COBURG UNIVERSITY
of applied sciences and arts

Readout Chip Characterization and Convolutional Autoencoder Trigger Concept for the CMS High Granularity Calorimeter Upgrade

Don Winter
Under supervision of
Prof. Dr. Michael Wick (HSC)
Dr. André David Tinoco Mendes (CERN)

Thesis submitted for the degree
Bachelor of Engineering

February 2021

Contents

1 Introduction

Particle physics is a branch of physics studying the fundamental constituents of matter and its interactions. The Standard Model (SM) [8], which has been developed over decades, represents a widely-accepted theory in this regard. In recent years, the predictions of the SM and other theories of particle physics have been put to test in particle collision experiments with ever increasing precision. One of the most sophisticated experiments is the Compact Muon Solenoid (CMS) detector [27], part of the Large Hadron Collider (LHC) complex [24, 25, 25] at the franco-swiss border near Geneva, Switzerland.

While the LHC delivers particle collisions with center-of-mass energies of up to $\sqrt{s} = 14\,\text{TeV}$, the detector is used for measuring the stable decay products emerging from collisions in order to infer the primary interactions which, in turn, allow for testing theoretical predictions. The latest endeavor to push the limits of statistical uncertainty is the High Luminosity Upgrade [14]. This upgrade will allow to collect more data per run, due to higher beam intensity and therefore more collisions per bunch crossing.

In order to cope with the increased amount of collisions, the calorimeter endcaps of the CMS detector need to be replaced along with other parts of CMS. The new endcaps must incorporate improved spatial resolution and be able to withstand higher doses of radiation of up to $10^{16}\,\text{neq/cm}$. The High Granularity Calorimeter [18] (HGCAL, Ch. 3) is being designed to satisfy these requirements. It is made of more than $600\,\text{m}^2$ of silicon sensors in the regions of highest irradiation that are operated by a new type of readout chip [59] which integrates several state-of-art technologies in order to achieve the desired dynamic range and time resolution. The contributions of this work to the HGCAL upgrade can be stated as follows:

To verify that each chip behaves as planned, several chip parameters need to be determined along with their uncertainty. Once determined, the production variation between chips can be mitigated by subsequent fine-tuning of these parameters. Ch. 4 introduces the stategy for testing chips together with the hardware and software components that were developed and used for a prototype version of the chip. Only chips that perform as needed will be assembled onto printed circuit boards and subsequently used on module prototypes or installed in the final calorimeter.

An open issue that has to be solved before commissioning the calorimeter is the varying performance of the trigger primitive generator (TPG) [52] with type of incident particle. The TPG is part of the trigger system which has the task to reduce the event data rate by selecting only events relevant to the CMS physics program. Recent developments enabled the implementation of hardware-based neural networks in the TPG chain. In theory, a TPG concept based on neural networks could deliver better performance than the current TPG implementation, due to its higher level of generalization, within the required latency and bandwidth. In Ch. 5 such a concept is introduced by assuming real detector constraints and a surrogate dataset. The chapter closes with a discussion about the feasibility of the concept and compares it to systems currently in place.

In Ch. 6, the conclusions of this work are summarized and an outlook on the still ongoing activities is given.

2 Particle physics at CERN

In this section the Standard Model (SM), the Large Hadron Collider (LHC) and the Compact Muon Solenoid (CMS) experiment are introduced as the foundations for doing particle physics at the European Laboratory for Particle Physics (CERN). Each of these topics is the product of decades of research and development undertaken by a myriad of researchers and engineers from all around the world. The following paragraphs, inspired by the presentation in Ref. [49], discuss them briefly.

2.1 Standard Model

Particle physics deals with the formulation and validation of rules for matter particles and the interaction between them. The Standard Model of particle physics (SM) represents our current understanding of these rules compiled into a theoretical framework. The SM is a consistent, finite theory that allows predictions to be made of fundamental particle interactions that have been experimentally verified with very high precision. It introduces the elementary matter particles quarks and leptons, together with their corresponding antiparticles and three elementary interactions between them acting on subatomic scales. The fourth known force, gravity, has not yet been reconciled with the SM. However, if one compares the magnitudes of all these forces gravity is by far the weakest and its influence on matter particles on microscopic scales can be considered negligible.

