.3 Quantization of Convolutional Neural Networks

2.3.1 Quantization of 2CNN_99 to 18 Bits

For the next part, the not quantized neural network had to be retrained quantization aware. This was done by implementing the same architecture in QKeras. Implementing a network in QKeras means replacing all layers with quantized versions of the layers. These layers differ from standard layers by using a quantizer that does the quantization. During training, the weights are initialized as floating point numbers. The quantizer is a function that is applied at the end of each epoch on every weight to reduce the bit width. This work used the standard quantization with Eq. 1.12 as a quantizer. Some alternatives use stochastic rounding, but those were not tried in this work.
Also, the activation function was replaced with a quantized version, in this case, QReLU, to quantize the layer's output.
For the implementation, the bit range has to be chosen first. The input data, which are normalized ADC values, are 16-bit numbers (1 sign bit and 15 integer bits). Quantization of the input would be possible but would result in a loss of information on the measured energies. This would result in a worse resolution and was therefore not tested. If resources in the future are very limited, quantization of the data could reduce the consumption further.
For the activation function, 18 bits are used at the start. Later, it will be tested to quantize this.
Next are the weights and biases in the network. Since there is no difference between both in terms of quantization, the term weights includes biases in this work unless said otherwise. It is possible to quantize them differently, but every filter uses just 1 parameter as bias while using its whole kernel as parameters for the weights. For this reason, there would be no significant gain from quantizing them differently.
Keras handles the weights as 32-bit floating point numbers. The quantization will start with 18 bits for them because it is the minimum requirement for the FPGA. These 18 bits include 1 sign bit, 1 integer bit and 16 fractional bits. Using more bits for the integer would be possible, but the weights of not quantized networks did not use higher integer numbers. Assuming that the quantized network will use weights in the range of the not quantized network, no additional integer bits are needed. Also, the data themselves use no integer bits, as does the activation function. Multiplication with large integers would, therefore, result in outputs that are rounded back to 1. Giving integer bits to the activation function, while limited to 18 bits, would mean that the data are rounded, which might lead to some loss of information. If the quantized networks do not perform well, the distribution of bits can be changed.
One exception is the last ReLU activation of every network. Using no integer bit for this would mean that the network will round outputs bigger than 1 back to 1. These outputs do not correspond with possible ADC values but will occur during the training because of their stochastic nature. The loss function of the network will be calculated with the quantized

output of the network. This could lead to the network underestimating the impact of some parameters. In order to deal with this, the last activation function will be given an additional integer bit (16 bits + 1 integer bit). This bit will not be needed for the FPGA implementation. With the networks implemented in QKeras, the trainings can be done. The network was now trained 20 times, and the training with the lowest loss will be evaluated. Reducing the trainings from 100 to 20 can worsen the achievable performance. It was necessary, though, because the quantization increases the time for trainings and training of many networks was needed. Fig. 2.9 shows the performance of the best 2CNN_99 network, quantized to 18 bits. Comparing this with Fig. 2.6 shows no visible decline in performance. The resolution is the same, and the distribution and density of outliers is equal. Even the error structure at 20 GeV was reproduced. The structure was seen in many trainings of this architecture. Some trainings with a higher loss had a better distribution at this energy. This shows that the networks can learn to avoid this error. The architecture of the network may be prone to errors in this range. Doing more trainings than 20 would likely result in networks with a lower loss and without this structure because of the better statistics.

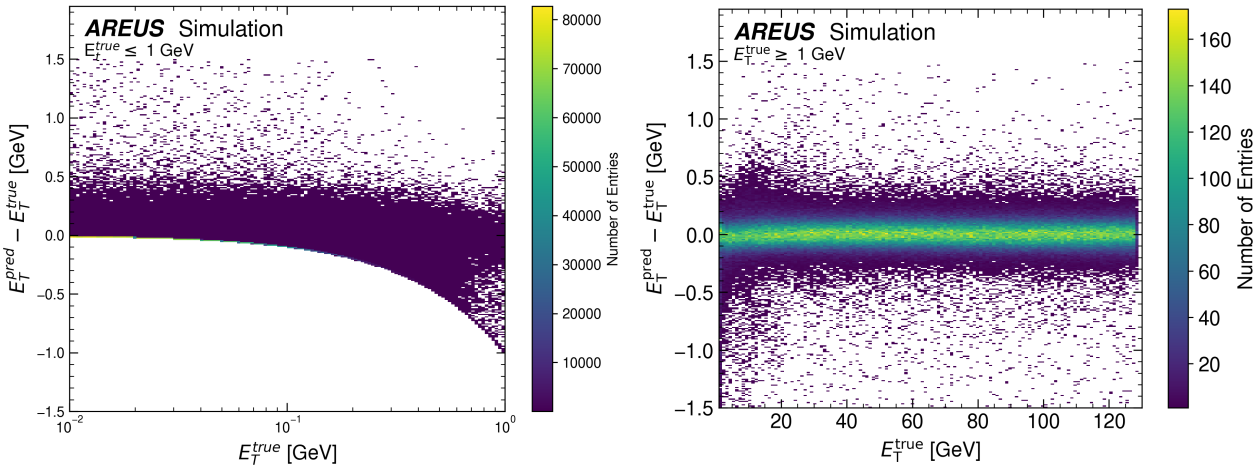

Figure 2.9: Differences of the energies predicted by the Optimal Filter with the true energies for a network quantized to 18 bits.

In Fig. 2.10 are standard deviations and 98% ranges shown for comparison. The standard deviation is again at a value of around 0.083 GeV for all energies. An increase in this value would have decreased the resolution, which would have resulted in a worse performance. Getting a similar value was important. The 98% range is also similar to the not quantized network. Its upper range is again at a mean value of 0.46 GeV, which is slightly better than using full bit width. The upper range for energies between 10 and 20 GeV decreases to 1.1 GeV, which is a small increase in performance. Looking at the lower range, this is again very similar, at a mean value of around -1.3 GeV for all energies.

Using fewer bits resulted in a similar performance to not quantized networks. This was the goal of quantization. Minor deviations between the neural networks, as seen here, are expected

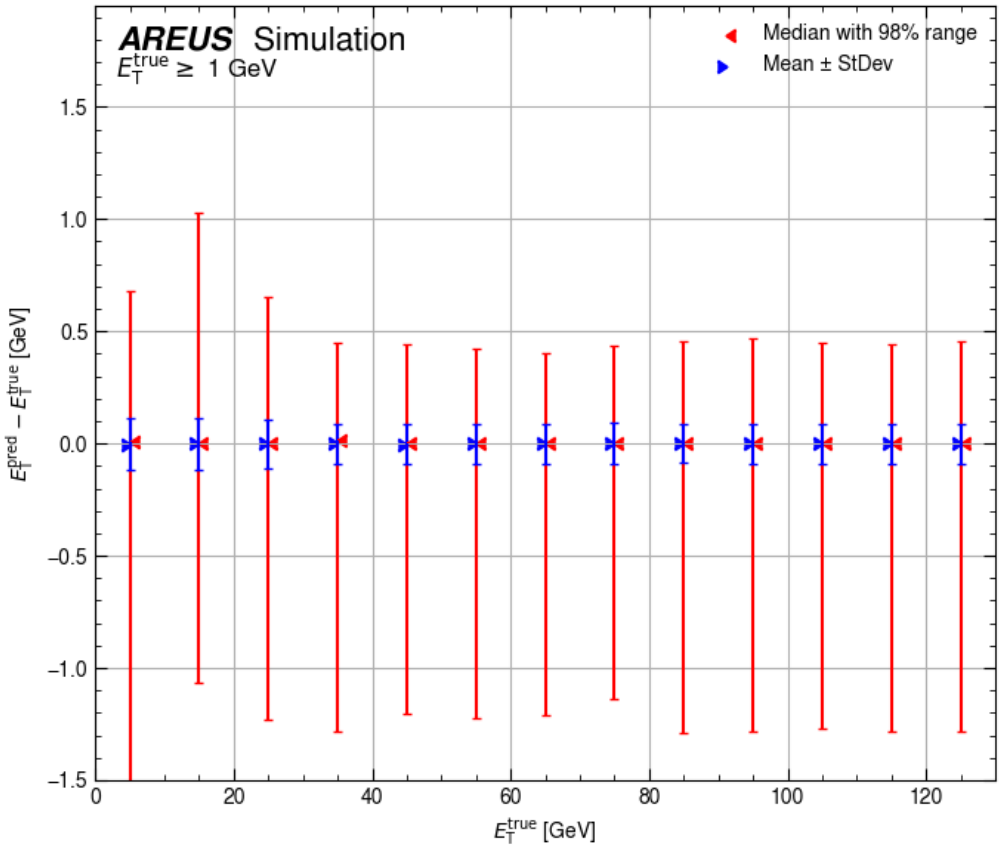

Figure 2.10: Mean value with standard deviation and median with 98% range of energies reconstructed with the 2CNN_99 quantized to 18 bits.

because of the stochastic training. Increasing the number of trainings would decrease these deviations, but the general trend can be seen regardless.

Fig. 2.11 shows the performance of the quantized network on different gap sizes. Here, the result of quantized trainings is also equal to the not quantized. The distribution is again very equal. For high gaps over 30 BC, both networks perform equally well. Its resolution for gaps between 20 and 30 BC BC is less smooth than the not quantized network but is more dense for lower BC. The number of overestimations, which means the outliers in the positive, were also reduced here.

This means quantization to 18 bits is possible without degradation of the performance. Implementation of the networks on the FPGA will be made easier with this.

As said before, quantization to even lower bits would save computational resources. For this reason, the 2CNN was trained quantization aware to bit widths under 18 bits in the next section.

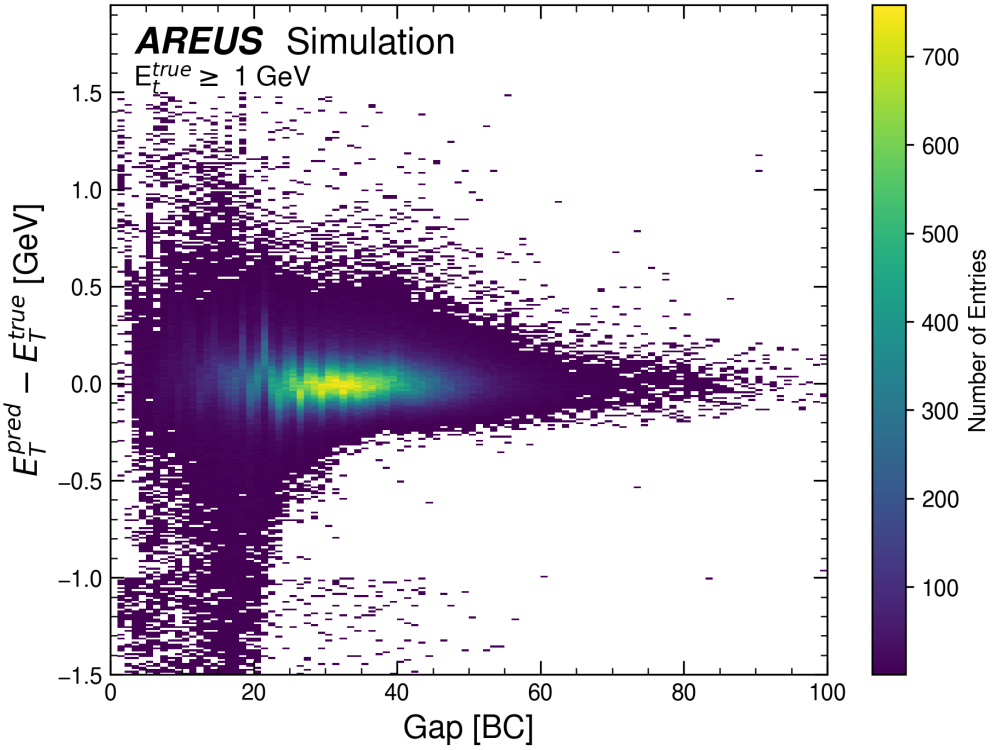

Figure 2.11: Differences of energies reconstructed with the 2CNN_99, quantized to 18 bits, and true energies plotted against the distance between two hits in BC.

2.3.2 Quantization of 2CNN_99 to low bit Widhts

In order to reduce the bit width further, the 2CNN_99 was trained again for different lower bit widths. This means the total bit width was stepwise reduced from 18 to 9 bits. Lower bits were not tested because a visible drop in performance started at 15 bits. At 10 bits, the trainings were getting unstable, meaning many trainings did not perform well. Every bit width was again trained 20 times, and the training with the lowest achieved loss was evaluated. This creates a slight bias against trainings with low bit widths. A worse performance with fewer bits means that the average performance drops. It is still possible, however, that a very low percentage of those trainings achieve a good performance. Those rare outlier networks are not useful in practice because the network is not final and will need to be retrained on different data sets and calorimeter cells. The reproducibility of the networks is evaluated in section 4.
In Fig. 2.12, the distribution of the energy residuals of 2CNN_99 quantized to different bit widths can be seen, together with the OF for comparison. This plot shows the general energy resolution of the algorithms.
In this plot, it can be seen that there is no large difference between using full 32 bits and quantizations down to 16-bit. These networks all have relatively equal height and width. They also outperform the OF. Starting with 15 bits, the height is sinking and drops under the OF. Its width, on the other side, is still relatively constant. This means that the network is losing its ability to deal with outliers the fewer bits it has available. The bias of the networks is relatively constant and always better than the OF. This seems to be independent of the quantization, except for very low bit widths under 12 bits, that are not shown in the plot.
In Fig. 2.13, the performance can be seen for networks with different bit ranges, measured again with mean value with standard deviation and median with 99% range.
The standard deviations of the networks are constant at 0.083 GeV until 11 bits. For lower values, this is increasing. Also, starting at 11 bits, the network gains a bias towards underestimating energies. This bias seems to grow linearly until nine bits, the lowest tested bit width.
The 98% range is relatively constant until 16 bits. At 16 bits, the range is also the lowest out of all networks. This could potentially mean that 16 bits are optimal for the network. A reason for this could be that the input consists of 16-bit numbers and that using a filter with the same bit width is beneficial. It could also be just some statistical fluctuation during the trainings. The differences are, however, small in total. For lower bits, it is starting to increase. At 11 bits, this increase is becoming very large.
On a first look, this means that it is possible to implement networks using 16 bits for the weights. This network would outperform the OF on subsequent hits and does not lose performance compared to the not quantized networks. Since the network uses 99 parameters in total, using 16 bits would result in 198 saved bits in total, or a reduction of 11% for the bits used on weights, compared to 18 bits. Using fewer bits than 16 is also possible, but this comes

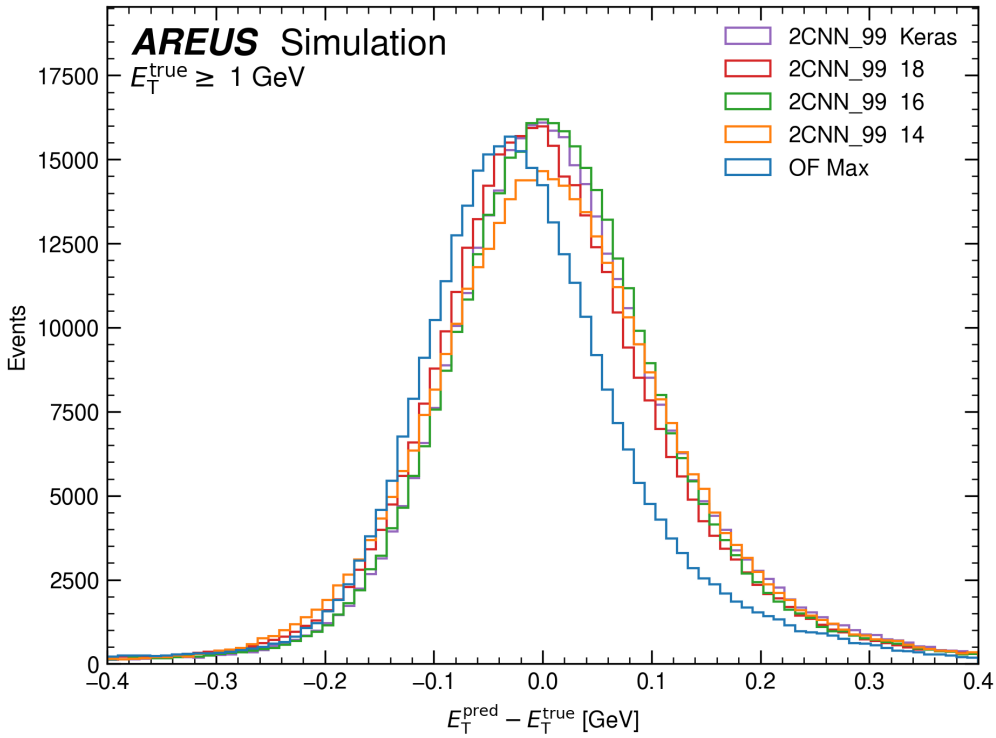

Figure 2.12: Performances of 2CNN_99 networks with quantization to different bit widths, compared with the OF. The number right to the network shows the used bit width. Keras stands for full 32 bits, as used in Keras.

with a trade-off in performance. Decreased performance means that the networks make more outlier errors. The minimal bit width that produced good results was 12 bits. Networks at this bit range were outperformed by the OF in the overall energy reconstruction. They still hold an advantage on overlapping hits, however.

Estimating the reduced ALM and power consumption would be desirable but was not possible at the time of this thesis. This needs a working implementation of quantized networks on the FPGA first, which was still in development. Earlier estimations saw a linear decrease, with every saved bit, on the ALM consumption.

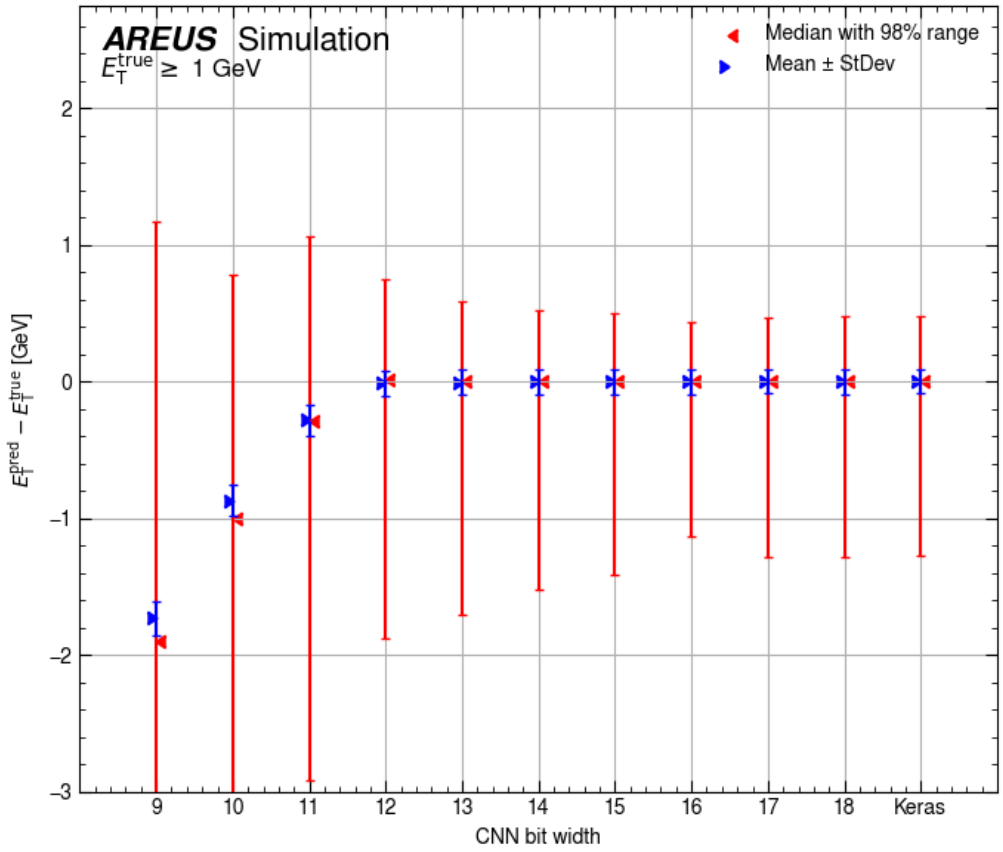

Figure 2.13: Mean value with standard deviation and median with 99% rang for the 2CNN_99 networks, trained quantization aware to different bit widths. Keras stands for 32 bits, as used in Keras.