The Standard Model is a quantum field theory based on special relativity, quantum mechanics, and classical field theory. It describes electromagnetic, weak, and strong interactions between elementary particles as couplings of quantized fields that span the entire universe and in which the observable particles manifest as field excitations. As seen in Fig. 1 the SM divides the fundamental particles in four groups. Leptons and quarks are the matter particles, called fermions. Each matter particle has a corresponding antiparticle (not shown in Fig. 1) with opposite quantum numbers. Quarks have a fractional charge compared to the electron and are subject to strong, weak and electromagnetic interactions while leptons have integer charge and only interact weakly and electromagnetically. The exceptions are the leptonic neutrinos that are electrically neutral and only interact via the weak force. The understood universe is comprised of protons, neutrons and electrons which are, except the electron, composite particles made up of three quarks each. Particles that are made of quarks are called hadrons. Beside many more classes of composite particles, only hadrons are of importance for the following chapters.
The bosons are the force-mediating particles by which the matter particles interact. Similar to the fermions the bosons are described as excitations of their respective fields. As result of a symmetry property called gauge invariance in the mathematical formalism of the SM, the boson fields are referred to as gauge fields and the bosons called (vector) gauge bosons. The Higgs boson on the other hand is a scalar boson that does not mediate any force. It describes a quantized excitation in the Higgs field by which fundamental particles acquire their masses via the Brout-Englert-Higgs mechanism [34, 37]. Albeit its theoretical and experimental success, the Standard Model leaves many questions open which physicists try to answer by building particle accelerators to collide particles in order to investigate their interactions and collision products.

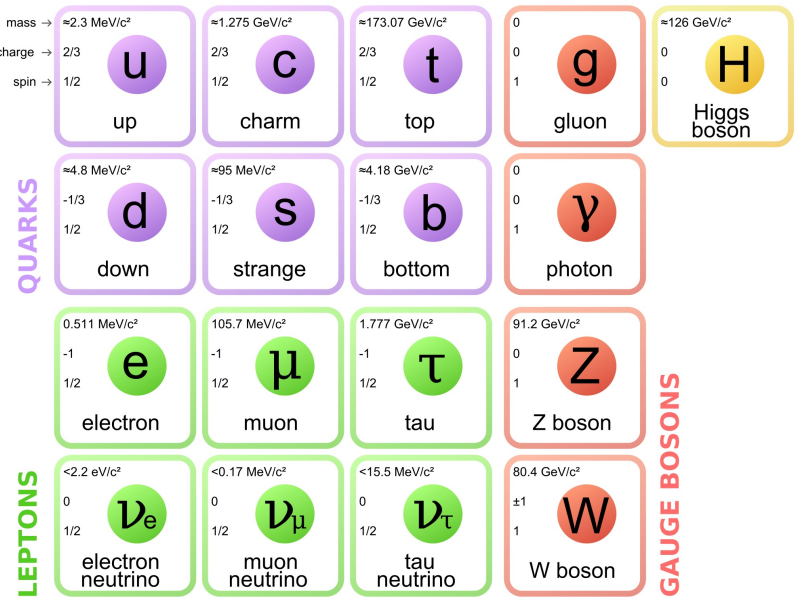


Figure 1: Elementary particles in the Standard Model. Antiparticles are not shown. Figure reproduced from Ref. [8].

2.2 Large Hadron Collider

The Large Hadron Collider (LHC) [21, 24, 25] is the largest and most powerful particle accelerator, storage ring, and collider ever built. It provides proton-proton[1] collisions at energies up to 14 TeV at four interaction points to study different aspects of the Standard Model and beyond.

The LHC is located at CERN on the Franco-Swiss border near Switzerland, Geneva in a circular tunnel 45 m to 170 m underground that was formerly used for the Large Electron-Positron Collider (LEP) [16]. The accelerator measures 26.7 km in circumference in which hadrons, which are composed of two or more quarks, are accelerated to nearly the speed of light in two parallel, evacuated beam pipes until they are brought to collision at one of the four collision points. In order to reach high collision energies the hadrons are pre-accelerated in a chain of four preceding beam facilities: LINAC 4, Proton Synchrotron Booster, Proton Synchrotron and Super Proton Synchrotron (Fig. 2), to reach proton energies of 160 MeV, 1.4 GeV, 25 GeV and 450 GeV, respectively.

Eventually, two beams are injected into opposite directions in the two beam pipes of the LHC. Inside the LHC the particles undergo further acceleration with each orbit via resonant waves of electromagnetic fields, generated in eight radio-frequency cavities. Furthermore, two beams are separated in so-called bunches (this happens already in the Super Proton Synchrotron). In the case of protons, a particle bunch consists of 1.2×10^{11} protons that are accelerated to a peak energy of 7 TeV and collided with another bunch at a center-of-mass energy of 14 TeV. The collision of particle bunches is referred to as bunch crossing (BX) which happens during operation in the LHC at a frequency of up to 40 MHz (every 25 ns) [5].

[1] The LHC also collides ions, which will not be considered in this thesis.

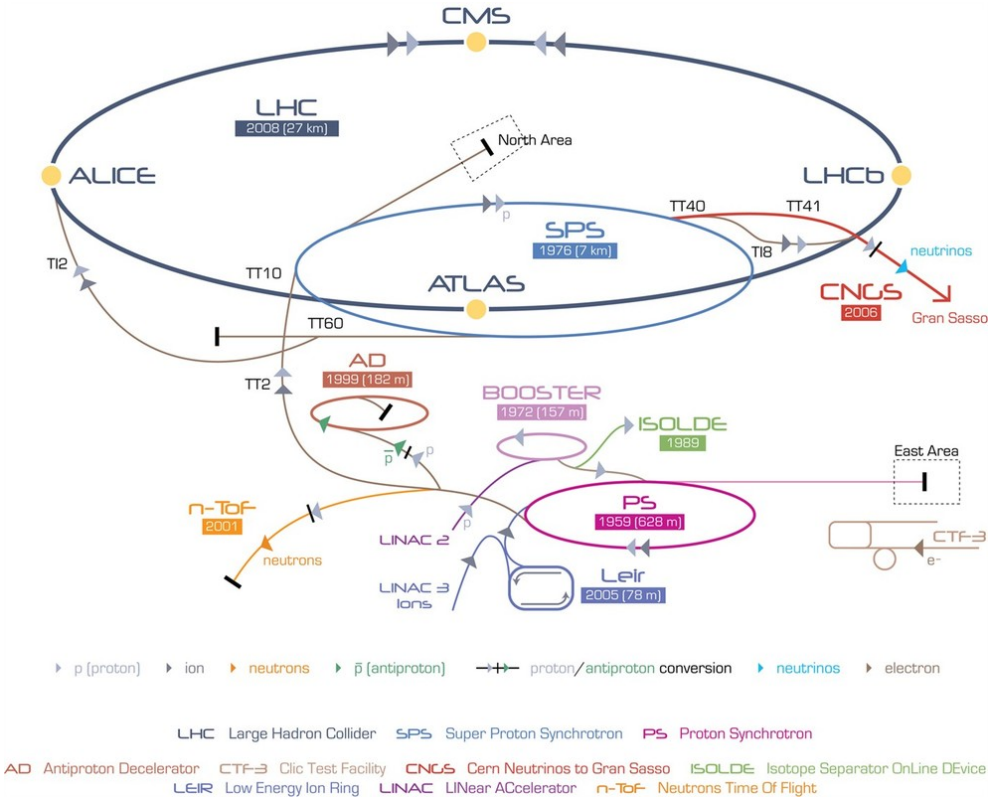


Figure 2: The CERN accelerator complex [5].

In order to bend the particles' motions inside the beam pipes into a circular trajectory, 1232 dipole magnets are used. Since the required magnetic field strength of up to 8.3 T can only be achieved by superconduction the dipole magnets are made of NbTi operated at a temperature of 1.9 K. Further multipole magnets are used to ensure that the beam stays focused despite the electromagnetic repulsion that pushes the charged particles apart. At the interaction points of the ATLAS and CMS experiments the beam is confined to a diameter of about 10 µm before collision [5].

All quantities related to the collider configuration can be summarized into a single quantity, the instantaneous luminosity L, that allows to calculate the interaction rate of a certain process with cross section σ as

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

$$\text{with} \quad L = \frac{N_b^2 n_b f_{\text{orb}} \gamma_\gamma F}{4\pi \epsilon_n \beta^*}, \tag{2}$$

with number of particles per bunch N_b, number of bunches per beam n_b, orbit frequency f_{orb} (around LHC circumference), relativistic gamma factor $\gamma_\gamma = E/m$, geometric reduction factor F due to inclined beam profiles, the normalized transverse beam emittance ϵ_n and the beta factor β^* at the collision point. The LHC was designed to provide an instantaneous luminosity of $\sim 10^{34}\,\text{cm}^{-2}\,\text{s}^{-1}$. The integrated luminosity $\mathcal{L}$ can be used to calculate the expected yield for a process with given cross section σ given all data the

LHC collected in a given time:

$$N = \int \frac{dN}{dt}\, dt = \sigma \cdot \int L\, dt = \sigma \cdot \mathcal{L}. \tag{3}$$

Whereas the instantaneous nominal peak luminosity in the last LHC run (Run 2) in 2016–2018 reached $\sim 2 \times 10^{34}\,\mathrm{cm^{-2}s^{-1}}$ [1], this value will be raised to $5 \times 10^{34}\,\mathrm{cm^{-2}s^{-1}}$ and possibly to $7.5 \times 10^{34}\,\mathrm{cm^{-2}s^{-1}}$ [12] with the implementation of the High Luminosity Upgrade [14] in 2025/26 (Fig. 3). While this implies the collection of more events and thereby more significant statistics it also poses major challenges on the accelerator and detectors located at the interaction points. Some of the challenges posed by the High Luminosity Upgrade on one particular detector, the Compact Muon Solenoid, are addressed in this thesis.