2.3.3 Quantization of the ReLU Activation Function

The 2CNN_99 is using two ReLU Activations. One is for the output layer and uses 16 bits (+1 integer bit only for training). These 16 bits were chosen because the ADC values, which are supposed to be the output, are 16-bit numbers. This will not be quantized because quantizing the ADC numbers would result in less precise outputs. Using more bits here would lead to overly precise ADC numbers.
The first activation function was using full 18 bits for now. It is possible to quantize this. If the network is interpreted as a filter for the input, quantizing it to 16 bits would make sense. This would keep the data flow at constant 16 bits in the network and make every layer's output be interpretable as ADC numbers. Saving two bits on every input number, together with the large data rates at HL-LHC, would also save some resources.
To test this, the 2CNN_99 was trained with weights quantized to 14 bits because the performance should equal 18 bits and with a quantized first activation function. Tested were ranges from 14 to 18 bits. Lower bits were not tested because the performance started to drop at lower bits.
In Fig. 2.14 can the results be seen, measured as mean value with standard deviation and median with 98% range.
The right plot shows that the standard deviation is constant from 18 to 16 bits. This means that these bit ranges do not influence the resolution. It slowly increases for lower bits, which means that the resolution worsens. The 98% has its lowest value at 16 bits. For lower and higher bit ranges, it is starting to increase. This increase is worse for lower bits.
An explanation for this is that the network treats every input as an ADC value during the entire network. Using fewer or more bits for calculations would make it less or overly precise and result in errors.
This means that 16 bits is the optimal range for this activation function, and 2 bits can be saved here. Since the decrease in performance is not very big for 14 bits, it would also be possible to save more bits if needed, with an increasing amount of outliers as a trade-off.

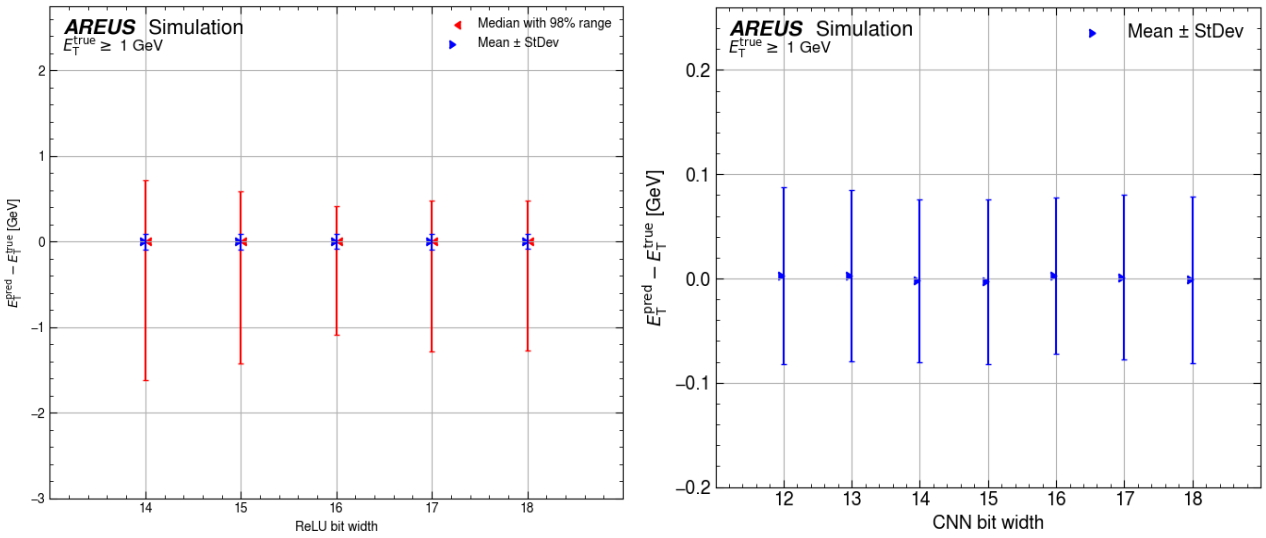

Figure 2.14: Performances of 2CNN_99 networks with quantization of the ReLU activation function to different bit widths, compared with the OF. This is shown on the left as mean value with standard deviation and median value with 98% range. Only the mean value with standard deviation is shown on the right side for better visibility.

3 Optimization of New Convolutional Neural Networks

As seen in the last chapter, quantization can be used to reduce the resource consumption of neural networks on FPGA. Together with optimizations on the framework that were done in parallel to this thesis, this motivates to look into larger neural networks with more parameters. In Fig. 3.1 can, a new estimation for the consumption of one network with 100 parameters be seen based on optimized VHDL implementation. Even without quantization, the ALM consumption was reduced by 68% through optimized VHDL implementation. With the new estimations for consumption, the new limit will be 500 parameters. At this point, the consumption of DSP and power would become a new limit. ALM and power usage can be reduced with quantization under 18 bits, but this does not influence the DSPs. Together with the fact that there is a need for other software on the FPGA, the full 500 parameters will not be used for the networks.

Instead, 400 parameters were chosen as a new limit. This should satisfy all resource demands without quantization under 18 bits while still quadrupling the available parameters. For comparison, a 2CNN with up to 250 parameters was also optimized. This enables a closer look into the eventual performance increase in dependence on the number of parameters.

For the basic architecture, CNNs and TCNNs were used again. Other architectures would require changes on the FPGA software and would have new resource demands. One problem of the TCNN networks was the very limited number of parameters that had to be used for two different networks. It was expected that the new TCNNs would significantly improve their performance.

Apart from the number of parameters, it is also a possibility to look into the number of layers. The software currently supports only up to four layers. Using more layers might be beneficial because a network with more layers can learn highly non-linear functions more easily. This thesis will not test more layers because two and 3-layered networks have been shown to have the best results. Four layered CNNs were tested but had no good performance. In case 4-layered networks would show a good performance over others with more parameters, this would motivate to look into deeper networks. Also, the architecture optimization used works well on small networks, but the time needed increases drastically with more layers.

FPGA	Network	Multiplexing	Detector cells	$f_{\max}$	ALMs	DSPs
Stratix-10	2-Conv CNN	12	396	415 MHz	8 %	28 %
	4-Conv CNN	12	396	481 MHz	18 %	27 %
Agilex	2-Conv CNN	12	396	539 MHz	4 %	13 %
	4-Conv CNN	12	396	549 MHz	9 %	12 %

Figure 3.1: Estimated Consumption of 2CNN and 4CNN networks on Stratix-10 and AGilex FPGA with optimized implementation.

Optimizing the architecture of a neural network can be done in different ways. Many neural networks use so-called grid search for optimization. For a grid search, the network is given lists or matrices of possible parameters (called grid). The network will then try every parameter or every combination from the grid, train the network with them, and give the result. This approach has the problem of taking a long time to compute and being biased in the choice of parameters in the grid. If the optimal parameters are not included, it will not give the best result. One advantage is that its computation can be easily parallelized and, therefore, done efficiently.

It has been shown that a random search that tests random combinations of parameters usually gives better results because it is not limited in its choice of parameters. [48] Such random algorithms have the backside of taking more time.

For this work, used was an optimized random search called Hyperband that discards the worst possible hyperparameters after a given amount of iterations [49]. This algorithm was implemented as KerasTuner [46]. It takes the general structure of the network (layers and maximal amount of parameters) and optimizes its architecture. Every set of parameters was trained three times to reduce the error from the stochastic nature of the training. Since the optimization is a stochastic process, the networks might not have the best architecture possible.

At the time of this thesis, KerasTuner was only compatible with standard Keras Layers. QKeras has a function called AutoQKeras that can change the architecture of not quantized networks into optimized quantized networks. AutoQKeras is currently limited in its functionality and does not support a limit in parameters or the field of view. For this reason, AutoQKeras wasn't used.

It was not possible to optimize quantized neural networks efficiently. Instead, not quantized networks will be optimized and retrained afterward with quantized layers. One problem is that the optimized architecture is not necessarily optimal for quantized layers. The 2CNN_99 has shown that quantization does not necessarily decrease the performance. Because of this, it was assumed that architectures optimized without quantization would perform well when quantized. The performance will, however, most likely drop at low bit ranges. Using optimization algorithms that support quantized layers would be preferable in the future. Such an algorithm could find architectures optimized for quantized networks and might enable lower bit ranges.

For the optimization, every architecture was optimized independently, and the best three results were evaluated by training them quantization aware at 18 bits. This was done separately for 2CNN, 3CNN, 4CNN, 3TCNN, and 4TCNN architectures. The trainings were again done with 20 iterations each and the networks with the lowest loss were evaluated. Each training went again for 100 epochs with early stopping. The following will show the general performance of the best networks.

Alternatively, it would have been possible to let KerasTuner optimize all architectures si-

multaneously. In this case, KerasTuner would have searched for the best network out of all possible architectures, eliminating the need to evaluate and compare the networks afterward. This was not done for multiple reasons. Most important is that this enabled a more detailed comparison of the performances. Networks may perform differently on specific measurements. One architecture might have a higher loss and perform worse, according to KerasTuner, while having a very good performance on subsequent hits. In order to spot this, if it happens, all architectures were evaluated.

First, as a motivation and to check if the performance actually increases, a look into 2CNN architectures with up to 400 parameters will be taken. For this, two networks were optimized. One with full 400 parameters and one with 250 for comparison and to see the gain in performance.

In Fig. 3.2 are the distributions of the reconstructed energies for the best networks from the two architectures seen. It is visible that both networks have a better performance than the 2CNN_99. Better means for both networks that the height of the distribution got visibly increased. The width on the other side hasn't changed much. This means that by using more parameters, the network learns to work with more outliers then before. Also, the increase for the 235 parameter network is not large compared to the 99 parameter one. The 391 parameter network, on the other side, has a much bigger increase.

This shows that an increased performance is possible and that more parameters benefit the networks.

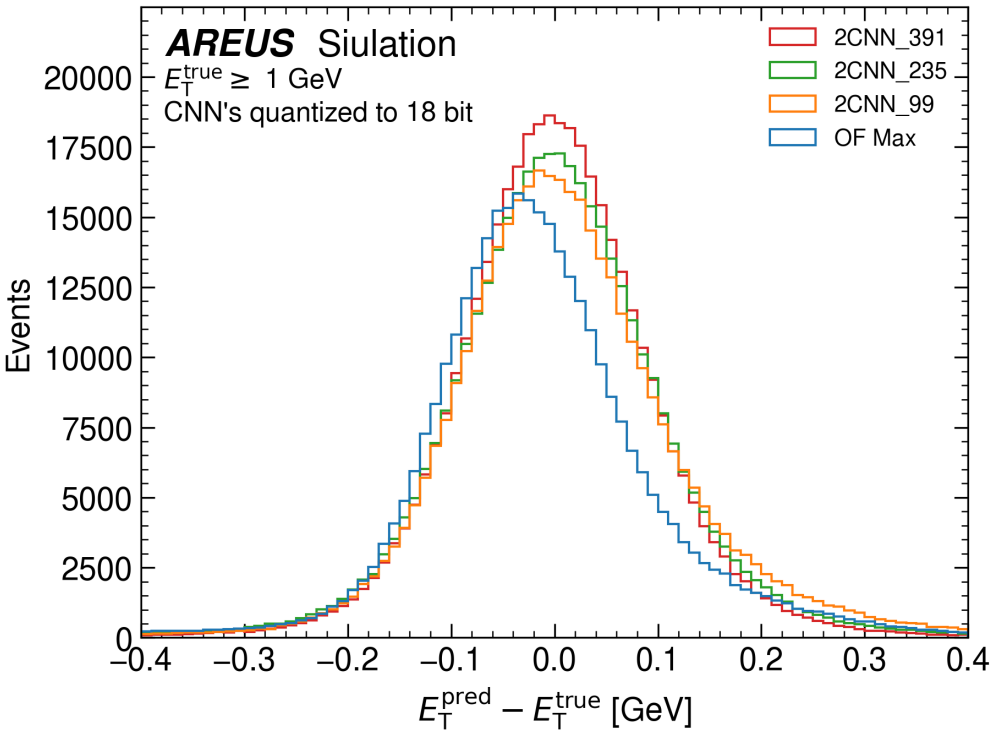

Figure 3.2: Comparision of 2CNN structure with different parameters.

In Fig. 3.3 can an overview of the performances of different optimized and trained networks be seen. The left plot shows the standard deviation of the networks with its mean value, together with the median and 98% range, while the left only shows the standard deviation with mean, for a more detailed view. Mean and standard deviation were again calculated with a Gaussian fit.

TCNN networks have become better than the 2CNN_99 but are still worse compared to the new CNNs. The 4TCNN has the worst result of the new architectures. Since 3TCNNs perform better than 4TCNNs, this thesis will concentrat on them when talking about TCNNs.

The bias of all networks is similar for all networks and much smaller compared to the OF Standard deviation of the networks is also relatively stable. Any increase in this regard from the higher number of parameters seems small.

The 4CNN network performs worse than 3CNN and 2CNN, without any performance gain. For this reason, no further look into networks with more layers will be done in this thesis. Four layers stay as the limit.

It can also be seen that the new architectures all have a much better 98% range. This, together with the constant standard deviation, means, as guessed before, that increased parameters enable the network to deal with more outliers. It can learn to deal with more special cases.

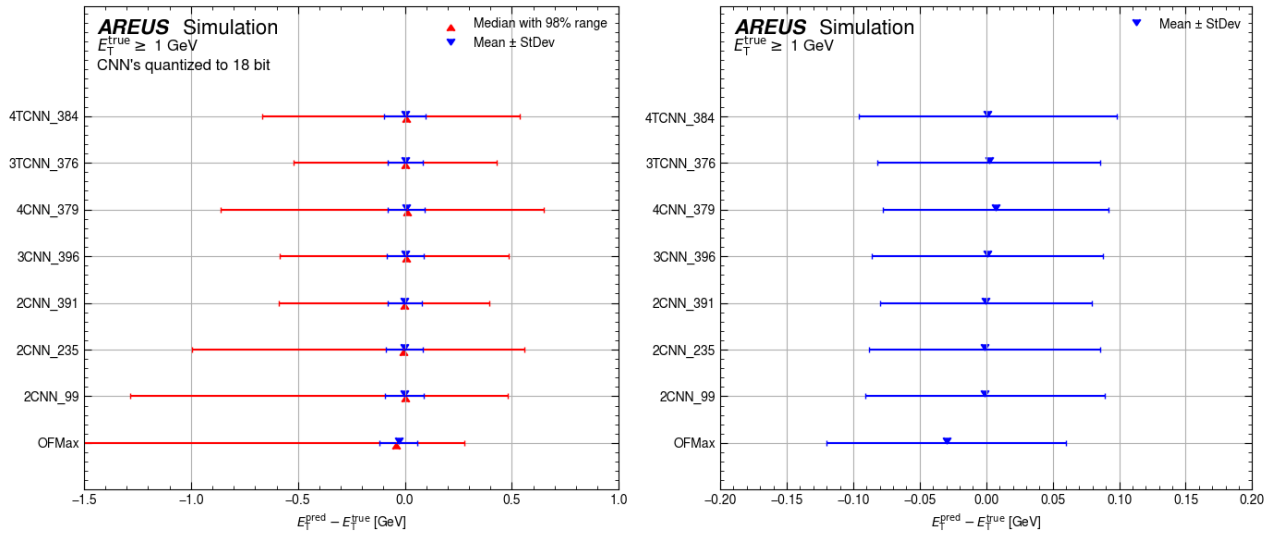

Figure 3.3: Performance on Neural Networks with up to 400 parameters. All the shown networks were optimized with KerasTuner and trained quantization aware to 18-bit.

The best-performing networks will be evaluated in more detail in the following. The 4CNN and 4TCNN networks are not included because they performed worse and had no significant advantage over the others. Plots on their performance can be seen in A2 and ref4tcnn.

3.1 Optimized 2CNN with 400 Parameters

In this section, the performance of the 2CNN network with increased parameters will be evaluated. The best network architecture can be seen in Fig. 3.4. This network uses 20 filters in the first layer, with a kernel of 5, and 1 filter in the second layer, with a kernel of 20. Dilation was not used for this network. It was still available in KerasTuner but was not used in any good-performing network. The reason for this is, most likely, that the full field of view can be used more easily by larger networks. The artificial increase with dilation, which can lead to problems during training, was therefore no longer an advantage.

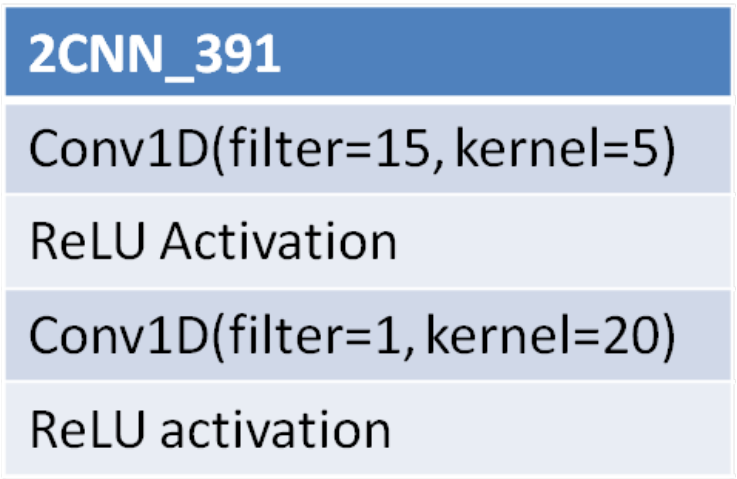

Figure 3.4: Architecture of the 2CNN_391.

As an observation during the architecture optimization, the 2CNN networks followed a trend of preferring many small filters in the first layer and one big filter in the second. It is possible that this general structure is optimal for the 2CNN. A possible interpretation would be that filters in the first layer make detailed stepwise operations along the input sequence to find specific structures, like maxima or undershoots. The last layer then combines this gathered information on a large part of the sequence to calculate the energies.
This 2CNN architecture was retrained quantization aware. The weights in the network were quantized to 18 bits, with 1 sign bit and 1 integer bit, and the activation functions were quantized to 16 bits with no integer bit. An additional integer bit was used in the last layer to stabilize the training.
Fig. 3.5 shows the distribution of the reconstructed energies. On the left again for energies under 1 GeV and on the right for energies above 1 GeV. A comparison with the 2CNN_99 (see Fig. 2.9) can be made to evaluate the increased performance.
Looking at the high energies first, a large increase in performance can be seen. The central resolution has increased its density, and the amount and density of outliers have decreased. Interestingly, the number of outliers is now more asymmetrical, even though both sides have a decreased width. Overestimations are now more common than underestimations, as seen in

the density. This means the network has learned to better deal with undershoots, as expected from close hits. The network may have preferred to use its increased capacity to learn those. The width of the distribution on both sides is now also better compared to the OF. It is still constant for all energies. On energies under 30 GeV, an increase in the number of outliers can be seen. However, the error structure from many former networks at this energy has vanished. The performance on low energies on the left side has not increased. Reason for this should be that the network is focusing more on higher energies because these have a bigger influence on the loss (energies above 1 GeV are weighted with a factor of 30). Based on the loss, this performance seems good enough for the network, and using more of its capacity for low energies seems not worth it compared to focusing on high energies. Changing the weighting could increase this performance for low energies, but this would decrease the performance of the more important high energies on the other side.

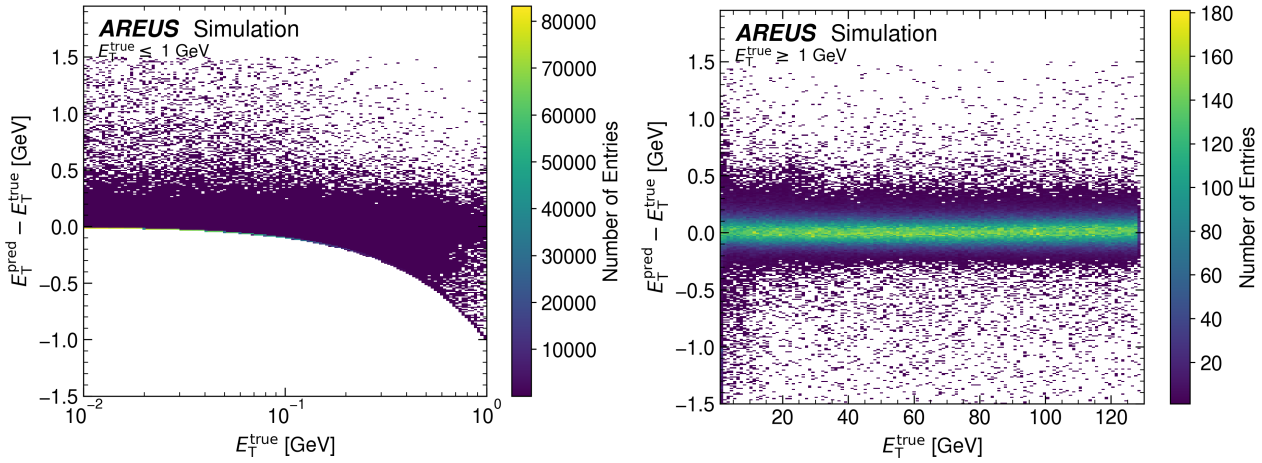

Figure 3.5: Difference of the energies predicted by the 2CNN_391 with the true energies.