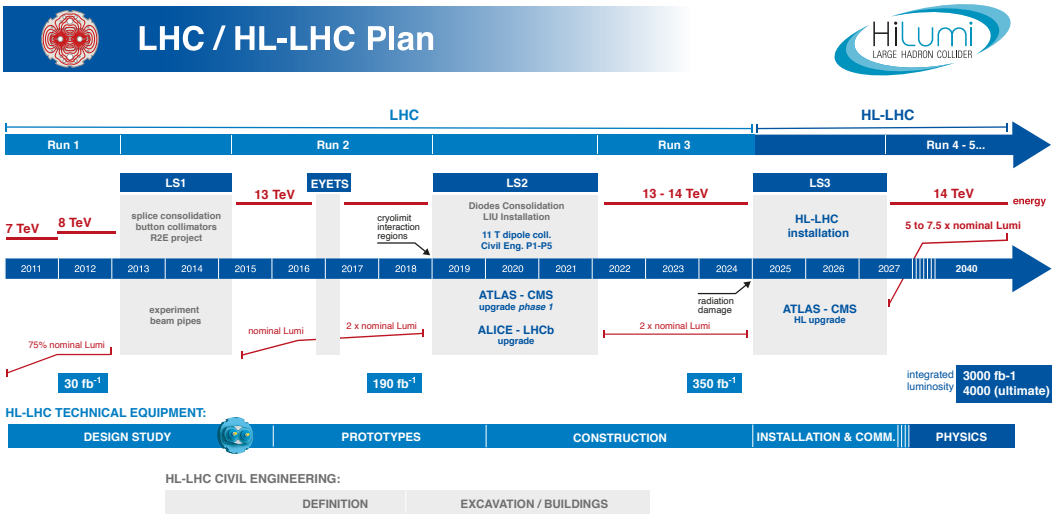


Figure 3: Future plans for the LHC [3].

2.3 Compact Muon Solenoid Experiment

The Compact Muon Solenoid (CMS) [29, 30] is a large multipurpose particle detector for investigating collisions at very high energies and instantaneous luminosities. The CMS detector is located about 100 m under ground near the village of Cessy in France at LHC interaction point five. It has a length of 21.6 m, a diameter of 14.6 m and a mass of about 14 000 t.

The detector can be thought of as a large high-speed video camera [48] taking three-dimensional high resolution ($\sim 100 \times 10^6$ pixels) pictures at a frame rate of 40 MHz (one picture per BX) but instead of light intensities the pixels show energy deposits caused by particles created in the collision events.

Except for neutrinos, the CMS detector is designed to measure all stable particles[2] produced in collision events [49]. Therefore, and because of its accuracy in energy and momentum measurements as well as its good spatial resolution, the CMS detector allows for precise reconstruction of intermediate, unstable particles, making it a versatile detector useful in the analysis of a wide range of collision end products.

Measured objects in the CMS detector are described by the coordinate system shown in Fig. 4. The origin is placed at the collision point, while the the x-axis points towards the center of the LHC and the y-axis points in the upwards direction. Thereby, the x- and y-axis span the transverse plane in which directions are defined by the azimuthal angle ϕ which is zero in the direction of the x-axis. The z-axis is perpendicular to the transverse plane and parallel to the beam, pointing in counter-clockwise longitudinal direction. The polar angle θ is often encapsulated in the pseudorapidity η as

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] = \frac{1}{2}\ln\left(\frac{\|\vec{p}\| + p_z}{\|\vec{p}\| - p_z}\right) = \operatorname{arctanh}\left(\frac{p_z}{\|\vec{p}\|}\right), \tag{4}$$

with $\vec{p}$ being the three-momentum and p_z being its z-component, such that $\eta = 0$ for $z = 0$ and $\eta = \infty$ along the beam pipe on the z-axis. Pseudorapidity is the preferred quantity for measuring longitudinal directions since differences in pseudorapidity $\Delta\eta$ are Lorentz-invariant and do not depend on longitudinal boosts of certain particles.

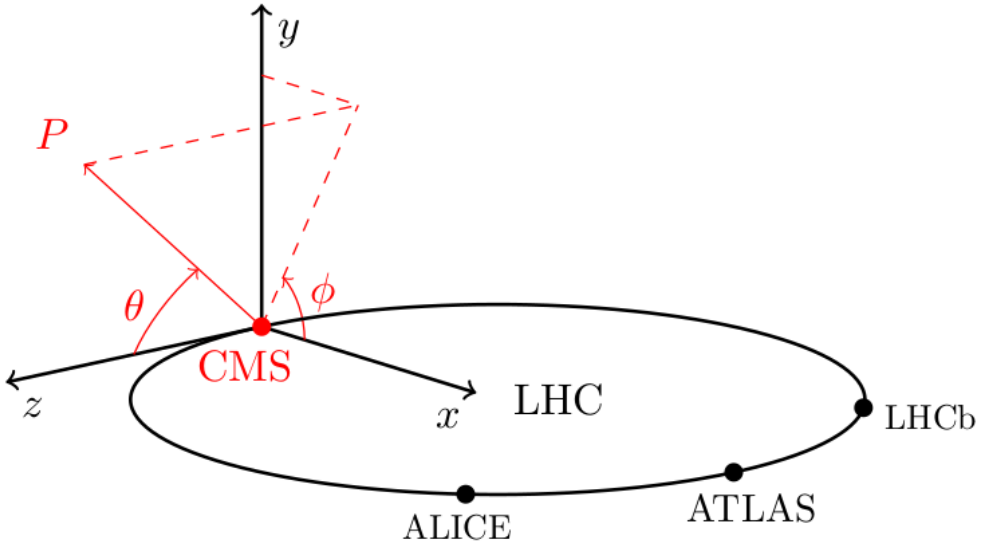


Figure 4: Coordinate system used by the CMS experiment. The x-axis points to the center of the LHC, the y-axis points upwards, and the z-axis points in counter-clockwise direction along the beam [49].

In order to measure the type and kinematic properties of particles, the CMS detector includes several sub-detectors arranged cylindrically around the beam pipe and symmetrical to the interaction point. Measurement quantities of interest are the vectorial

[2]Stable refers to relativistic particles that do not decay while traversing the detector.

three-momentum $\vec{p}$, energy E, the sign of charge, and the origin (interaction vertex) of the particle. The latter becomes important to distinguish between particles traversing the detector at the same time. In 2018, at peak luminosity, CMS registered an average of 40 simultaneous events caused by inelastic proton-proton collisions in every bunch crossing. Because these events lead to additional occupancy of detector cells, they are referred to as pile-up. Determining the interaction vertex of an event allows for a better allocation of the detector signals despite large amounts of pile-up.

The CMS detector allows to measure protons, neutrons, kaons, pions, electrons, photons and muons. Neutrinos can be indirectly detected by inference from missing transverse momentum [50]. In the following, the sub-detector systems of CMS, seen in Fig. 5, are discussed.