In Fig. 3.18 was the performance measured again with mean value with standard deviation and median with 99% range. The range of the network, which measures the outliers, has decreased a lot. Its upper range has a mean value of around 0.42 GeV, and its lower range is around 0.38 GeV. This is a large performance increase, particularly in the lower range. The range is now also more stable for the different energies. For energies over 40 GeV, this value varies only slightly. Lower energies see an increase, up to 10 GeV. Under 10 GeV, the upper range decreases again, but the lower range increases a lot. This comes again from increased overestimated and overlapping small energy hits.

Its standard deviation, on the other hand, is around 0.083 GeV. With this value, the network has reached the OF performance. Increasing the resolution further might be more difficult, and this value is already slightly better than the OF performance. It is also possible that a Gaussian standard deviation of 0.083 GeV is the best achievable performance. The resolution is limited because of the technical resolution of the ATLAS LAr calorimeter. Also, the OF was designed to have the best possible resolution. Further improvements might not be possible

by using reasonable resources of the network. This shows that the network uses its increased performance to work better with outliers.

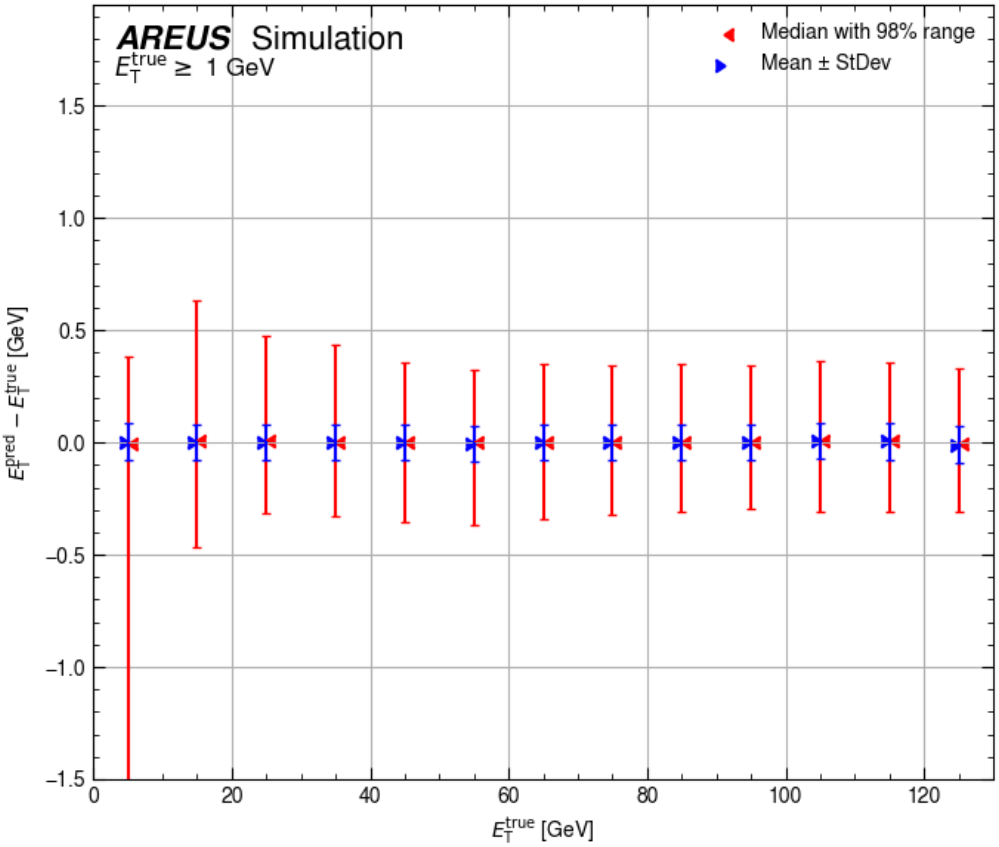

Figure 3.6: Mean value with standard deviation and median with 98% range of energies reconstructed with the 2CNN_391.

Fig. 3.11 shows the network's performance on different gaps between the hits. Here, a big improvement can be seen, compared to the 2CNN_99. The distribution got much smoother, with a visibly reduced number of outliers. Its total resolution, on the other side, has stayed the same. This shows that increasing the network's capacity mostly leads to better handling of outliers and smoother distributions.
Gaps under 40 BC improved a lot, and even under 20 BC, a much-increased density of the distribution can be seen. It is also more symmetrical, which means that fewer underestimations are made. This shows that the network can deal better with overlapping hits.
For gaps over 40 BC, the performance is similar to the OF. Those hits correspond to isolated hits. The OF was optimized to work on those hits. It is possible that non-linearity does not offer any advantage here. In this case, approaching the OF is the best the network can achieve. Gaps under 40 BC, on the other side, see a great improvement. The distribution becomes denser, and the resolution is now more constant. Undershoots are still the most common source of errors in this range but are much reduced in this network.
In order to get a better look at close hits, Fig. 3.8 shows the distribution of the reconstructed energies for gaps between 0 and 20 BC. Here, the increased performance of the neural can be seen. The distribution of the OF visibly decreases starting with 20 BC. Decreasing means that it first gets a bias towards underestimating the energies, and it broadens a lot. The 2CNN_391 is relatively stable down to 7 BC. Stable means the distribution stays centered around 0 and has a relatively constant width. The distribution also broadens at lower bits but stays centered at 0, compared to the OF.
The 2CNN architecture has benefited a lot from the increased parameters. It used the increased capacity to learn to deal better with outliers, which decreased the 98% range and to work better with closer hits, which increased the performance gaps under 40 BC.
On large hits, however, there was no big gain in performance. The resolution of the network was also not increased. It will be compared with other good-performing networks with up to 400 parameters in the following.

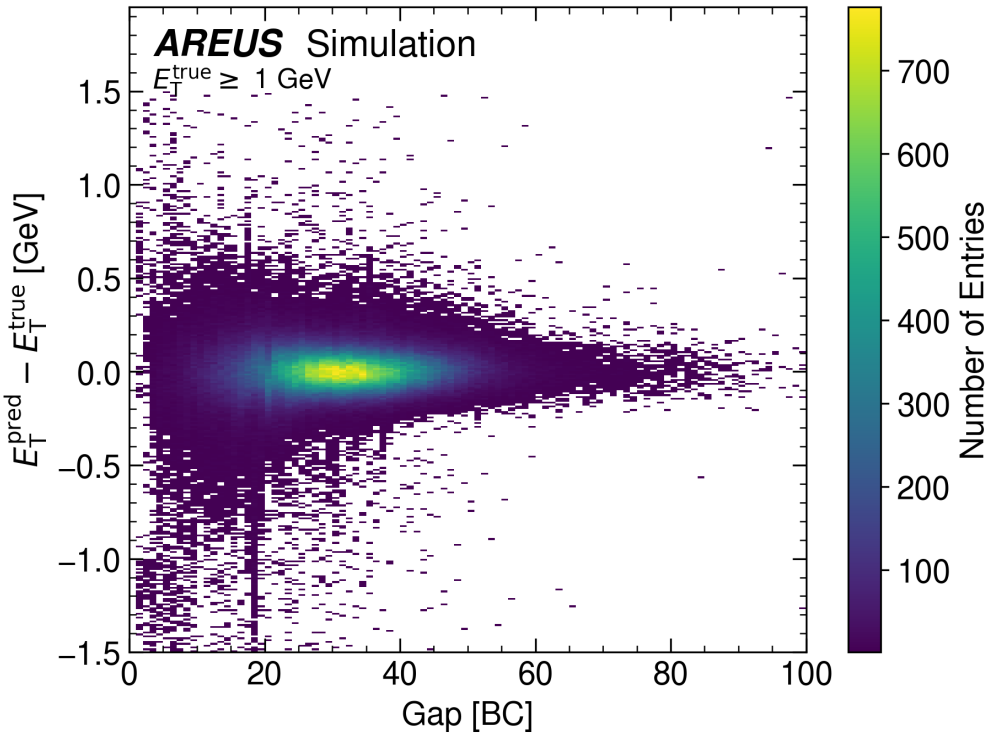

Figure 3.7: Differences of predicted and true energies plotted against the distance between two hits in BC. On the left is the 2CNN_391 network quantized to 18 bits seen and on the right the OF.

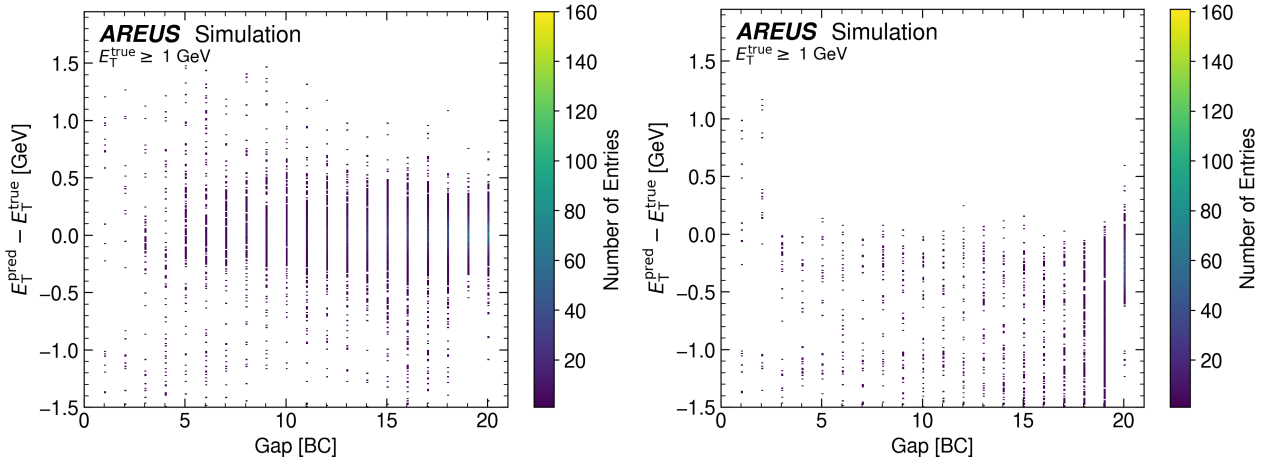

Figure 3.8: Energy difference plotted against the distance between two hits in BC. On the left shown is the 2CNN_391 and on the right the OF.

3.2 Quantization of 2CNN to low bit Widhts

As the best-performing network, the 2CNN_391 was quantized further to lower bit widths. This was done again by training the network 20 times quantization aware for every bit width. The total bit width of 16 bits for the weights was again quantized by reducing the fractional bits. The activation functions stayed quantized to 16 bits. Quantizing it further decreased the performance and would make the ADC values less precise.

Fig. 3.9 shows the performance of the network quantized to different bit widths. Here, the mean value with standard deviation and median with 98% range can be seen again. The standard deviation and 98% range are constant until 13 bits, with slight deviations of the 98% range. For lower bit widths, the range is increasing, while the standard deviation stays constant until 10 bits. Starting with 14 bits, the network is gaining a bias towards over or underestimating energies. This bias can create some error, but until 11 bits, it is in the order of the bias from the OF. The bias can potentially be a product of the relatively low number of trainings. If quantization makes the trainings less table, then more trainings might be needed for the best performance. Apart from this bias, the performance is overall stable until 13 bits. For lower bits, the 98% range is increasing, and starting with 11 bits, the bias and standard deviation also increase. Compared to the OF, networks over 11 bits have a better standard deviation and much smaller total 98_ range.

In Fig. 3.10 shown is the distribution of the reconstructed energies for 2CNN_391 networks with different bit widths and the OF. It can be seen that the network with 14 bits has some bias, compared to the 18-bit network. Height and width are not changed otherwise. This bias is also smaller compared to the OF. At 12 bits, the performance of the network has decreased a lot. It drops under the OF and has a much-increased number of overestimations. This changes the former Gaussian distribution into some landau distribution, with a tail for high energy differences.

It has not been checked yet, but the performance on gaps is important. This is not directly measured with the normal energy reconstruction. This might have changed "in secret" with a lower bit width while the overall performance is constant. Fig. 3.11 shows this performance on subsequent hits of the 2CNN_391 network trained quantization aware at 14 bits. Comparing this with Fig. 3.11 shows the difference to 18 bits. The distribution has stayed the same for the most part. Its resolution and number of outliers haven't changed in total. The biggest difference is that the resolution for gaps between 30 and 40 BC became less smoother. This is the region where the OF started to lose performance, too. The decrease is not large, though, and still much improved compared to the OF or 2CNN_99. For smaller gaps, the performance is similar to 18 bits.

In summary, the 2CNN_99 has a stable performance until 16 bits. The new 2CNN_391 is stable down to 15 bits, with a less significant drop in performance for lower bits. The new network seems to handle reduced bits more easily because of the increased capability with

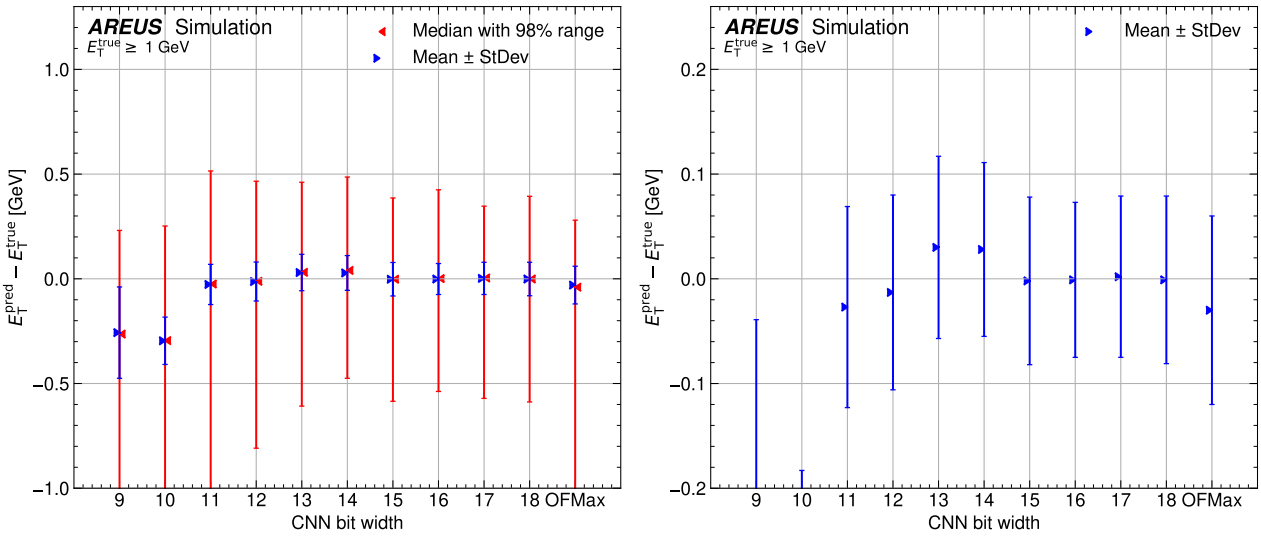

Figure 3.9: Performance of the 2CNN_391 quantized to different bits. The mean value with standard deviation is on the left side, calculated via a Gaussian fit, and the median with 98% range. On the right is only the standard deviation for better visibility. The bit number shown is the total bit width of the weights in the network. Activation functions and data were always quantized to 16-bit.

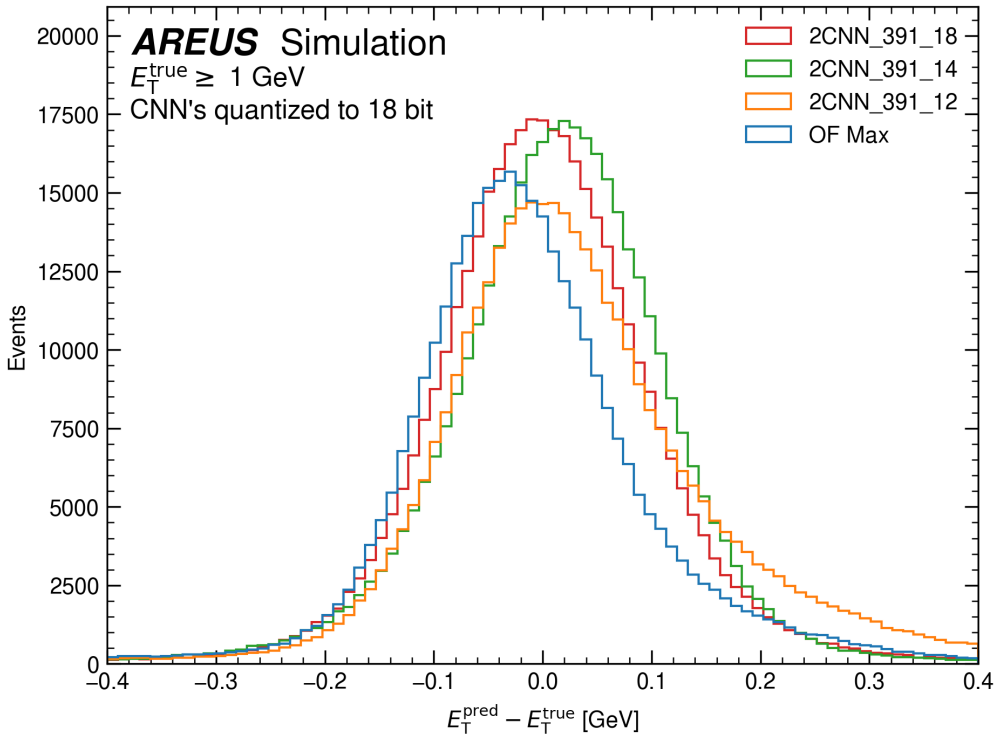

Figure 3.10: Distribution of the residuals for energies reconstructed with 2CNN_391 network trained quantization ware to different bit widths. The number right to the network shows the used bit width.

more parameters. This enables a safe implementation of networks with 15 bits and reduced resource consumption. If some bias of the networks is also acceptable, going down to 13 bits

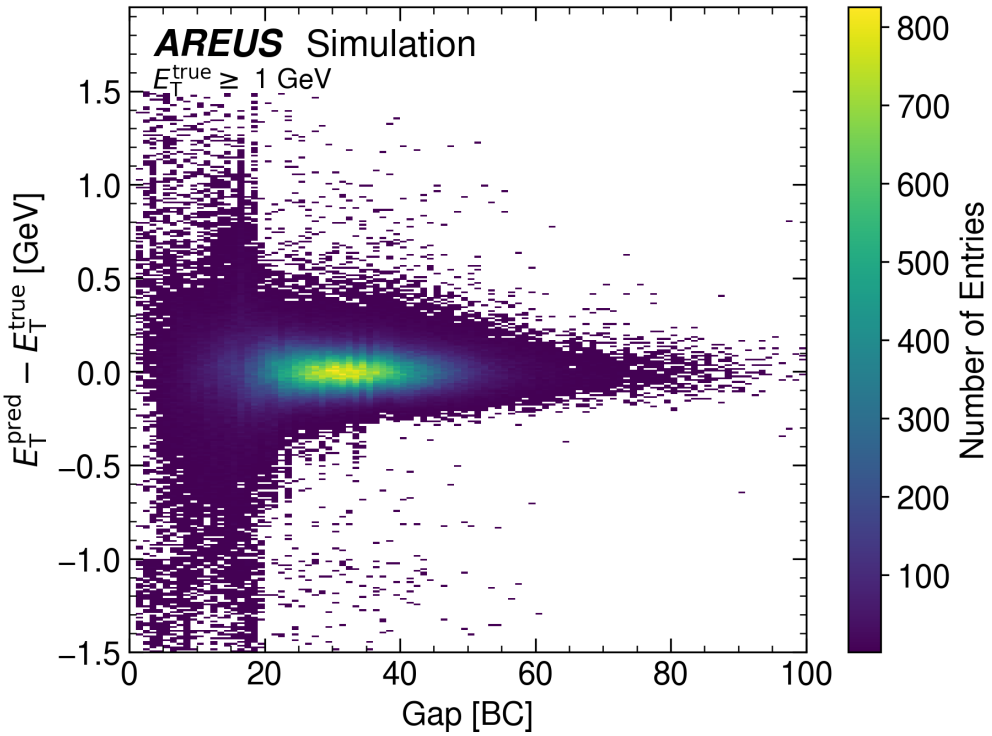

Figure 3.11: Energy differences of predicted energies by the 2CNN_391 quantized to 14 bits and true energies, plotted against the distance between two hits in BC.

would be possible. Assuming quantization to 13 bits, this is a decrease of around 28% bits needed for the weights, compared to a possible reduction of 11% for the 2CNN_99. The total consumption of the bigger network will be larger, but the relative possible decrease is higher for it.