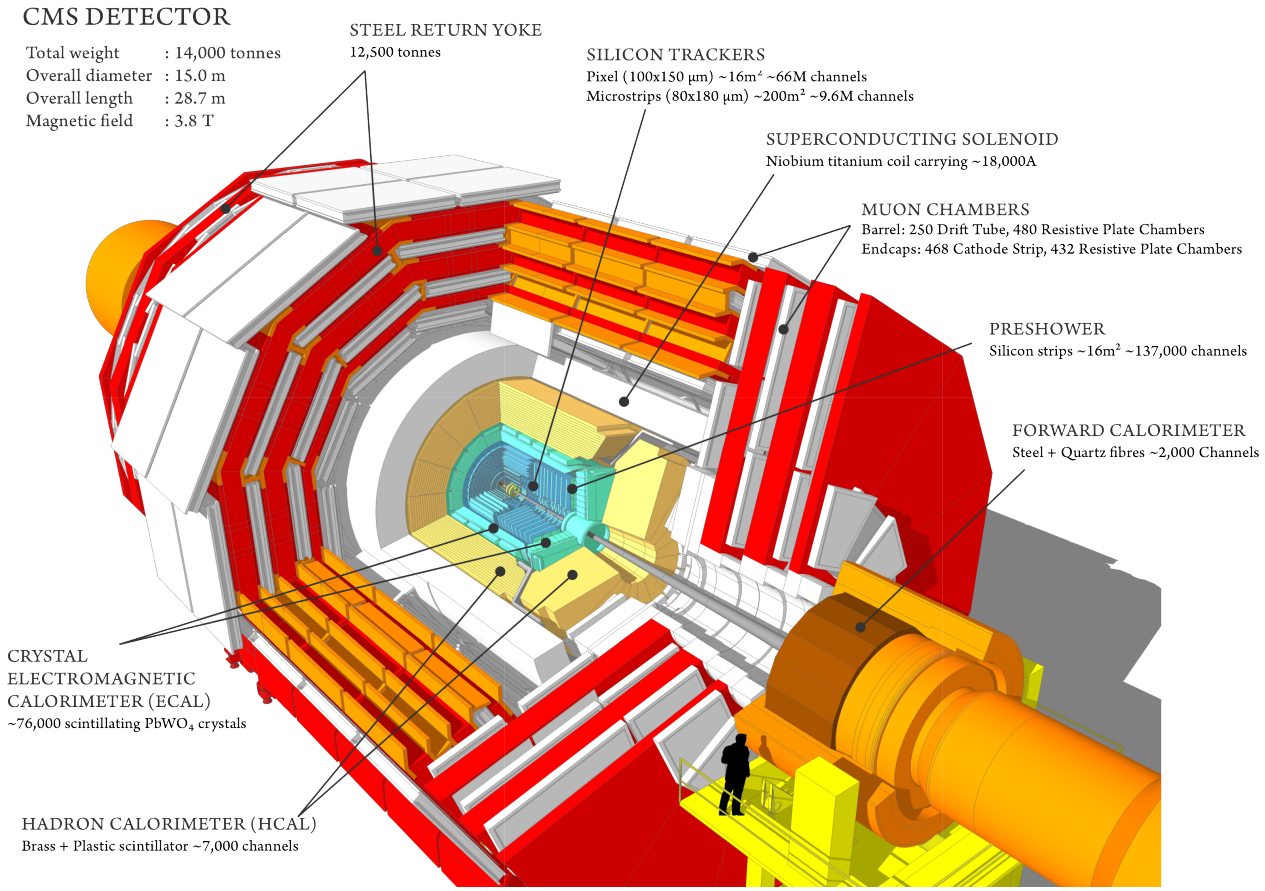


Figure 5: The CMS detector [33].

Tracking System

The inner tracking system consists of two components: Concentric silicon-pixel layers and silicon-strip detectors in close proximity ($\sim$4 cm) to the interaction point. The aim of the tracking system is to identify charged particles and measure their trajectories, momenta, and charge signs. The detection principle is based on silicon semiconductors operated with reverse-bias voltage so that traversing charged particles cause a measurable signal in the depletion region by ionization, which is referred to as a tracker hit. Tracker hits can

be combined using advanced pattern recognition algorithms to build particle tracks [54]. The tracking system was designed with the expected particle flux at various distances from the interaction point in mind to achieve the best possible spatial resolution. In total, the tracking system provides more than 100 million readout channels [30, 53].

Calorimeter System

The calorimeter system includes the electromagnetic calorimeter (ECAL) and the hadronic calorimeter (HCAL) to measure energy and direction of traversing particles. ECAL comprises barrel ECAL (EB), endcap ECAL (EE) and endcap preshower (ES); HCAL comprises barrel HCAL (HB), endcap HCAL (HE) and forward HCAL (HF). While the ECAL detects particles that interact electromagnetically, mostly electrons, positrons and photons, the HCAL is designed to detect charged and neutral hadrons, mostly protons, neutrons, charged pions and kaons. Particles except for muons and neutrinos are fully absorbed inside the calorimeters. The ECAL is a homogeneous calorimeter that includes $\sim$70 000 lead tungstate crystals ($PbWO_4$) that produce scintillation light when excited by a traversing charged particle. The intensity of the scintillation light is proportional to the energy of the incident particle. HCAL on the other hand is a sampling calorimeter that consists of alternating layers of non-magnetic brass absorber material and plastic scintillators [29, 30]. Strongly interacting particles not contained in the ECAL will be stopped by the dense brass absorber layers in which the traversing particle loses energy by creating hadronic showers which excite the subsequent scintillators that in turn produce detectable light. This light is guided via embedded wavelength-shifting fibers via $\sim$7000 channels to the readout system [28]. Additionally, photon-sensitive preshower detectors are placed in front of the endcap ECAL to aid in particle identification for rejection of background processes.

Solenoid Magnet

The large solenoid magnet encapsulates both the calorimeter system and tracking system in which it induces a strong magnetic field that is returned by three outer layers of return yoke which are interspersed with the muon detectors. The target of the solenoid is to introduce a strong enough field such that charged particles are forced to bend in a certain direction while traversing the tracking and calorimeter systems which allows to identify charge and measure momentum [29, 30]. The solenoid magnet is composed of five 2.5 m-long modules, which are assembled into a hollow cylinder of length $L = 12.5\,\mathrm{m}$ and diameter of 6.3 m. Each module consists of an aluminium shell with four internal layers of winding, each with 109 turns, such that all modules together have $N = 5 \cdot 4 \cdot 109 = 2180$ turns, which are operated in a vacuum with a current of up to $I = 18\,000\,\mathrm{A}$, thereby creating a magnetic field of $B = \mu_0 NI/L \approx 4\,\mathrm{T}$ [6]. The large magnetic field strength is achieved by superconducting NbTi cables as coil material that are cooled down to 4.5 K. The solenoid magnet is the largest of its type ever built [27].

Muon Detectors

The outermost sub-detector forms the muon detection system, interspersed by the iron return yoke of the solenoid magnet. Ideally, the muon detectors are only traversed by muons and neutrinos, since all other particles should have been fully absorbed inside the calorimeters. Hits detected by the muon system are combined with the particle tracks

recorded by the tracking system in order to identify muons and measure their momenta and charges. The muon detectors are gaseous detectors, producing electron avalanches through ionization of the gas molecules when traversed by a charged particle causing measurable signals. Like the calorimeters, the muon detectors are split in two sections for the barrel region and the two endcaps that use two different detection techniques. Both barrel and endcaps additionally include resistive plate chambers with very fast response times for trigger purposes.

Trigger System

At a levelled instantaneous luminosity L=$5 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ and 140 pile-up events in the high luminosity environment as well as a bunch crossing rate of 40 MHz, the LHC produces 5.6×10^9 proton-proton collision events every second at the interaction point of the CMS experiment [29, 30]. Considering all event information provided by the detector one bunch crossing yields ~ 4.5 MB of event data which would lead to 252 TB event data per second if it were all stored on disk. The data acquisition (DAQ) system has neither readout bandwidth nor storage capability to handle all this data - the peak storage rate of the DAQ system is 27 GB/s [12]. The trigger system was designed to filter selectively (trigger on) events at a rate of 40 MHz based on a realtime analysis using coarse-grained event data and thereby reducing the storage rate of events. Only after the trigger system accepts an event is it written to permanent storage. The realtime analysis filters those events that exhibit signatures of processes that are interesting for the CMS physics program [49]. In order to record unknown processes the triggering thresholds can be set sufficiently low. The CMS Trigger is divided in two stages The Level-1 trigger (L1T) and the high-level trigger (HLT). The L1T is a high bandwidth, fixed latency system based on ASICs (Application Specific Integrated Circuits) and FPGAs (Field Programmable Gate Arrays) located on-detector and in a neighboring underground cavern respectively, while the software-based HLT runs on a computing cluster of 3000 compute units [52].

Recorded detector data can be held in internal buffers of the on-detector readout systems for up to 3.2 µs. During this time, coarse-grained data from calorimeter and muon systems is processed by the L1T until a trigger decision is issued. The buffers account for the latency of the trigger system, however a decision is issued every 25 ns with each bunch crossing. During this process the L1T constructs so called *trigger primitives* which are 3-dimensional energy clusters representing muons, electrons, photons, jets, or global sums of transverse or missing transverse energy. Events that the L1T accepts are further redirected to the HLT. The HLT includes tracker data, tries to reconstruct the Level-1 trigger decision and filters the events by more strict constraints. In total, the L1T reduces the storage data rate from 40 MHz to 100(500) kHz by a factor of 400 (80) and the HLT further reduces it from 100 (500) kHz to 0.4 (5) kHz by factor 250 (100) in the Phase-1 (Phase-2) upgrade of the trigger system [12, 50]. The generation of trigger primitives for a specific sub-detector will be discussed further in the next chapter.

3 High Granularity Calorimeter

The increase in instantaneous peak luminosity due to the High Luminosity Upgrade imposes two major challenges on the calorimeter systems [19]:

1. The increased amount of hard proton-proton collisions at the interaction point generate more shower-inducing particles and thereby more showers inside the calorimeter which leads to a higher detector occupancy, i.e. pile-up. Larger pile-up makes it harder to distinguish between showers in order to identify its inducing particles. In the high luminosity environment an amount of ~ 140 pile-up events per bunch crossing is expected which is more than a 3-fold increase to the prospective value in Run 3 [63] (2022–2025). In Fig. 6a the effect of pile-up (PU) on the energy resolution for the particular example of $H \to \gamma\gamma$ can be seen. The energy resolution $\sigma_{\text{eff}}(E)/E$ deteriorates (increases) especially in the endcap region ($\eta > 1.5$) with the amount of pile-up and integrated luminosity. Energy resolution indicates how precisely the incident particle's energy can be inferred given the calorimeter signal.