3.3 Optimized 3CNN with 400 Parameters

This part will evaluate a three-layered CNN. Using more layers offers the advantage of easier approximation of non-linear functions for the network. This might be an advantage when reconstructing close hits. It comes with the disadvantage of taking more time for optimization because more layers enable more combinations of parameters.
The 2CNN networks followed a general trend for its architecture. For the 3CNN, this was more complicated. One trend that could be seen, and which the best network followed somewhat, was to use successively fewer and smaller filters in each layer. This layered architecture is commonly used for CNNs, even without hyperparameter optimization. It originates from biological neurons and the way human eyes perceive and analyze what they see. Starting with the whole picture and going to successive smaller details. Interestingly, the stochastic hyperparameter search also goes with this design for ADC sequences. Not all 3CNN networks followed this design, though.
This section will now evaluate the best 3CNN; its architecture can be seen in Fig. 3.12. The network had the lowest loss out of all trained networks and the best performance, according to KerasTuner. This could mean that the network has the overall best performance. It could also mean that the network overfits more easily compared to others. To check this, the network is evaluated together with the other architectures.

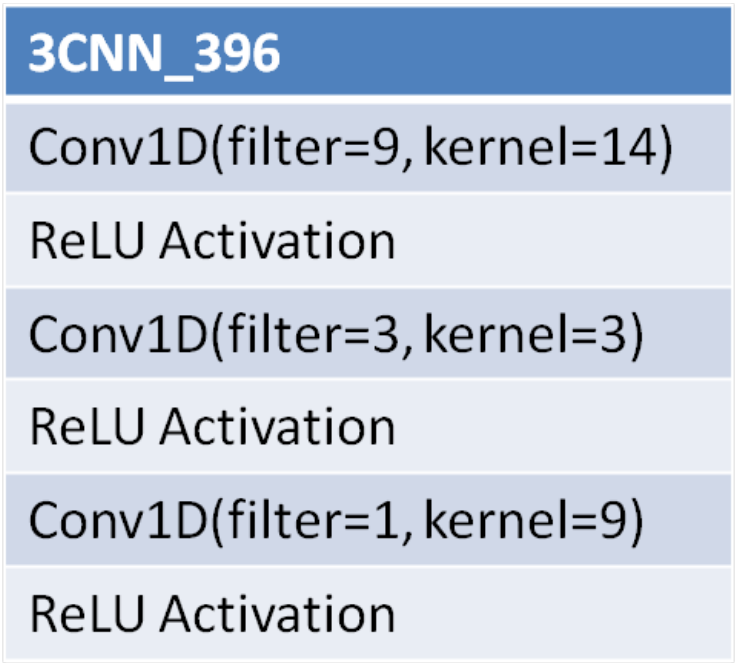

Figure 3.12: Architecture of the 3CNN_396.

The performance of the network on energy reconstruction can be seen in Fig. 3.13. On lower energies, this network performs as well as other networks. For higher energies, the network has a low amount of outliers compared to other networks. This is also constant for most energies, except at around 15 GeV. Here is an increased amount of overestimations, which results in an error structure similar to the quantized 2CNN_99. The standard deviation is, however, not worse at this energy, and the lower 98% is relatively small there. Overlaps of pulses at this energy may create this error. This error vanished mostly at the 2CNN_391, which shows that networks can learn to work with this.

Fig. 3.14 shows a more detailed performance, with calculated mean and median values. The standard deviation of this network is at a value of around 0.089 GeV for all energies. This is a slight increase compared to the other networks tested or the OF, which all had a value of 0.083 GeV. With this increase, the resolution of the network will be worse than the resolution of other networks.

The 98_ range is also increased, at an upper value of around 0.48 GeV for the upper range and 0.44 GeV for the lower range. These values also vary slightly with the energy. Energies under 10 GeV have a much increased lower range again.

Overall, this shows that the network does not have the best performance in energy reconstruction. Together with the best loss values, this shows overfitting.

The performance on close gaps can be seen in Fig. 3.15. Here, the network performs well compared to the OF. It has a dense distribution centered around 0, which is stable until 20 BC. For lower BCs, the number of outliers increases, and the network tends to underestimate close hits. The resolution here is also not smooth compared to the 2CNN_391 but improved compared to the 2CNN_99 and OF.

In summary, the 3CNN architecture performs well and benefits from the increased number of parameters. The network outperforms the 2CNN_99. It is, however, outclassed by the 2CNN_391 network. Especially on close hits, where it could be assumed that more layers

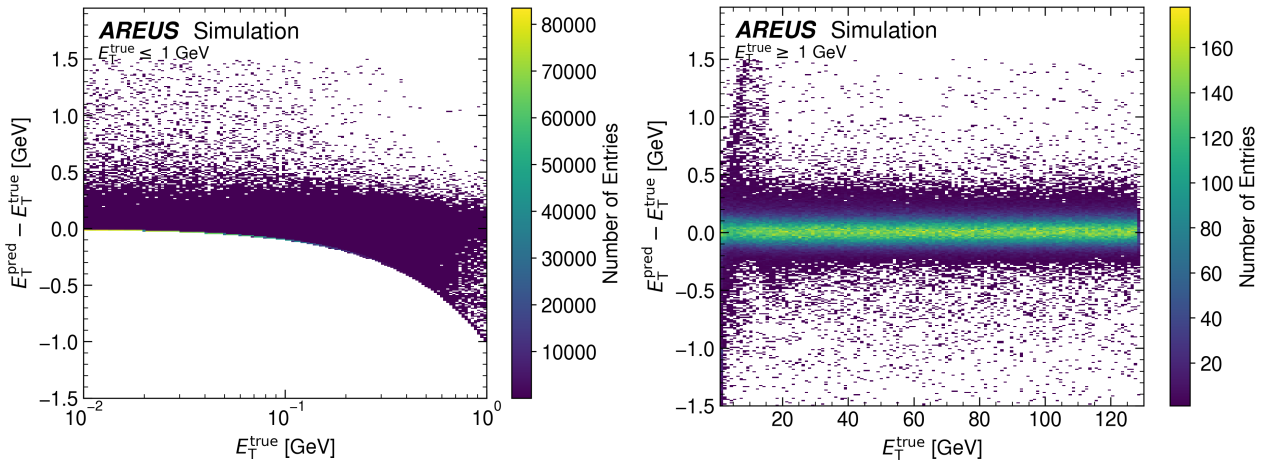

Figure 3.13: Difference of the energies predicted by the 3CNN_396 with the true energies.

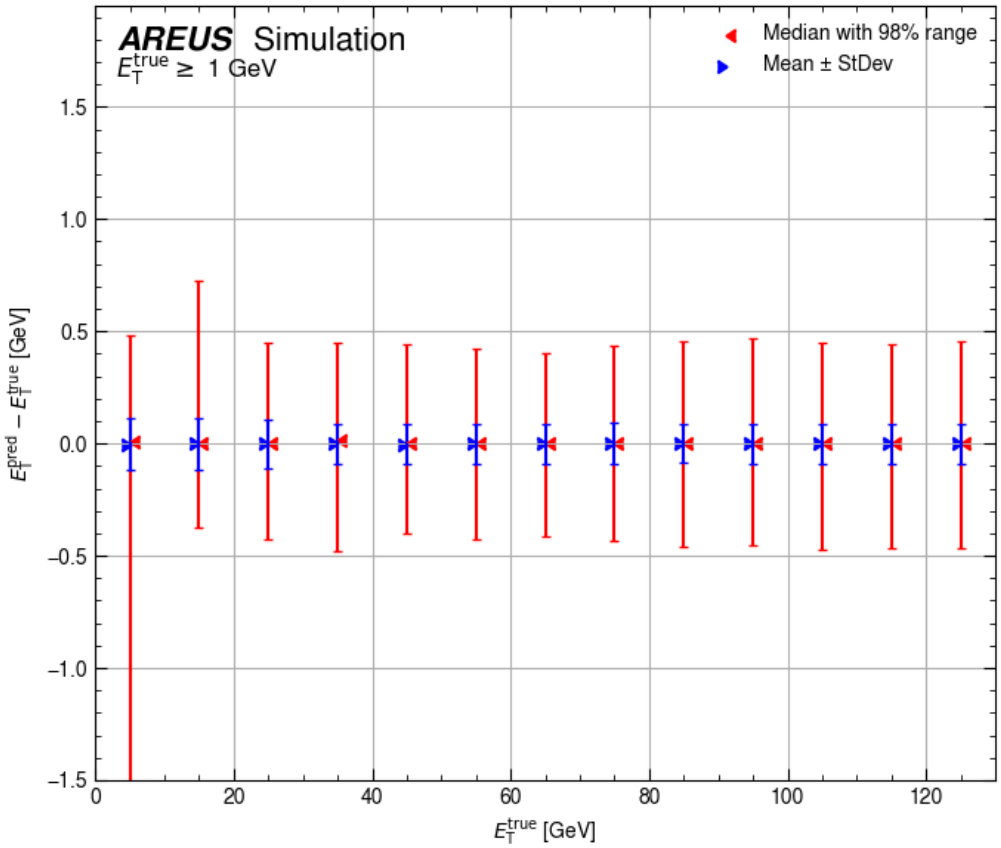

Figure 3.14: Mean value with standard deviation and median with 98% range of energies reconstructed with the 3CNN_396.

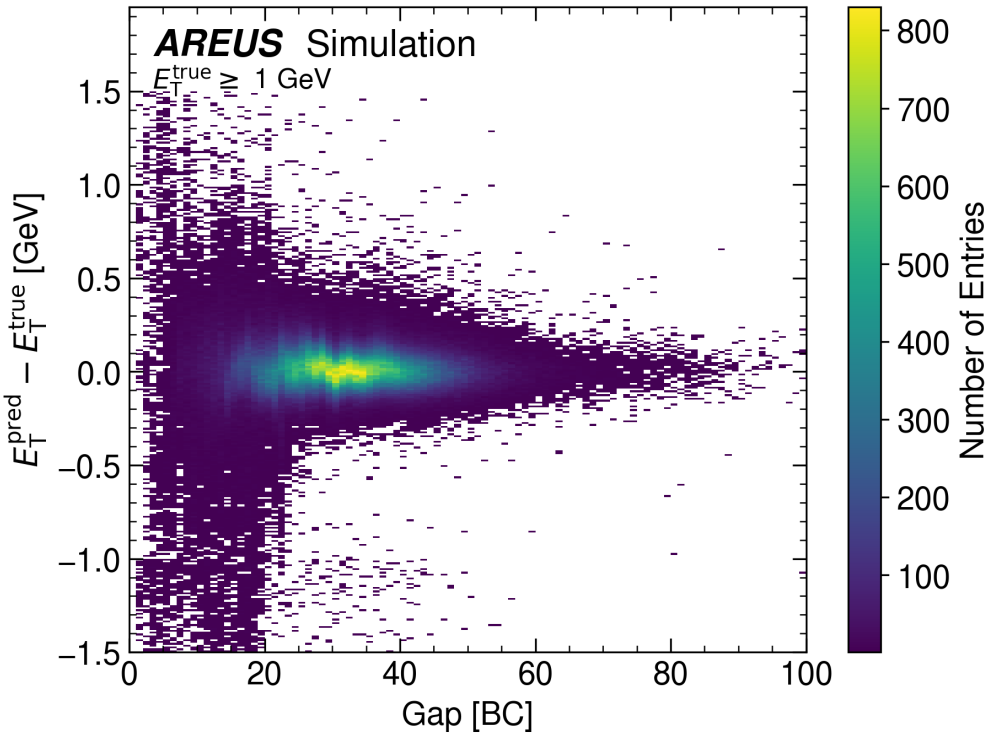

Figure 3.15: Energy difference plotted against the distance between two hits in bunch crossings for the 3CNN_396.

benefit the performance, the network is outclassed by the 2CNN architecture. Quantizations under 18 bits of this network were not done for this reason. It was also assumed that the 3CNN architecture behaves similarly to the 2CNN architecture with quantization. However, it will later be compared with the other networks on different data sets to see if this architecture performs better on them.

3.4 Optimized 3TCNN with 400 Parameters

The last of the networks is the 3TCNN_376. As a reminder, the TCNN networks were supposed to learn the position of hits in the input sequence and use this information for the energy reconstruction. The idea was that a network with this information could work better on close hits. TCNNs with 100 parameters did not work well compared to the OF or 2CNNs. The reason for this is, most likely, that the limited parameters had to be shared between two different networks. While the tagging network itself worked well, the energy reconstruction had problems.

Now, with more available parameters, it might be possible that the tagging will give these networks an advantage.

The network uses two layers for energy reconstruction and one for tagging. A general trend for the architecture was not observed during optimization. It is possible that the tagging enables different combinations that work equally well. One observation is, though, that most working 3TCNNs did not use the full field of view for energy reconstruction. An explanation for this could be that the tagging makes full visibility of the pulse unnecessary. Knowing the position of hits might be enough, and the CNN networks need a full kernel as an alternative to tagging.

The best 3TCNN network has an architecture that can be seen in Fig. 3.16. Its first layer with the Sigmoid activation is the tagging part. The concatenate layers combines the output of tagging with the ADC sequence and brings them together into the layers for energy reconstruction. It is using 376 parameters in total. The tagging used one filter with a large kernel. This was a trend for all working 3TCNN networks. It can be interpreted that having a large field of view is important to tag hits. The energy reconstruction part is different from the 2CNN architectures. It still uses many smaller filters in the first layer but no big filter in the second. Also the full field of view is not used for energy reconstruction. Tagging might make this redundant. Knowing the position of hits seems to be enough, and the network can instead use more smaller filters.

During training, the tagging was pretrained in order to let the network learn the position of hits. For this, the binary cross entropy (BCE) was used as loss. After pretraining, the network has learned to detect hits.

The next step was training the whole network on energy reconstruction, using the MAE as loss. During this training, the tagging part could still be adjusted, just like the energy reconstruction part. The idea behind this was that knowing the position of hits might not be ideal for energy reconstruction. Longer trainings of the tagging allowed the network to adjust this accordingly, e.g., to find other structures, like overlapping hits. This had the disadvantage of potentially overfitting the tagging because of the long training time. Training of both parts separately was also tried but did not give good results.

Fig. 3.17 shows the performance of the 3TCNN_376 on energy reconstruction. The first observation on this is that the low-energy reconstruction got better. In the left plot, it can

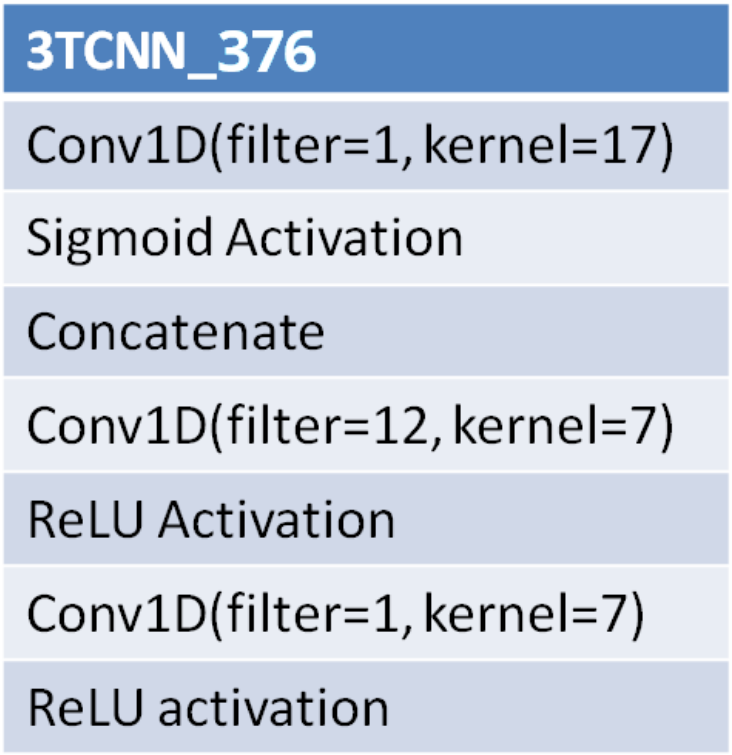

Figure 3.16: Architecture of the 3TCNN_376.

be seen that the network increased its resolution for low energies close to 0. On the former networks, the performance for energies under 1 GeV did not change much. An explanation is that the tagging of hits gives an advantage here. Especially very small energies improved. This could mean, that this network learned to better deal with background or pileup.

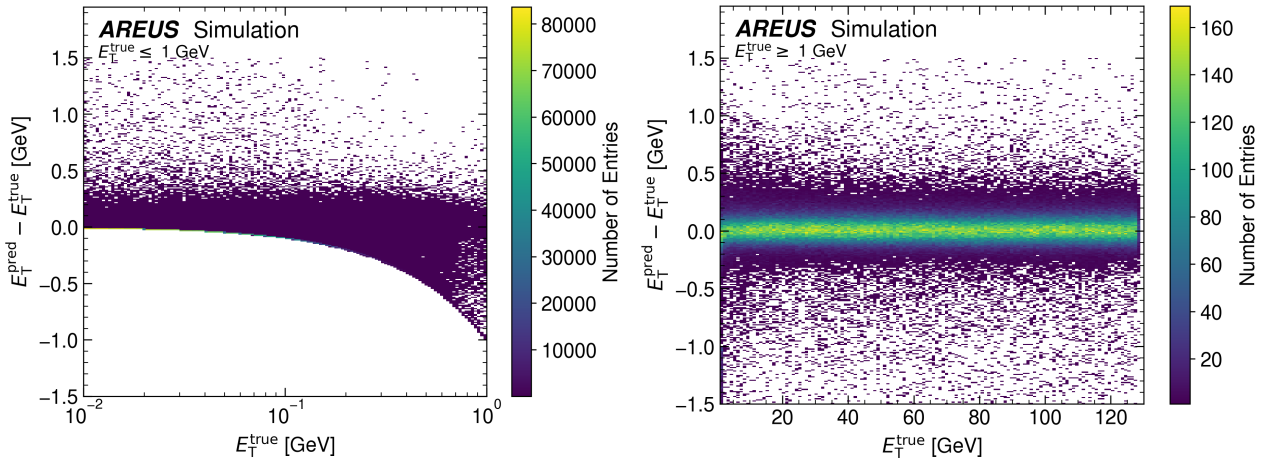

Figure 3.17: Difference of the energies predicted by the 3TCNN_376 with the true energies.

For energies above 1 GeV, the distribution is comparable to the 2CNN_391 but with a slightly worse resolution. Overall, the distribution is constant for all energies over 1 GeV.
In Fig 3.18 seen is the performance of the network, measured with mean value with standard

deviation and median with 98% range shown again. This network has the most constant 98% range of all the networks tested so far. For energies above 10 GeV, this value only slightly varies, and there are no regions with an overly increased range. The values of the range are larger than for the 2CNN_391 for most energies, though. Its range is at a mean value of around 0.46 GeV. Under 10 GeV, the lower range increases a lot again because of the large number of underestimations for smaller energies.
The standard deviation of the network is similar to the 2CNN_391 at a value of around ±0.083 GeV. This is the same value as for the OF and 2CNN_391.

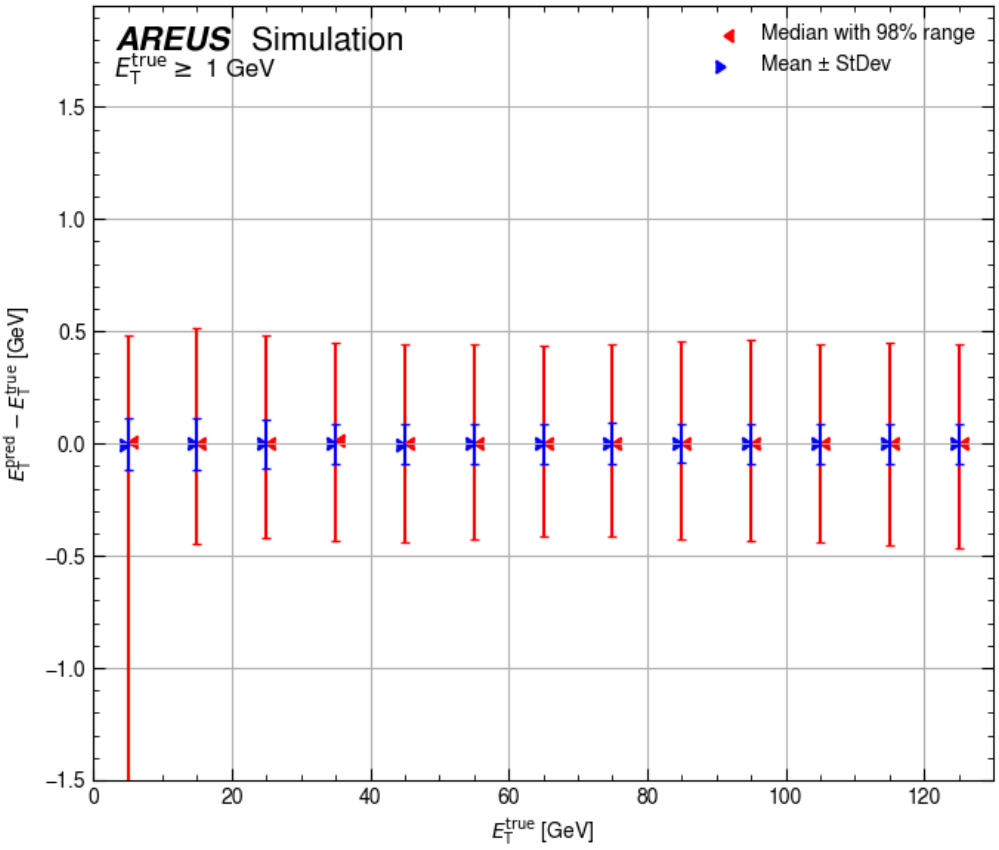

Figure 3.18: Mean value with standard deviation and median with 98% range of energies reconstructed with the 3TCNN_376.

In Fig. 3.19 is the performance on subsequent hits shown. This network performs better than the 2CNN_99 and 3CNN_396, but worse than the 2CNN_391. The distribution is overall good with a low amount of outliers until 20 BC, and the network performs comparable to the 2CNN_391 for close hits. Its resolution for gaps between 20 and 40 BC is also less smooth compared to the 2CNN_391.
This architecture performs well and outperforms the OF and all other tested networks, except the 2CNN_391. Its performance is slightly worse overall compared to the 2CNN architecture, but its performance is less varied for different energy ranges.
Of all the tested networks with 400 parameters, the 2CNN and 3TCNN architecture performed

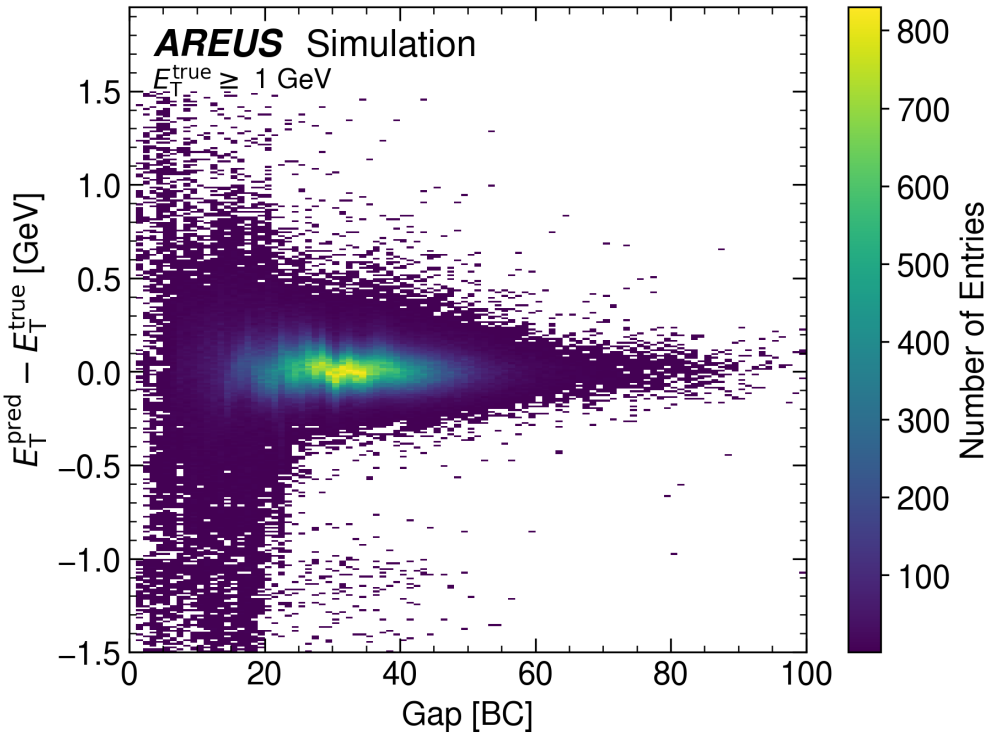

Figure 3.19: Energy differences of predicted energies by the 3TCNN_376 quantized to 18 bits and true energies, plotted against the distance between two hits in BC.

the best, with 2CNN being the best. All tested networks could outperform the OF and the 2CNN_99, in terms of overall energy reconstruction.

3.5 Quantization of 3TCNN to low bit Widhts

The 2CNN_391 was the best-performing network and could be quantized to 14 bits. This enables to save resources on the FPGA. Saving additional resources could be possible, though. Other networks might behave differently under quantization. For this reason, the 3TCNN_376 was also trained quantization aware.

It was chosen over the 3CNN_396 because its performance was better, and it was assumed that it would behave differently under quantization because of the different structure. One possibility is that tagging can help to negate the reduced precision from the low bit width.

In Fig. 3.20 is the performance of different 3TCNN_376 shown, trained quantization aware to different bit widths. The performance of the network is constant, down to 14 bits. Standard deviation is constant, and this range and 98% only slightly varying. At lower bits, the network's bias increases and varies towards over or underestimating energies. This could result from instability because of the quantization, since the total performance is stable. More trainings at this bit range might give networks with a lower bias. Resolution and 98% range are regardless stable until 11 bits. Starting with 10 bits, both values increase, and the OF outperforms the networks.

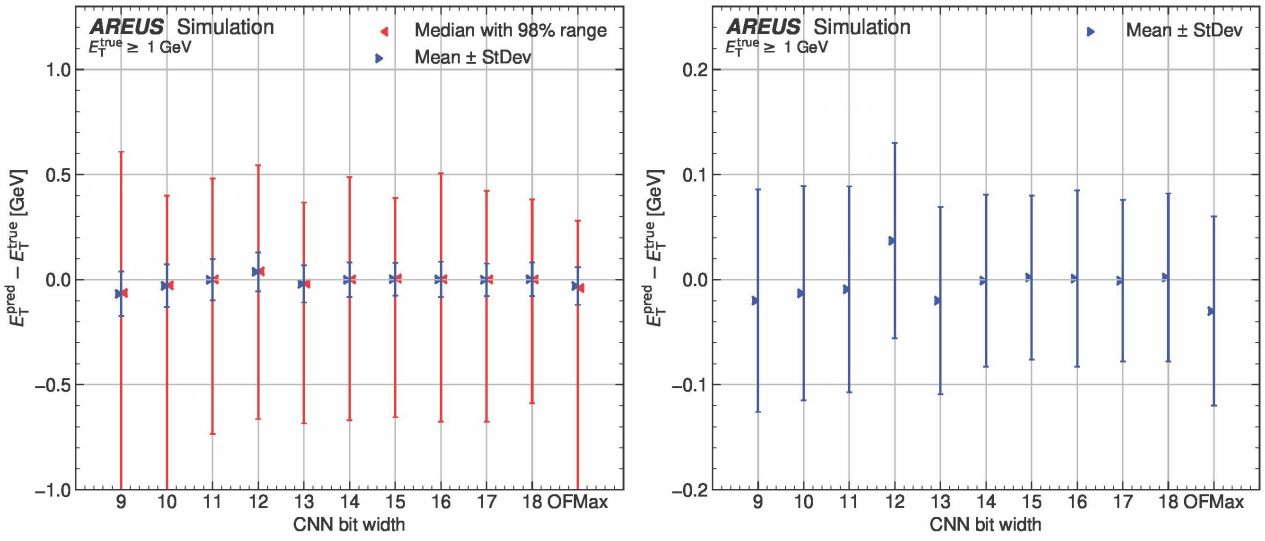

Figure 3.20: Performance of the 3TCNN_376 quantized to different bits. On the left side is the mean value with standard deviation, calculated via a Gaussian fit. On the right is only the standard deviation for a better visibility. The bit number shown is the total bit width of the weights in the network. Activations functions and data were always to 16 bit.

Fig. 3.21 shows the distribution of the reconstructed energies for different bit widths. It can be seen that there is no significant difference between the resolution of the full 18 bits and quantization to 11 bits. The 11-bit network has a slight bias towards underestimation, which can be seen in the distribution.

The performance on subsequent hits of the network quantized to 11 bits can be seen in Fig. 3.22. Compared to the 18-bit performance in Fig. 3.19, this network is losing more perfor-

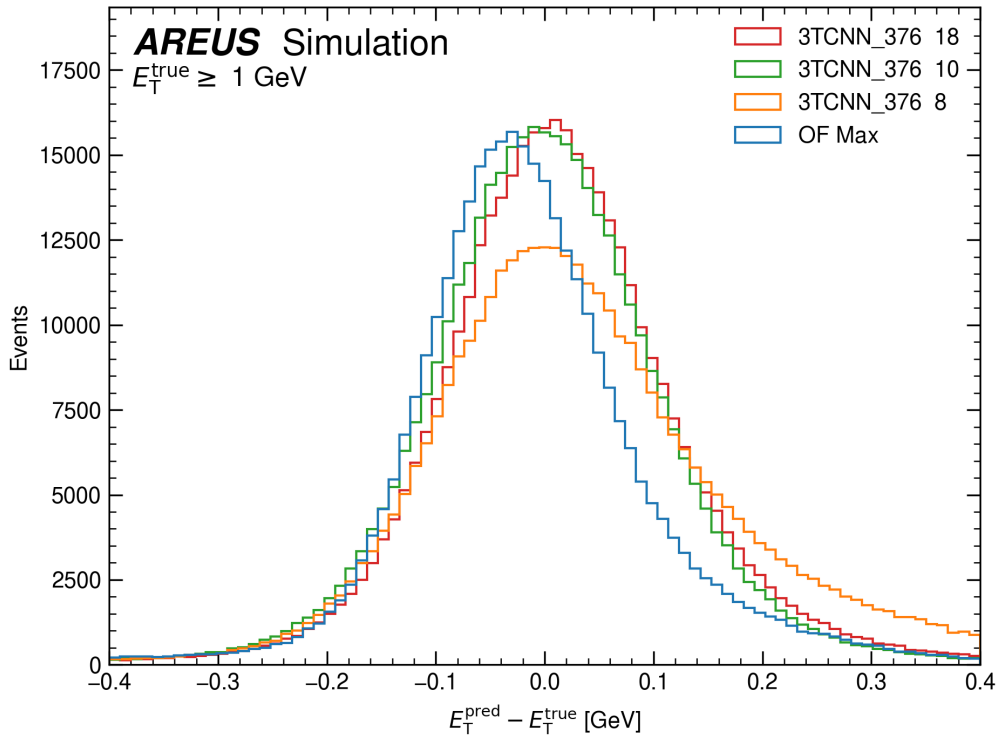

Figure 3.21: Distribution of the reconstructed energy residuals of the 3TCNN_376.

mance. Both networks perform relatively equally for gaps over 20 BC. For lower gaps, when the pulses overlap more, the resolution of the 11-bit networks worsens, and the number of outliers increases. It still has an advantage compared to the OF for close hits, though, but the decrease is worse compared to the 2CNN_391. This has to be kept in mind if these networks will be implemented. Going down to 11 bits might not be optimal, even with an overall good performance for energy reconstruction.

In principle, this makes quantization of this network down to 11 bits possible if a trade-off in performance on subsequent hits is acceptable. Otherwise, 14 bits are possible without degradation in performance. This offers potentially a bigger decrease compared to 2CNN networks. Even when quantized to 11 bits, the total consumption will be higher, though. The reason for this is the sigmoid activation. This function is computationally expensive compared to linear functions or ReLU. To reduce this, linear approximations of Sigmoid were tried to work on the network. If a linear approximation can keep the performance of Sigmoid, it would be possible to implement very efficient and good-performing networks.

There are multiple approximations that can be tried as an alternative. For technical reasons on the FPGA, only linear functions with a slope that is a power of two can save good amounts of resources. This limits the choice of functions, so two approximations were tested. PLAN-Sigmoid is a piecewise approximation that was designed to be a computationally cheap alternative to Sigmoid [50]. The function and its derivative were designed with a slope that is a power of two. A common problem of linear approximations is that the derivative is not

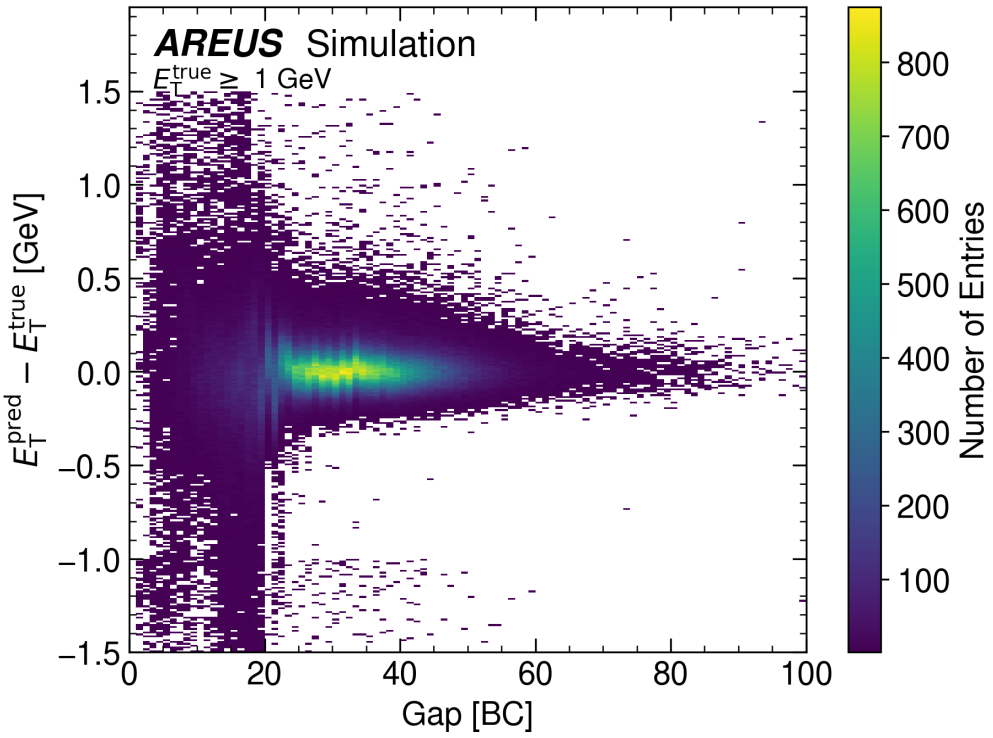

Figure 3.22: Energy differences of predicted energies by the 3TCNN_376 quantized to 11 bits and true energies, plotted against the distance between two hits in BC.

continuous, which can lead to problems with gradient descent optimization. PLAN-Sigmoid was designed with this in mind. The function can be seen in Fig. 3.23.
As an alternative, Hard-Sigmoid was tried:

$$\text{Hard-Sigmoid}(x) = \begin{cases} 0 & \text{if } x \leq -\frac{1}{2} \\ \frac{1}{2}x & \text{if } x \in (-1, 1) \\ 1 & \text{if } x \geq \frac{1}{2} \end{cases} \tag{3.1}$$

This is a very rough approximation, but it keeps the properties of Sigmoid (saturation at 0 or 1 and linear around 0) and is used for comparison.
Both functions were again quantized to 18-bit first. This quantization should be no problem for Hard-Sigmoid but might influence PLAN-Sigmoid, since it was not designed with quantization in mind. A linear approximation of Sigmoid that is designed for low-bit quantization would be useful but was not available during this work.
Another thing that could be changed, to make network even more efficient, is separate quantization of the tagging and energy reconstruction. Tagging networks can work with much fewer bits compared to regression networks [51].
For this reason, the 3TCNN_376 was trained with very low bits on the tagging part, while keeping bits for energy reconstruction high and approximations of the sigmoid function.

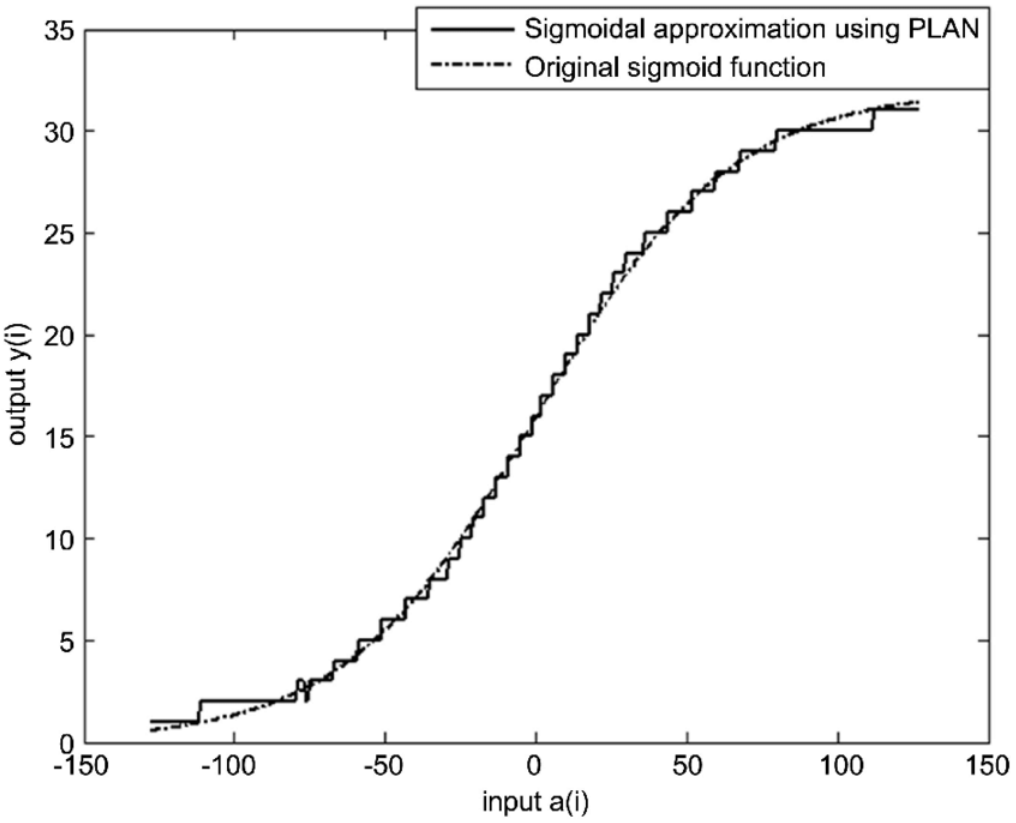

Figure 3.23: PLAN-Sigmoid function is a linear approximation of Sigmoid, which is supposed to smoothly approximate Sigmoid for hardware implementation.

First, the 3TCNN_376 was trained with low tagging bits. This was done by pretraining the tagging part only. It is assumed that networks with equal tagging perform equally well on energy reconstruction. This performance will be later shown for low tagging bits. In Table. 3.1 is the tagging performance shown for different bits.

Table 3.1: Performance of quantization aware trainings of the tagging network for the 3TCNN_376 for different bit widths.

Bit width of tagging weights	Accuracy	BCE
18	95.4	0.311
14	95.4	0.314
10	95.4	0.313
9	95.4	0.316
8	95.4	0.314
7	95.4	0.320
6	95.4	0.320
5	95.4	0.325
4	91.6	0.659
3	82.3	0.816

It is measured with two metrics. First is the accuracy, which is the percentage of correctly tagged positions. Second, the Binary Cross Entropy (BCE) was used to measure the uncer-

tainty of the tagging. Shown are the results for the pretraining only. After the full training, the network is not necessarily optimized for correct tagging of hits but for energy reconstruction, which makes these metrics less useful. The table shows that the accuracy is stable until 5 bits. For lower bits, it is dropping. The cross-entropy on the other side is slowly increasing with lower bits. What this means can be seen in Fig. 3.24. Here, the tagging distribution is seen for 10 bits and 5 bits. The 10-bit tagging has a high and narrow peak for correctly tagged hits and an asymmetric, flat distribution for wrongly tagged hits. Wrongly tagged for this network mostly means that it assumed a hit where no hit was in the sequence. Missed hits did not happen in large quantities, which is shown in the asymmetry of the distribution. At 5 bits, this changes into a structure with two peaks. In the center, the peak for correctly tagged hits transforms into a Gaussian distribution, with some width. There is now a new peak for wrongly assumed hits in the sequence. Also, the network is now missing some hits.

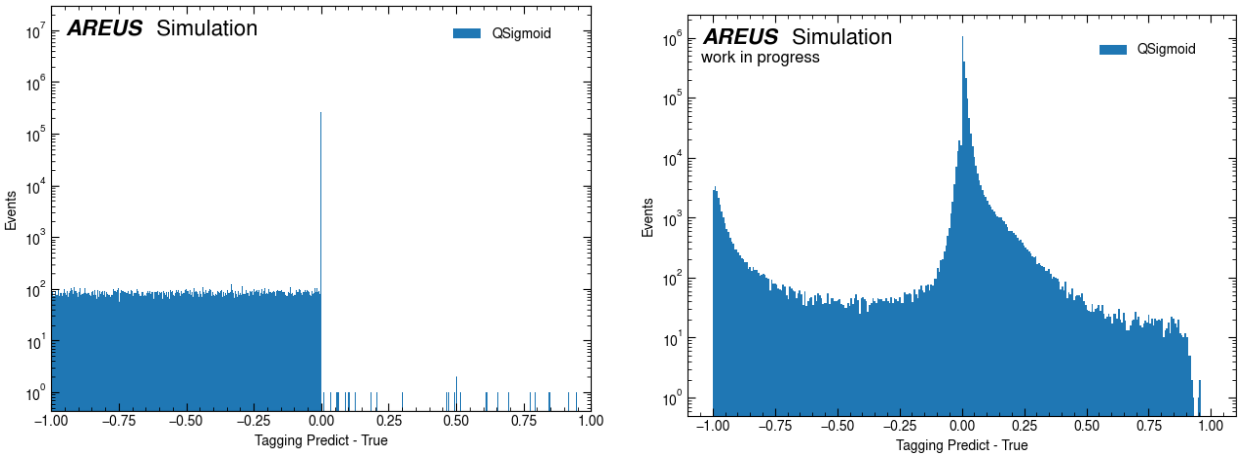

Figure 3.24: Performance of the pretrained tagging of the 3TCNN_376, trained quantization aware to different bits. On the left is the network quantized to 10 bits, and on the right to 5 bits.

This changes the output of the tagging and might result in errors during energy reconstruction. Since the total accuracy is constant, though, it is also possible that the network performs equally well. The 3TCNN_376 trained quantization aware with 14 bits used for energy reconstruction and six bits for tagging can be seen in Fig. 3.25. Comparing this with Fig. 3.17 shows this network's increased amount and range of outliers. This increase becomes larger for higher energies and would worsen the resolution. Comparing the performance on subsequent with 3.19 shows no significant decrease for large gaps. The distribution becomes worse for gaps under 30 BC, and the number of outliers increases much for gaps under 20 BC. These differences together could mean that the 3TCNN_376 becomes worse in reconstructing close hits with high energy, with fewer bits available. The overall performance in energy reconstruction is constant, though. This could mean that a good performance on these hits is expensive in terms of parameter usage for the network. Fewer bits make the network less precise on high-energy hits with an overlap, which might make it easier to focus on lower energies.
This shows that saving additional bits on the tagging network is possible but comes with a worsened performance on close hits. The performance is better than the OF or 2CNN_99, but worse than the 2CNN_391.

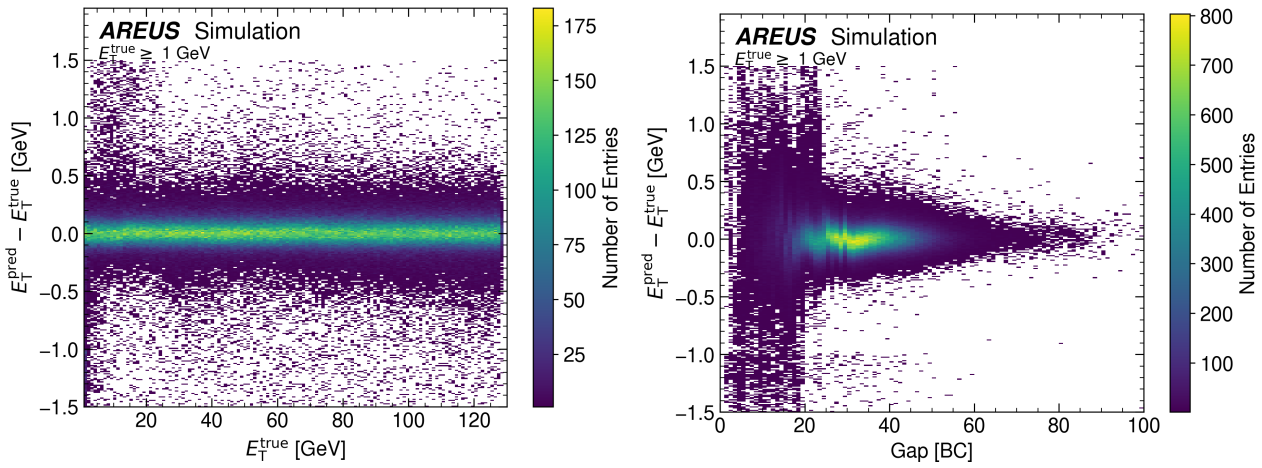

Figure 3.25: Performance of the 3TCNN_376, trained quantization aware to 14 bits for energy reconstruction and six bits for tagging. On the left is the energy reconstruction seen, and on the left is the performance on subsequent hits.

As a next step, the activation function was changed into a more hardware-friendly version. For this, the network was trained 20 times with a Hard-Sigmoid activation and 20 times with Plan-Sigmoid. The best of these trainings can be seen in Fig. 3.26. Hard-Sigmoid did not perform well on any networks. The best result was with PLAN-Sigmoid, but this led to a drastic decrease in performance. It can be seen that the resolution on energy reconstruction decreased while the number of outliers increased a lot. The performance on subsequent hits also deteriorated. Specific for these trainings are error structures that can be seen for different energy ranges. They originate from the design of PLAN-Sigmoid. It is a linear approximation made up of many different linear functions. This results in constant derivatives with jumps between. During prorogation for the training, this can create different loss regions, with discontinuous transitions, which results in these errors.
PLAN-Sigmoid was designed with a derivative to cope with this effect, but it was designed for using full 32 bits. Quantization might have decreased the performance of the function. For the same reason, quantizations of Sigmoid under 16 bits were not tested. A test training with full 32 bits and PLAN-Sigmoid showed good performance.

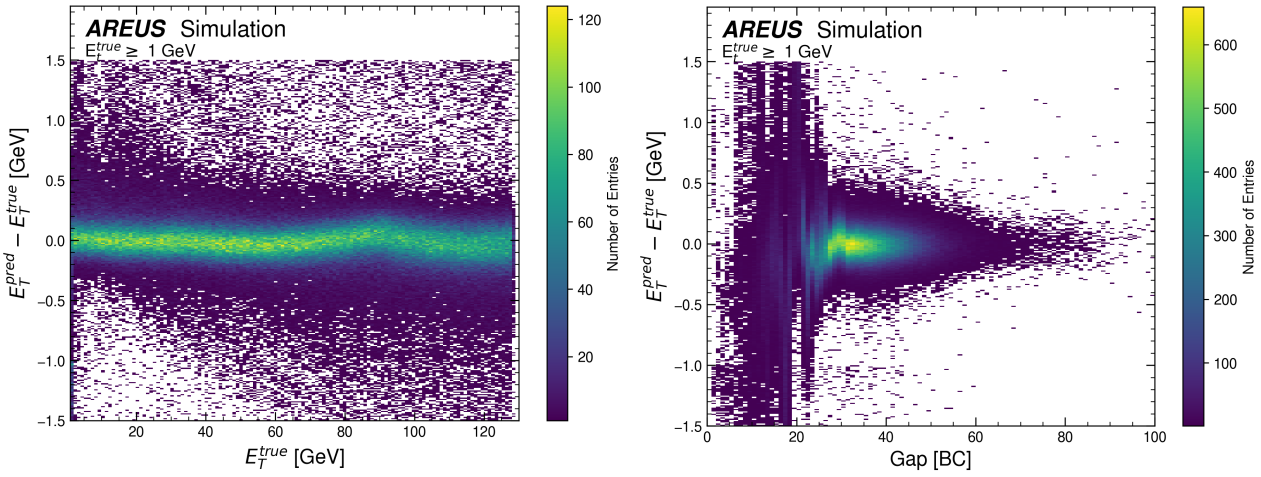

Figure 3.26: Best performing 3TCNN_376 network with PLAN-Sigmoid activation and quantized to 18 bits.

In summary, quantization of the 3CTNN_376 is possible down to 11 bits, with stable performance on energy reconstruction. This enables to save more bits compared to tested 2CNN networks. However, the performance is worse, and the networks must use Sigmoid, significantly increasing ALM consumption. A linear approximation of Sigmoid that can work with low bits would be beneficial for this network.

3.6 Performance Comparison of Neural Networks for Different Datasets

The networks in this thesis were trained with a combination of different data, depending on the timely distance between hits. The evaluation was done only for random gap data until now, which have a Gaussian distribution of the gap sizes. In the next part, the best working networks were compared in their performance on the other data sets. As a reminder, other used data are constant gap, with a gap size of 30 and uniform gap, with a uniform distribution between 1 and 80 BC. The networks tested were the 3CNN_396, 2CNN_391, 3TCNN_383. For comparison, shown are also the 2CNN_99 and OF.
Also used for the training were only pile-up data. An evaluation on these was not done in this thesis because they show no actual hits. Looking at the performance on these might be interesting in the future because it would show how well the network learns the background.

3.6.1 Uniform Distributed Gaps

In Fig. 3.27 shown is the distribution of the reconstructed energies for the uniform data. This dataset has a relatively large amount of very close hits and should, therefore, show the performance on these better. Included are also huge gaps between hits, making them isolated from each other without any overlap. To evaluate the performance on those isolated hits, constant gap data are looked into in the next section.
In Fig. 3.28 shown for comparison is the 98% range of the networks. Here, it can be seen first that all networks and the OF have a relatively high 98% range, in the order of $\pm$ 30 GeV, compared to a very small standard deviation. This is because of the high amount of very close hits in this set. These create relatively large numbers of outliers.
Looking at the different networks shows that the 98% range for the 2CNN_391 is asymmetric. It is the best out of all networks in the positive range, but in the negative range, it is worse than the other networks with 400 parameters. This means that this network tends to overestimate the energies. An explanation is that this network might have a worse performance on close hits. The other networks all had more layers, which might have helped them on overlaps.
The 3TCNN_376 has the most symmetric distribution and performs well on these data. It also has the lowest and most symmetric 98$ range. The reason for this is, most likely, that this network performs better on very close hits when quantized to 18 bits. Its performance on isolated hits suffers on the other side. This would give this network an advantage when many close hits are expected.

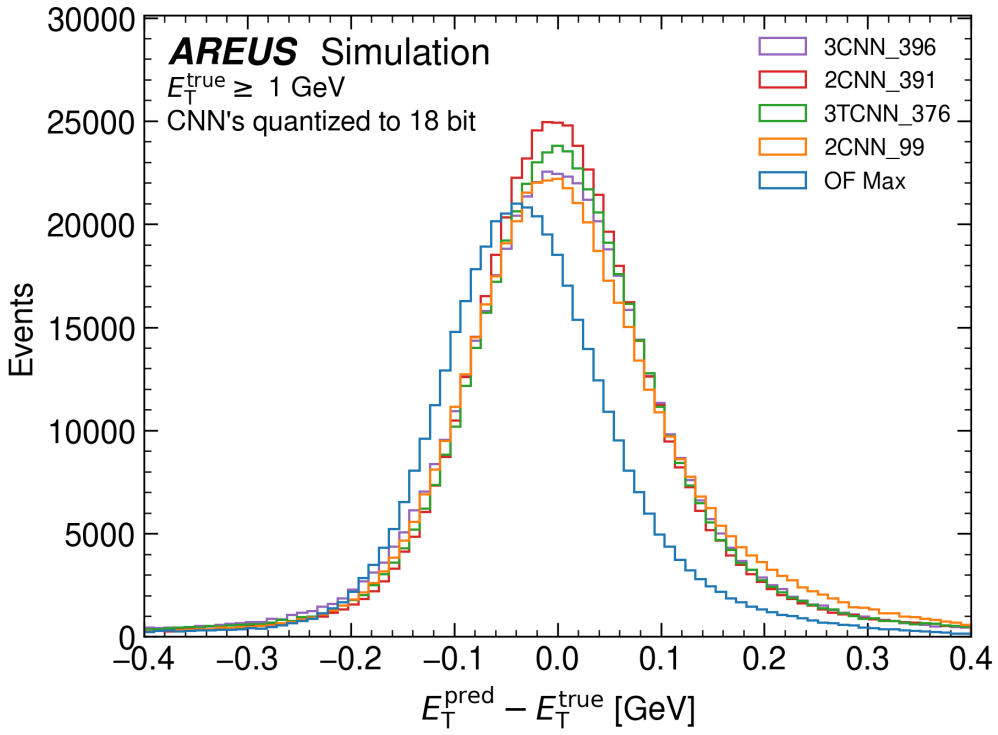

Figure 3.27: Distribution of the reconstructed energies for the 3CNN_396, 2CNN_391, 3TCNN_376 and 2CNN_99 networks for ADC sequences with a uniform gap distribution between hits.

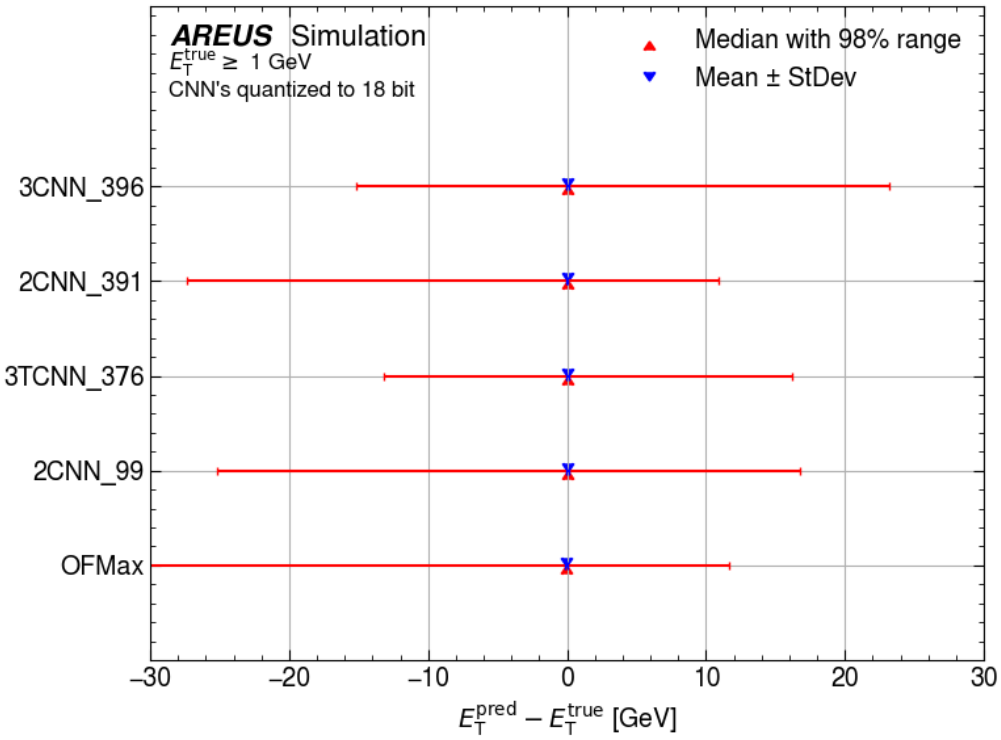

Figure 3.28: The performance of neural networks for ADC sequences with a uniform gap distribution between hits, measured as mean value with standard deviation and median value with 98% range.

3.6.2 Constant Distributed Gaps

Constantly distributed gaps are hits with a distance of 30 BC between them. These act as isolated hits without any overlap. As a reminder, the OF is mathematically proven to be the best linear algorithm to work on those. Neural networks were technically designed to work on close hits, as expected for the Hl-LHC. This part will regardless look into closer hits to compare the networks with the OF on those.
In Fig. 3.29, the energy distributions for the algorithms on constant gap data can be seen. Fig. 3.30 shows for comparison the 98% range again. When working with constant gap data, the networks have a relatively low 98% range but a similar standard deviation to random gap data. They also perform very similarly on these data. This is because these data correspond to isolated hits, which should provide no challenge. The 2CNN_391 and the 3TCNN_383 have overall again the best performance. Both have a very similar distribution with equal height and width. Their 98% range is equal, but the 3TCNN_376 has again a more symmetrical one. The 2CNN_391 tends to overestimate hits here.
The OF performs equally well on these data as the 2CNN_391 and 3TCNN_376 This shows that neural networks do not offer any significant advantage on those hits. Regardless, they can achieve OF performance while also improving performance on overlapping hits.
The 3CNN has been shown to work well on random gap data. As seen in this section, its performance suffers when working on other data types. In uniform and random gap data, it had the worst performance out of all networks with an increased number of parameters. A possible interpretation is that this network works best on only slightly overlapping hits, which are common on random gap data.
On isolated hits, the OF as a linear function is the best working linear algorithm. The three-layered neural network, on the other side, might be overfitting on those hits because it is non-linear, and this does not offer any advantage here. On very close hits, it is also underperforming. Here, the 3TCNN and 2CNN networks outperform it. Both of these networks use two layers for energy reconstruction. This can mean that two layers for energy reconstruction offer the best performance on close hits. Adding more layers seems to make the networks more vulnerable to overfitting.
On only slightly overlapping hits, this network has a good performance. It is possible that these overlaps create the most complicated pulse shapes. When working on those, easier non-linearity of the network might be an advantage. A more detailed look into the resulting pulse shapes might be interesting for further analysis.
Overall, this demotivates further training of deeper networks in the future. Two layered networks for energy reconstruction and maybe a tagging part seem optimal. Depending on the expected gap distribution for the HL-LHC, the 3CNN architecture might still hold an advantage over others though.

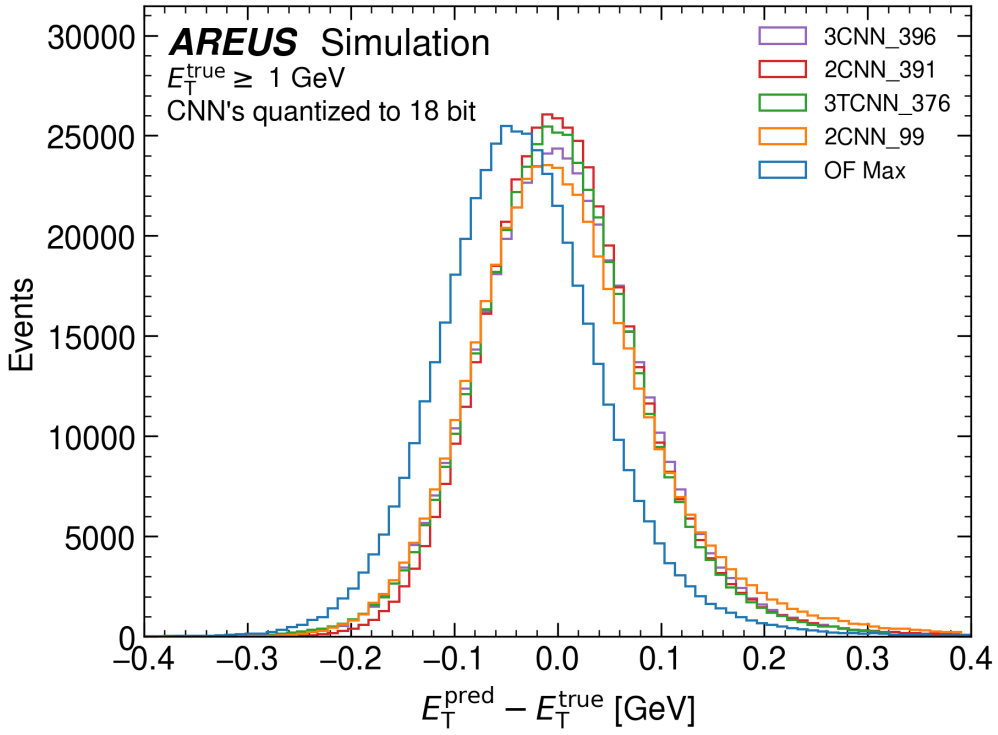

Figure 3.29: Distribution of the reconstructed energies for neural networks for ADC sequences with a constant gap distribution between hits of 30 BC.

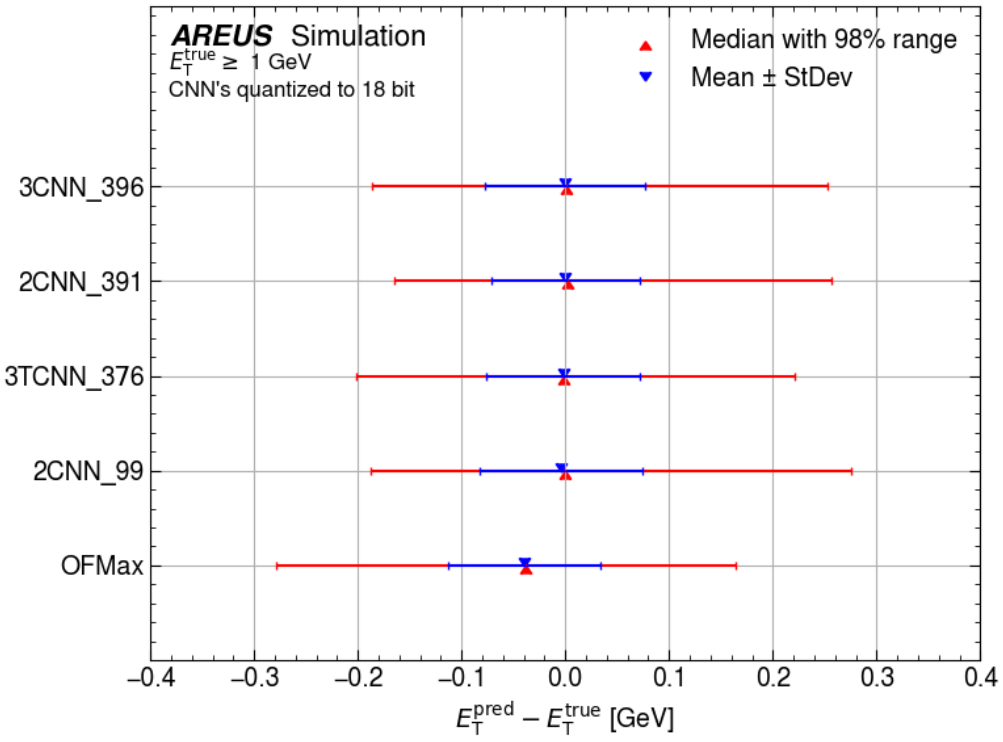

Figure 3.30: The performance of the 3CNN_393, 2CNN_391, 3TCNN_376 and 2CNN_99 networks for ADC sequences with a constant gap distribution between hits of 30 BC is shown, measured as mean value with standard deviation and median value with 98% range.

4 Performance Evaluation of Neural Networks with Starplots

In this section, the performances of the networks that were trained and tested in this thesis are now compared by using different metrics via starplots. Since the 22CNN_391 was the best-performing network and should have the lowest resource consumption, this section will focus on this network.

In order to evaluate and compare the networks, there is a need for a baseline. This is the Optimal Filter. There is also a need to know which performance measurements are important and the goals for analysis. This is not given at the moment and is still subject to change. In order to still provide some final comparison, starplots are used to give a broad overview of the performance.

Fig. 4.1 shows the first starplot for the performance of the 2CNN_99 quantized to 18 bits. This will serve as an explanation for the starplots.

Multiple performance measurements, called scores in this thesis, are ordered circularly. Every score is meant to compare the network with the OF. Scores are calculated as:

$$\text{Score}_i = \frac{\text{Performancee}_i^{\text{OF}} - \text{Performance}_i^{\text{NN}}}{\text{Performancee}_i^{\text{OF}}} \tag{4.1}$$

With the performance measurement i, its value for the Neural Network $\text{Performance}_i^{NN}$ and for the OF $\text{Performance}_i^{OF}$. A score of 0 means that the performance of the network is equal to the OF. The OF performance can be seen as a red circle around the middle of the starplot. For positive values above the red circle, the network is becoming increasingly better than the OF, and for negative values inside the circle, it's worse.

The starplots used have, in principle, four different measurements. Most scores are calculated for different ranges of $E_{\text{T}}^{\text{true}}$ and BC, to differentiate the performance for different energy values and timely distances between them. The range $[1, 130]$ GeV is supposed to measure the performance for high energy. 1 GeV as a cut between low and high energies was chosen arbitrarily. $[0.20115, 1]$ GeV is the low energy range. 0.20115 GeV is the expected value of noise in the LAr, which is why it was used as an additional cut. For the bunch crossings, hits with a distance over 31 BC are so far away that the OF has no problem dealing with them. Since the OF is mathematically proven to be the best linear transformation on LAr pulse shapes, it can be expected that the networks have problems getting a better score in this range. In the range of $[6, 31]$ BC, the OF drastically loses performance because of overlapping hits. This is the range where the networks should have the biggest improvements compared to the OF. Hits under 6 BC are so close together that the OF cannot deal with them anymore.

The first score is the mean value of the energy residuals $\overline{\Delta E}$; see 1 o'clock for its value in the low energy range. This score is calculated normally or with a Gaussian fit, depending on the

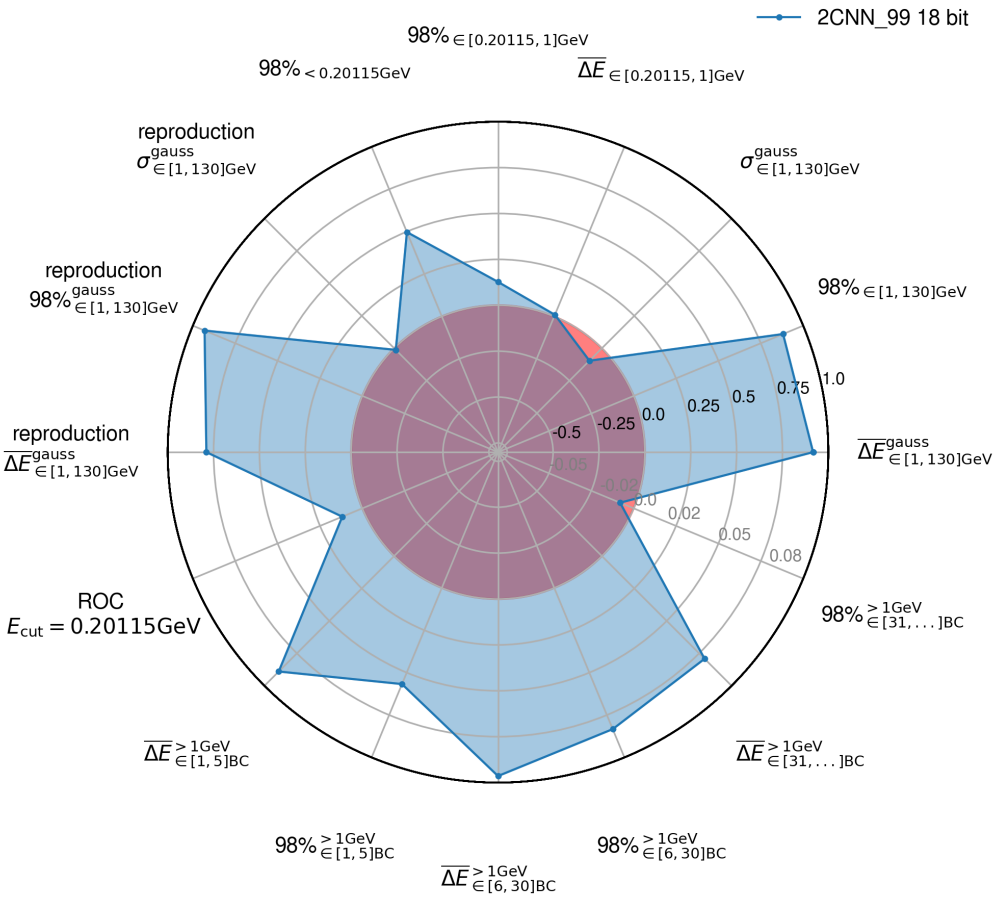

Figure 4.1: Starplot to compare the performance of the 2CNN_99 network, quantized to 18 bits, with the Optimal Filter

index. This value represents the bias of the algorithm. Ideally, this should be $\overline{\Delta E}$ close to 0. The OF always has a bias towards underestimation, as could be seen in the former chapters, so it can be expected that the networks have a better performance in this score.

Next score is the corresponding standard deviation σ calculated as a Gaussian fit for high energies; see 2 o'clock for an example in the high energy range. This is supposed to measure the resolution of the algorithm. When evaluating the networks, it could be seen that most networks had a similar resolution to the OF. Achieving a better resolution is difficult because the OF was optimized at this value.

Since the resolution is calculated with a Gaussian fit in the high energy range, it includes no information on outliers or low energies. To evaluate these, the 98% range was used as a score; see 3 o'clock for an example. This is again calculated by ordering the values and cutting the highest and lowest percentages. The 98% range is given for different energy and BC ranges. On low BC and most all energies, the networks usually had a better 98% range than the OF. Reason for this is that the OF tends to underestimate close hits. Networks should have good scores here, especially for close hits, because the best networks were chosen based on this performance.

The only exception is the 98% range on isolated hits, with a distance over 31 BC. This score can be expected to be troublesome again for networks because the OF was designed for those isolated hits. In the former chapters, it could be seen that the best working networks had a similar performance to the OF on those hits, but no big gain was observed.

The last performance measurement is the ROC (receiver operating characteristic) curve, which measures the background suppression. It is calculated by determining the area under a curve (the ROC curve). This curve is created by plotting the true positive rate against the false positive rates. A large area, therefore, shows a better performance.

All the scores and plots discussed so far have been calculated only for the best network out of 20 trainings. In practice, the best one will be used, of course, but there is a need to look into the reproducibility of the networks. Since neural networks are trained stochastically, the best network may be a singular outlier that is very unlikely to be repeated. This would be a problem because the networks optimized in this work will not be the last ones made for the LAr calorimeter. In the future, the networks need to be optimized on different data. For example, the networks in this work were only trained on one out of 180.000 detector cells. The performance will drop for other cells because they have other pulse shapes. In the future networks have to be trained on other cells.

Measuring the reproducibility was done with additional reproduction scores. These scores are calculated over all the 20 networks from the training. Excluded were only trainings that have not converged its loss function. These networks had very bad loss values and did not perform well. In the good working bit ranges, those networks only made up single networks from a training, which would prove no problem in practice. With their bad performance, they would

decrease the reproduction score significantly. Since those networks did not appear in large quantities, they were ignored for this. At low bit ranges, those trainings happened with much bigger frequency, but the networks did already perform poorly at those ranges.

A good reproduction score means that the corresponding score is equal for all training. Bad reproduction scores mean that the values are changing much in the training.

As an example, see Fig. 4.1. The performance of the best 2CNN_99, quantized to 18 bits, can be seen here. Most parameters are better than the OFMax. The scores for close and overlapping hits are all better than the OF. This means that the network, even with 100 parameters, succeeded in improving this performance. Much better than the OF are also the 98% range and the Gaussian mean value for high energies (see 3 o'clock). This translates into a reduced bias of the network and fewer numbers of outliers during energy reconstruction.

The ROC curve is better than the OF but close to it. This means that the networks also have an increased background suppression. The networks were not directly trained on this, but achieving this is also a success.

Very close to the OF is the $\sigma^{gauss}_{[0.20115,1]GeV}$ score. This measures the bias at low energies. It could be expected that both networks are close here because both algorithms can not give out negative values. Since the OF tends to underestimate hits, this value is forced closer to 0.

Worse than the OF were the $\sigma^{gauss}_{[1,130]GeV}$ score at 2 o'clock, which is the resolution, and the $98\%^{1\mathrm{GeV}_{[31,\ldots]BC}}$ score at 4 o'clock. These scores were expected to be difficult for the network. Both scores are close to the OF, though. One motivation to use larger networks was to improve these scores. The $98\%^{1\mathrm{GeV}_{[31,\ldots]BC}}$ score is zoomed in, which means their axis does not show the whole range. This is because this score was very close to the OF. These zoomed axes are always marked with the new scale.

When looking at the corresponding reproduction score, it can be seen that the σ^{gauss} score for high energies is not very good. This is one of the scores where the networks also struggle to compete with the OF. Being worse on reproduction means that the networks are varying a lot or that most networks had a worse performance. It does not mean that all networks were much worse. Some other networks with a higher loss might have been better in this regard. The other reproduction scores are very good in comparison, which means that the other σ scores do not vary much.

For comparison is the starplot of the 2CNN_391 quantized to 18 bits shown in Fig. 4.2. Important when comparing two starplots is, that scores that were already superior to the OF will not change much, even when the performance increases further. E.g., the 98% range has increased with more parameters, as seen in Fig. 3.18. The corresponding scores, however, haven't changed because the starplot compares only with the OF. If the scores were already high, the starplot would no show much difference. For this reason, the gain of using more parameters can only be seen on formerly bad scores. If wanted, it would be possible to modify the starplots to compare with the 2CNN_99.

The 2CNN_391 network outperforms the OF in all scores except the 98% range for high BCs. In this area, it is slightly worse and has not gained much performance. This was expected, as already said because the OF was optimized for exactly this score. Every other score is, however, better for the 2CNN_391. It is particularly good in mean energy scores for high energies and all BCs. Especially its performance for lower BCs is much better than the OF. Even in background suppression, it outperforms it.
One performance that the CNN could still improve is its Gaussian sigma score for high energies. Here, it is again only slightly better than the OF, and the corresponding reproduction score is relatively low, which means this value can vary.

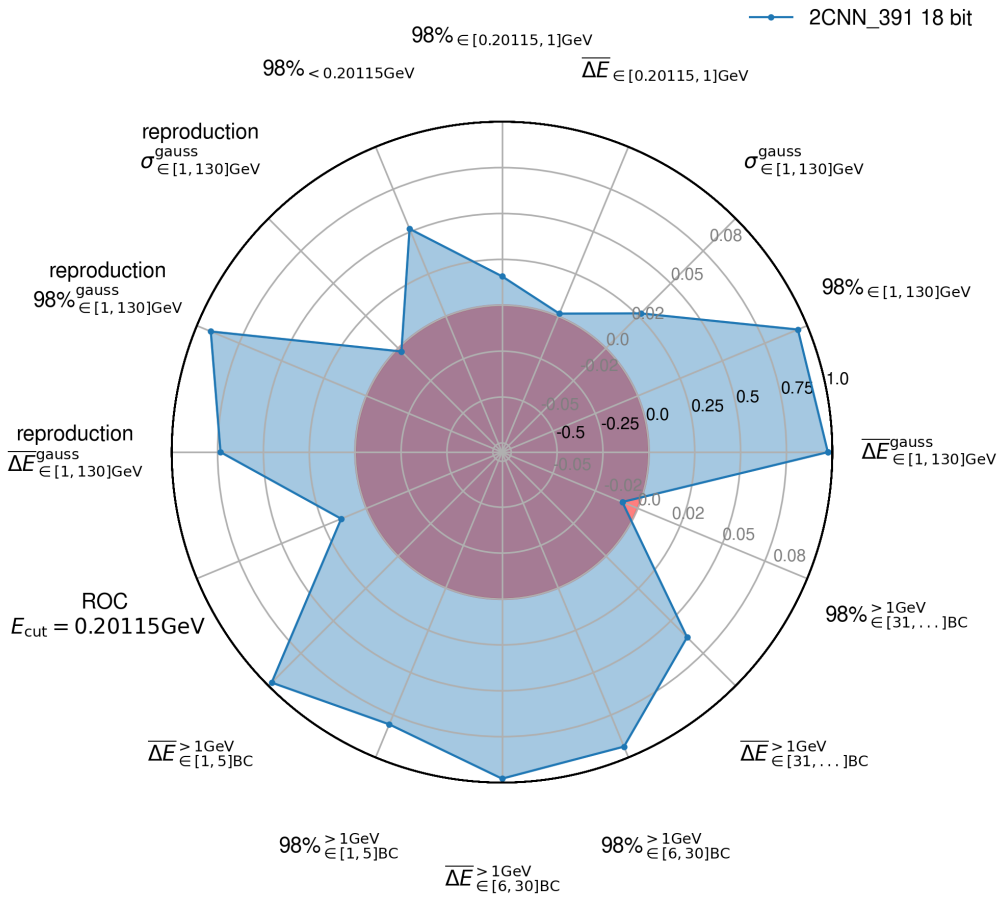

Figure 4.2: Starplot to compare the performance of the 2CNN_391 network, quantized to 18 bits, with the Optimal Filter)

In Fig. 4.3 shown is the performance of the 2CNN_391 quantized to 15 bits. Comparing it with Fig. 4.2 shows that both versions of the network have a very similar performance. In some scores, the 15-bit network is even better compared to its 18-bit version. This comes from the stochastic nature of the training, which creates some variations. Looking at the reproduction scores shows that all of the 20 networks have a similar performance. This means that this 15-bit network is not some singular training and that its performance is reproducible.
With the starplots, it can be seen that quantization does not provide any significant loss in performance for the 2CNN_391.

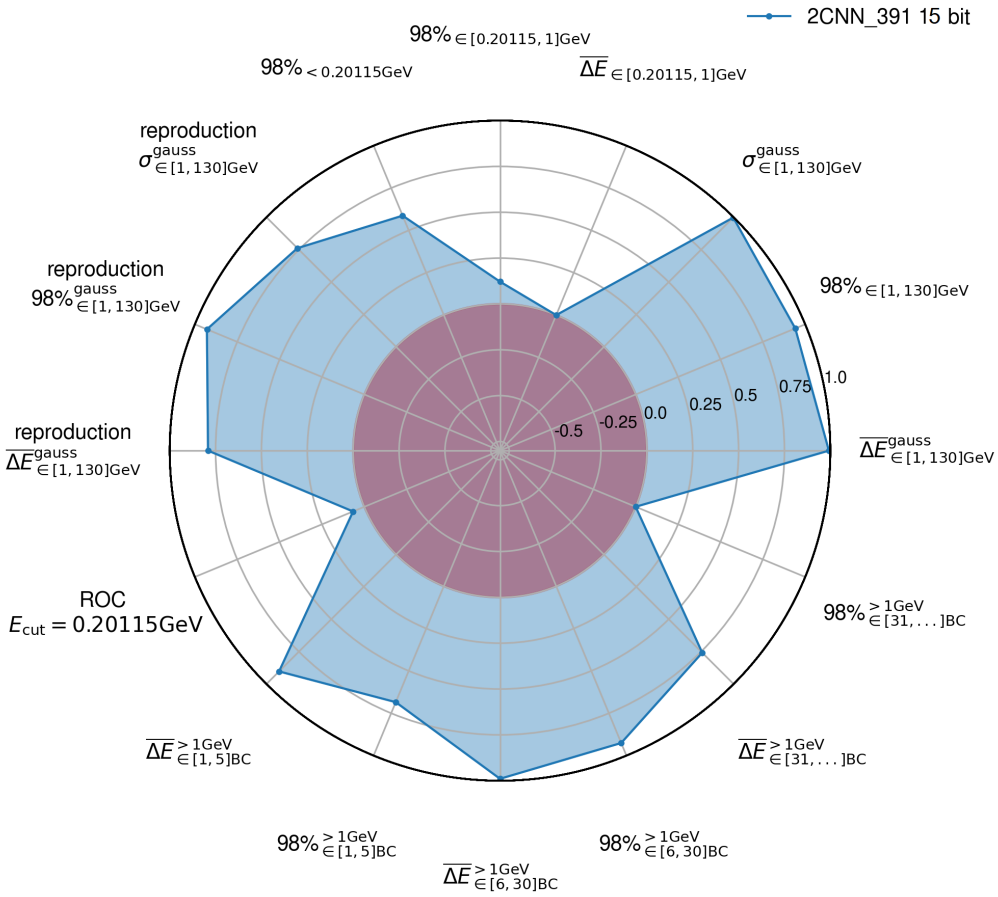

Figure 4.3: Starplot to compare the performance of the 2CNN_391 network, quantized to 15 bits, with the OF.)

For comparison and to see the influence of quantization, a starplot for the 2CNN_391 with 13 bits is seen in Fig. 4.4. This is again the best out of 20 trainings. The worsening of some scores can be clearly seen here. Mostly degraded have the σ^{gauss} score for high energies, the corresponding reproduction score, and the $\overline{\Delta E}$ for low energies. These scores are now under the OF performance. This shows that the resolution of the network decreased.
Even with this worse performance, it still outperforms the OF in other scores, especially for the performance of overlapping hits. The worsened scores are also very close to OF performance. If additional computational resources need to be saved, it might be worth looking into these

lower bit widths and take the worsened performance in some scores as a trade-off for more savings.

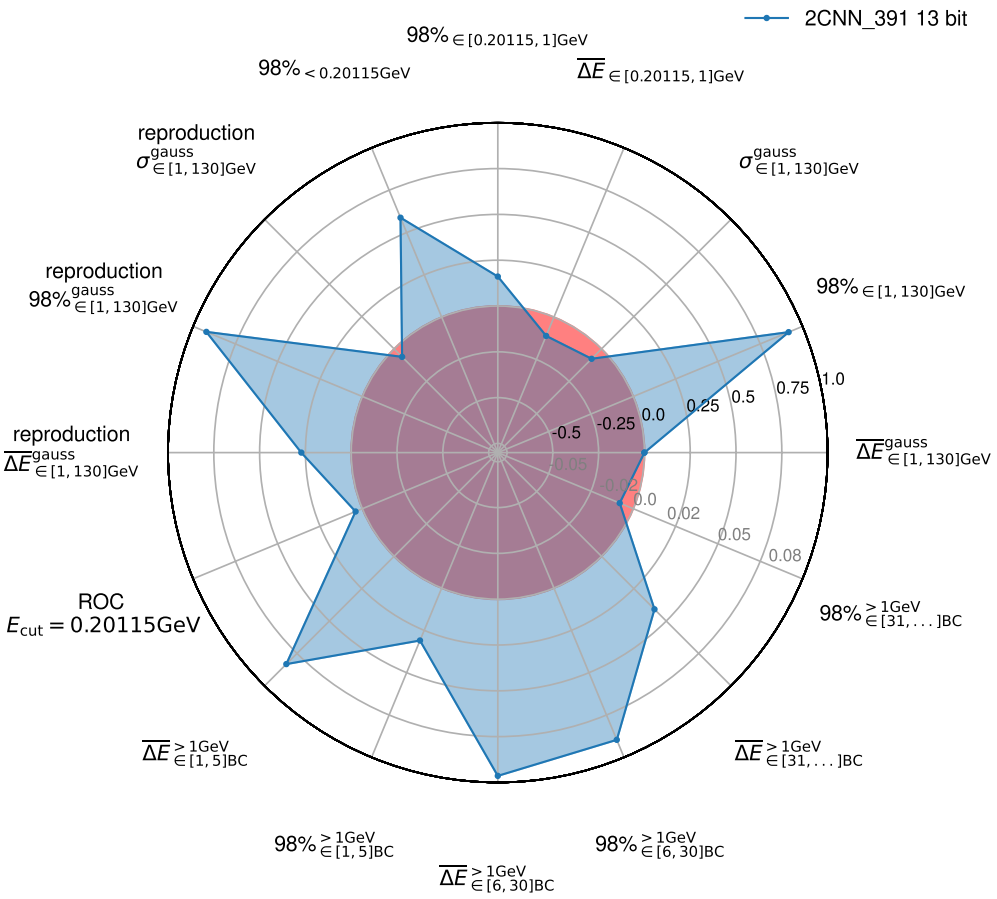

Figure 4.4: Starplot to compare the performance of the 2CNN_391 network, quantized to 13 bits, with the OF.)

5 Summary and Outlook

5.1 Quantization of Convolutional Neural Networks with 100 Parameters

A convolutional neural network limited to 100 parameters, the 2CNN_99, was trained quantization aware with reduced bit width. Training and evaluation were done for the EMB $\eta =$ 0.5125x0.0125 cell with low gain data. This quantization means that the weights and activation functions inside the network used reduced bit widths compared to the 32 bits normally used. This was done to simplify the implementation of FPGA and to reduce the consumption of ALMs and power. The tested network showed a stable performance at 18 bits compared to not quantized networks. Notably, the performance on overlapping hits, which are expected in large quantities at the HL-LHC, did not decrease and showed significant improvements compared to the OF. This is the bit width required for FPGA implementation. It will make implementation easier and will reduce differences between the network run on FPGA and on Keras.

Further quantization to lower bit widths showed that the network kept its performance until 16 bits. For lower bits, its performance started to decrease. This enables the implementation of the network on FPGA with a reduced bit width. Reducing the bits on the FPGA will reduce the consumption of ALMs and power, enabling a more efficient implementation and leaving more space for other software. Using less than 16 bits would also be possible if a trade-off in performance is possible. This might be acceptable since the performance on close hits is still superior compared to the OF.

An estimation of the saved resources on the FPGA is needed in the future. This requires the implementation of quantized networks on the FPGA, which is in development at the time of this thesis. Further analysis is needed for different cells of the LAr calorimeter.

5.2 Optimization of Neural Networks with 400 Parameters

New architectures with up to 400 parameters were tested and evaluated to increase the networks' performance. The architectures were optimized for not quantized networks and trained quantization aware to 18 bits.

The best architectures were a 2CNN with 391 parameters and a 3TCNN with 376 parameters. Both networks were trained quantization aware to 18 bits. Using more layers on the network did not improve the performance further. The 2CNN_391 had overall the best performance. It outperformed the Optimal Filter in the energy reconstruction, saw a large improvement in the reconstruction of close hits and reduced the number of outliers. The 3TCNN saw a similar yet slightly worse performance.

Further quantization showed that the 2CNN had a stable performance until 15 bits. The

resolution and number of outliers remained stable until 13 bits, but the network gained a bias towards overestimation at these bit widths. A bias in the same order also occurs when using the OF. If this is acceptable, going down to 13 bits is also possible. For lower bits, the performance degraded again.

The 3TCNN was stable until 11 bits, with an increased bias starting at 13 bits. This could enable a bigger decrease in the bit width, but this network has the downside of using a Sigmoid function, which is computationally expensive. Linear approximations of Sigmoid were tested but did not perform well. The reason for this is, most likely, that the approximations were not designed to work with quantization. A new approximation for quantization aware training could benefit the network and enable a very efficient implementation. The network was, in general, worse compared to the 2CNN_391. This, together with the Sigmoid functions, makes implementation on the FPGA unlikely.

It was seen that the performances of the networks varied depending on the data set used for evaluation. The 3TCNN_376 performed the best on uniform gap data, particularly on close hits, while the 2CNN had the best performance on random and constant gap data. An explanation for this is that the TCNN architecture is better suited to work on close hits, Such a specialization would have happened during the training, if every architecture is by design better suited for those hits. The difference between both networks wasn't very large, though. Larger networks with more parameters might generalize better between the data. Those networks are currently unrealistic for FPGA implementation without further resource usage optimization. In particular, DSP usage would need to be reduced.

In the future, the networks must be trained on different LAr cells and on high gain data. This needs a decision on how these differences will be treated. It is possible to train one network on all cells, high gain, and low gain data. This requires the network to generalize on all data. A network with 400 parameters might be able to do this, but this would most likely result in a worse performance on all individual cells. Training one neural network for every cell is unrealistic, but creating clusters of cells with similar pulse shapes and training networks on those clusters should be possible. For the gain selection, the increased availability of resources makes it possible to implement two networks on the FPGA, as an alternative to one network for both. One network for low gain selection and one for high gain. This in turn requires a set distribution for the parameters of each network.

Also, quantization needs to be checked on those networks. High gain or different LAr cells might enable different bit widths compared to the tested cell.

All networks in this thesis were optimized with KerasTuner for full 32-bit width. These architectures are, however, not necessarily optimal for working with 18 or less bits. Architectures that are optimized to a specific bit width could perform better and enable quantization to lower bits. Such optimization of quantized networks was not possible with KerasTuner at the time of this thesis. Gridsearches could be used for this, but they take a long time and are

biased in the choice of parameters.

5.3 Comparion of Neural Network Performances

A detailed comparison of the performance of neural networks with the Optimal Filter was done with starplots. These enable a comparison with different performance measurements. Based on these plots, the 2CNN_99 outperforms the OF in most measurements. Especially the performance for close hits is very good compared to the OF. The network performed slightly worse on Gaussian fitted standard deviations for high energies and the 98% range for isolated hits.
When increasing the number of parameters to 400, the 2CNN_391 showed an increase on all parameters and outperformed the OF at the resolution. The 98% range for isolated hits is slightly worse than the OF.
In the future, a more qualitative analysis is required. This needs analysis on which parameters are most important.
There is also need to decide which network will be used. The best-performing network from this thesis is the 2CNN_391. Using full 400 parameters, however, comes with a high the resource consumption. It has to be decided which consumption is acceptable and if a trade-off in performance has to be made.
One open question is if even larger networks can increase the performance further. Such increases could be possible if larger networks can learn to work better on close hits and outliers. Trainings of bigger networks are possible, but the DSPs become the limiting factor on the FPGA at around 500 parameters. Their usage is not influenced by quantization and would, therefore, need optimizations on the software or advanced techniques for the neural network to eliminate weights after training. One of those techniques is pruning. This would remove parameters of the network that are close to 0 and have, therefore, no large influence on the output after training. Doing this could decrease ALM consumption further and also decrease DSP usage. It was not tested for this thesis because weights close to 0 were not observed in the networks, and pruning, in general, is mostly used for larger networks. There is also still need for other software on the FPGA, which makes bigger networks unlikely.

Appendix

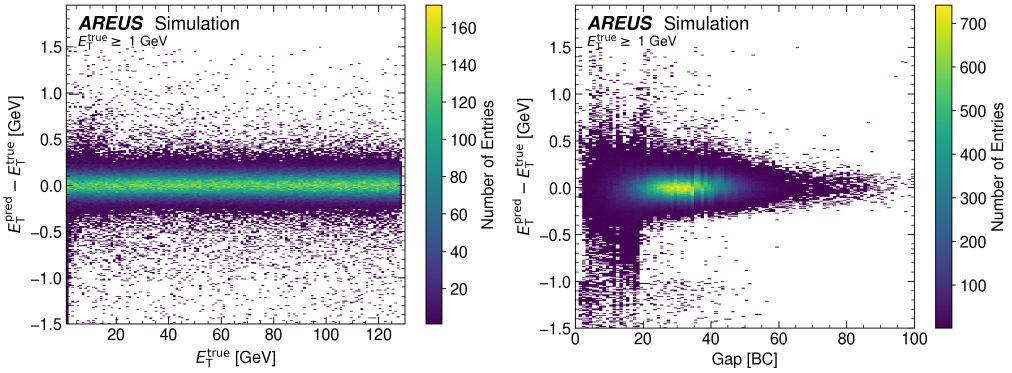

Figure A1: Performance of the 2CNN_235. On the left shown is the distribution of the energiy residuals and on the left the gap performance. This network was not further evaluated, because the gained performance was small compared to the 2CNN_391.

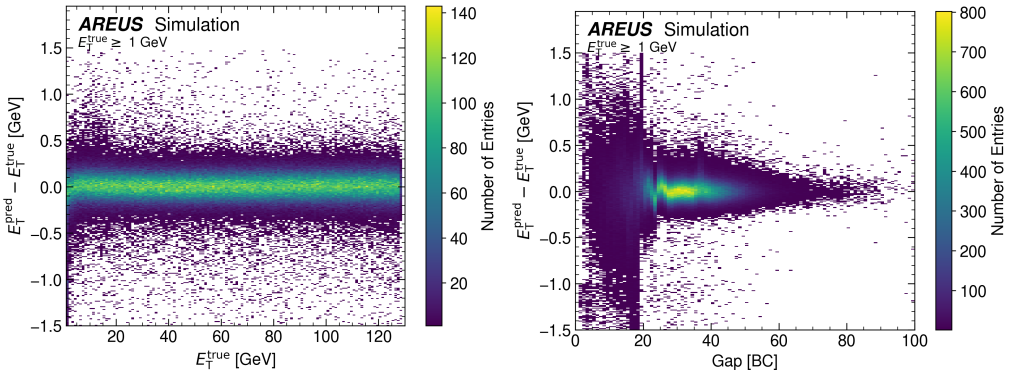

Figure A2: Performance of the 4CNN_384. On the left shown is the distribution of the energy residuals and on the left the gap performance. It performed worse than other CNN architectures and was therefore not further evaluated.

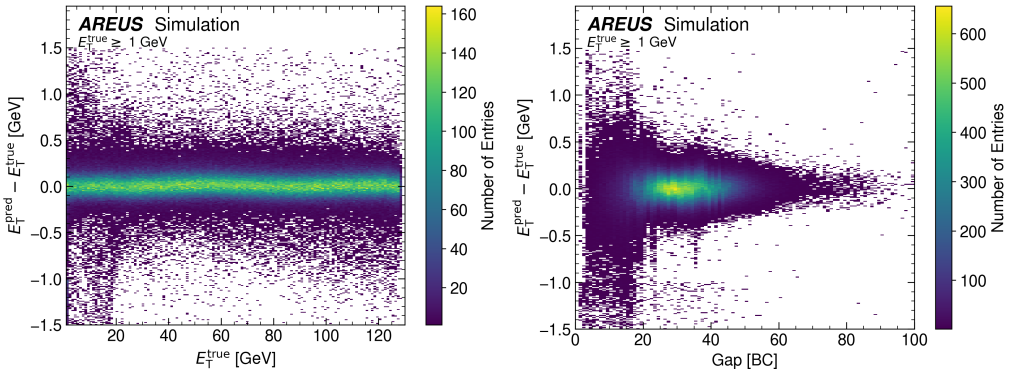

Figure A3: Performance of the 4TCNN_434. On the left shown is the distribution of the energy residuals and on the left the gap performance. This network performed the worst out of all tested and optimized architectures.

List of Figures

Bibliography

[1] W. N. Cottingham and D. A. Greenwood, *An Introduction to the Standard Model of Particle Physics*, 2nd ed. Cambridge University Press, 2007.

[2] T. A. Collaboration, "Observation of a new particle in the search for the standard model higgs boson with the ATLAS detector at the LHC," *Physics Letters B*, vol. 716, no. 1, pp. 1–29, sep 2012. [Online]. Available: https://doi.org/10.1016%2Fj.physletb.2012.08.020

[3] K. R. Labe, "The muon g-2 experiment at fermilab," 2022.

[4] X. V. et al., "Beyond the standard model physics at the hl-lhc and he-lhc," 2019.

[5] "The standard model of particle physcis," https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg, accessed July 25 2023.

[6] L. Roszkowski, E. M. Sessolo, and S. Trojanowski, "WIMP dark matter candidates and searches—current status and future prospects," *Reports on Progress in Physics*, vol. 81, no. 6, p. 066201, may 2018. [Online]. Available: https://doi.org/10.1088%2F1361-6633%2Faab913

[7] O. S. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, and P. Proudlock, *LHC Design Report*, ser. CERN Yellow Reports: Monographs. Geneva: CERN, 2004. [Online]. Available: https://cds.cern.ch/record/782076

[8] T. A. Collaboration, "The atlas experiment at the cern large hadron collider," *Journal of Instrumentation*, vol. 3, no. 08, p. S08003, aug 2008. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/3/08/S08003

[9] T. C. Collaboration, "The cms experiment at the cern lhc," *Journal of Instrumentation*, vol. 3, no. 08, p. S08004, aug 2008. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/3/08/S08004

[10] T. A. Collarboration, "The ALICE experiment at the CERN LHC. A Large Ion Collider Experiment," *JINST*, vol. 3, p. S08002, 2008, also published by CERN Geneva in 2010. [Online]. Available: https://cds.cern.ch/record/1129812

[11] T. L. Collarboration, "The LHCb Detector at the LHC," *JINST*, vol. 3, p. S08005, 2008, also published by CERN Geneva in 2010. [Online]. Available: https://cds.cern.ch/record/1129809

[12] J. Pequenao, "Computer generated image of the whole ATLAS detector," 2008. [Online]. Available: https://cds.cern.ch/record/1095924

[13] *ATLAS inner detector: Technical Design Report, 1*, ser. Technical design report. ATLAS. Geneva: CERN, 1997. [Online]. Available: https://cds.cern.ch/record/331063

[14] *ATLAS liquid-argon calorimeter: Technical Design Report*, ser. Technical design report. ATLAS. Geneva: CERN, 1996. [Online]. Available: https://cds.cern.ch/record/331061

[15] *ATLAS tile calorimeter: Technical Design Report*, ser. Technical design report. ATLAS. Geneva: CERN, 1996. [Online]. Available: https://cds.cern.ch/record/331062

[16] *ATLAS muon spectrometer: Technical Design Report*, ser. Technical design report. ATLAS. Geneva: CERN, 1997. [Online]. Available: https://cds.cern.ch/record/331068

[17] D. South and M. Turcato, "Review of searches for rare processes and physics beyond the standard model at hera," *The European Physical Journal C*, vol. 76, 05 2016.

[18] T. A. Collaboration, "Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System," Tech. Rep., 2013, final version presented to December 2013 LHCC. [Online]. Available: https://cds.cern.ch/record/1602235

[19] D. Oliveira Damazio, "Signal Processing for the ATLAS Liquid Argon Calorimeter : studies and implementation," 2013. [Online]. Available: https://cds.cern.ch/record/1616585

[20] A. Bazan, F. Bellachia, A. Blondel, M. Citterio, M. Gomez, L. Fayard, G. Ionescu, M. Kado, A. Karev, L. Kurchaninov, R. Lafaye, B. Laforge, D. Marra, W. Lampl, S. Laplace, A. Leger, P. Matricon, J.-M. Nappa, and I. Wingerter, "The atlas liquid argon calorimeter read-out system," *Nuclear Science, IEEE Transactions on*, vol. 53, pp. 735 – 740, 07 2006.

[21] T. A. Collaboration, "Atlas data quality operations and performance for 2015–2018 data-taking," *Journal of Instrumentation*, vol. 15, no. 04, p. P04003, apr 2020. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/15/04/P04003

[22] F. Zimmermann, M. Benedikt, M. Capeans Garrido, F. Cerutti, B. Goddard, J. Gutleber, J. M. Jimenez, M. Mangano, V. Mertens, J. A. Osborne, T. Otto, J. Poole, W. Riegler, L. J. Tavian, and D. Tommasini, "HE-LHC: The High-Energy Large Hadron Collider: Future Circular Collider Conceptual Design Report Volume

4. Future Circular Collider," CERN, Geneva, Tech. Rep., 2019. [Online]. Available: https://cds.cern.ch/record/2651305

[23] T. A. Collaboration, *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*, ser. CERN Yellow Reports: Monographs. Geneva: CERN, 2020. [Online]. Available: https://cds.cern.ch/record/2749422

[24] "Hl-lhc planned timeline," https://hilumilhc.web.cern.ch/article/ls3-schedule-change, [accessed July 25 2023].

[25] "Expected pileup values at the HL-LHC," CERN, Geneva, Tech. Rep., 2013, all figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-UPGRADE-PUB-2013-014. [Online]. Available: https://cds.cern.ch/record/1604492

[26] "Convolutional Neural Networks with Event Images for Pileup Mitigation with the ATLAS Detector," CERN, Geneva, Tech. Rep., 2019, all figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2019-028. [Online]. Available: https://cds.cern.ch/record/2684070

[27] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[28] M. B. et al., "Accurate prediction of protein structures and interactions using a three-track neural network," *Science*, vol. 373, no. 6557, pp. 871–876, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.abj8754

[29] J. Yacim and D. Boshoff, "Impact of artificial neural networks training algorithms on accurate prediction of property values," *Journal of Real Estate Research*, vol. 40, pp. 375–418, 11 2018.

[30] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015.

[31] Wikipedia contributors, "Artificial neural network — Wikipedia, the free encyclopedia," 2023, [accessed 6-September-2023]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1170800588

[32] A. e. a. Humaidi, " Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00444-8

[33] "Example of a convoltional layer," https://developer.nvidia.com/discover/convolution, [accessed September 8 2023].

[34] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," 2017.

[35] "Example of loss function for neural networks," https://machinelearningmastery.com/why-training-a-neural-network-is-hard/, [accessed September 8 2023].

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[37] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, 10 1989.

[38] "Overfitting and underfitting," https://towardsdatascience.com/overfitting-vs-underfitting-a-conceptual-explanation-d94ee20ca7f9, [accessed September 7 2023].

[39] G. Aad, A.-S. Berthold, T. P. Calvet, N. Chiedde, E. Fortin, N. Fritzsche, R. G. Hentges, L. A. O. Laatu, E. Monnier, A. Straessner, and J. C. Voigt, "Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters," CERN, Geneva, Tech. Rep., 2021. [Online]. Available: https://cds.cern.ch/record/2775033

[40] M. Khaled, "Enhancing the performance of digital controllers using distributed multi-core/heterogeneous embedded systems," Ph.D. dissertation, 01 2014.

[41] "Intel agilex 7 fpga," https://www.intel.de/content/www/de/de/products/details/fpga/development-kits/agilex/agi027.html, [accessed September 30 2023].

[42] C. N. Coelho, A. Kuusela, S. Li, H. Zhuang, J. Ngadiuba, T. K. Aarrestad, V. Loncar, M. Pierini, A. A. Pol, and S. Summers, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," *Nature Machine Intelligence*, vol. 3, no. 8, pp. 675–686, jun 2021. [Online]. Available: https://doi.org/10.1038%2Fs42256-021-00356-5

[43] M. Nagel, M. Fournarakis, Y. Bondarenko, and T. Blankevoort, "Overcoming oscillations in quantization-aware training," 2022.

[44] P. Horn, "AREUS - a software framework for the ATLAS Readout Electronics Upgrade Simulation," 2018. [Online]. Available: https://cds.cern.ch/record/2314235

[45] G. Aad, T. Calvet, N. Chiedde, R. Faure, E. Fortin, L. Laatu, E. Monnier, and N. Sur, "Firmware implementation of a recurrent neural network for the computation of the energy deposited in the liquid argon calorimeter of the ATLAS experiment," *Journal of Instrumentation*, vol. 18, no. 05, p. P05017, may 2023. [Online]. Available: https://doi.org/10.1088%2F1748-0221%2F18%2F05%2Fp05017

[46] "Kerastuner," https://keras.io/keras_tuner/, [accessed July 25 2023].

[47] J. Yu and K. Spiliopoulos, "Normalization effects on deep neural networks," 2022.

[48] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012. [Online]. Available: http://jmlr.org/papers/v13/bergstra12a.html

[49] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," 2018.

[50] J. Si, S. Harris, and E. Yfantis, "Neural networks on an fpga and hardware-friendly activation functions," *Journal of Computer and Communications*, vol. 08, pp. 251–277, 01 2020.

[51] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 6869–6898, jan 2017.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Kern- und Teilchenphysik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Alexander Lettau
Dresden, November 2023