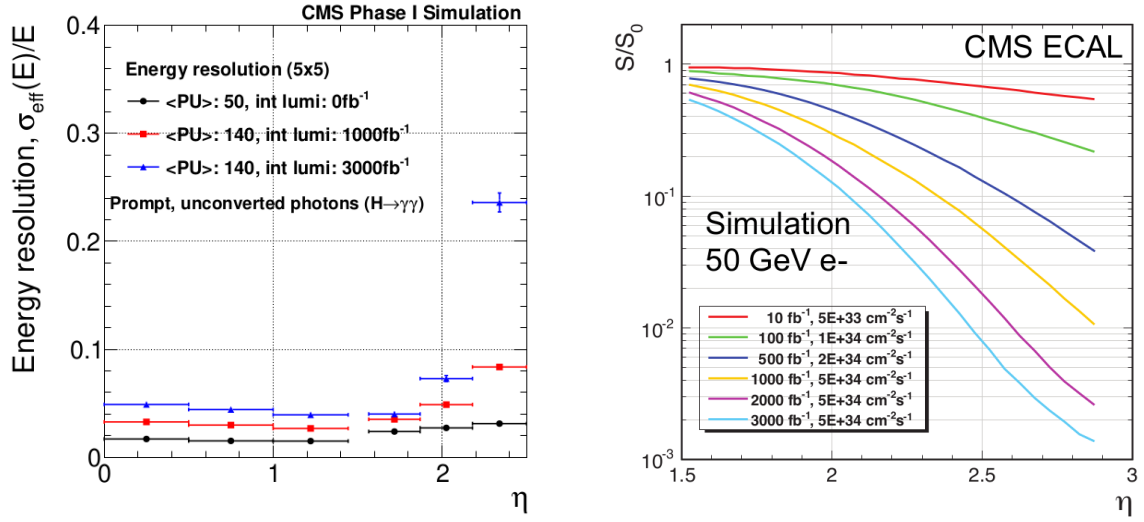
2. Due to larger numbers of proton-proton collisions, the irradiation level increases which causes severe damage to sensitive detector components. The $PbWO_4$ crystals of the current ECAL endcap for example will not be able to cope with the expected fluence of $\sim 10^{16}$ 1 MeV neutron-equivalent, which exceeds their design values by over one order of magnitude [13,17,64]. The radiation damage causes less light to be collected as the crystals get progressively darker, as seen in Fig. 6b that ultimately leads to poorer signal quality, i.e. a lower signal-to-noise ratio. In Fig. 6b it can also be seen that the effects of radiation damage are particularly concerning in the endcap region of $\eta > 1.5$.

These two reasons motivated the redesign of the CMS endcap calorimeter systems which are currently under development. The new endcaps will tackle the challenges of pile-up and radiation damage by using a highly-granular detector geometry and radiation-hard silicon sensors. Due to cell pitches of down to 0.5 cm being the prominent feature of the devices the new endcap calorimeters are referred to as High Granularity Calorimeters (HGCAL).

HGCAL is designed as a sampling calorimeter, consisting of 50 layers of active material interspersed with absorber material. Each endcap is divided into two sections with 28 layers in the electromagnetic section (CE-E) and 22 layers in the subsequent hadronic section (CE-H). In the CE-E, copper, copper-tungsten, and lead are used as absorber, and silicon as active material. The CE-H uses steel as absorber material and silicon sensors as well as scintillator tiles as active materials for high and low radiation regions, respectively, which can be seen in Fig. 7a.

The active layers are composed of a total of 30 000 sensors (28 056 hexagonally-shaped 8-inch silicon sensors and 3744 scintillator tileboards) which are attached to printed circuit boards (PCBs) that together mount about 100 000 (95 826 silicon and 4320 scintillator) readout chips (ROCs). Each ROC is connected to a maximum[3] of 72 sensor cell [15, 40]. In total there will be ∼6 million silicon readout channels for both endcaps. The high granularity becomes apparent by comparing this amount to the combined ∼14 000 readout

[3] In practice some channels are not connected to sensor cells.



(a) (b)

Figure 6: (a) Energy resolution $\sigma_{\text{eff}}(E)/E$ for photons from Higgs boson decay for different integrated luminosities and pile-up as a function of pseudorapidity η. (b) Expectation of relative light collected S/S_0 of ECAL crystals for 50 GeV electron showers as function of pseudorapidity η for various aging conditions with S_0 being the light collected of unaged crystals. In 2016–2018 (Run 2) the integrated luminosity amounted to 190 fb^{-1} whereas for High Luminosity an amount of 3000 fb^{-1} to 4000 fb^{-1} is expected [31].

channels of the endcap electromagnetic calorimeters of the current CMS detector [56]. When fully assembled, each endcap will weigh ∼230 t with a conic shape of ∼2 m in length and ∼2.3 m radius at its larger end. Each calorimeter will cover about $1.5 < \eta < 3$ of incident particles.

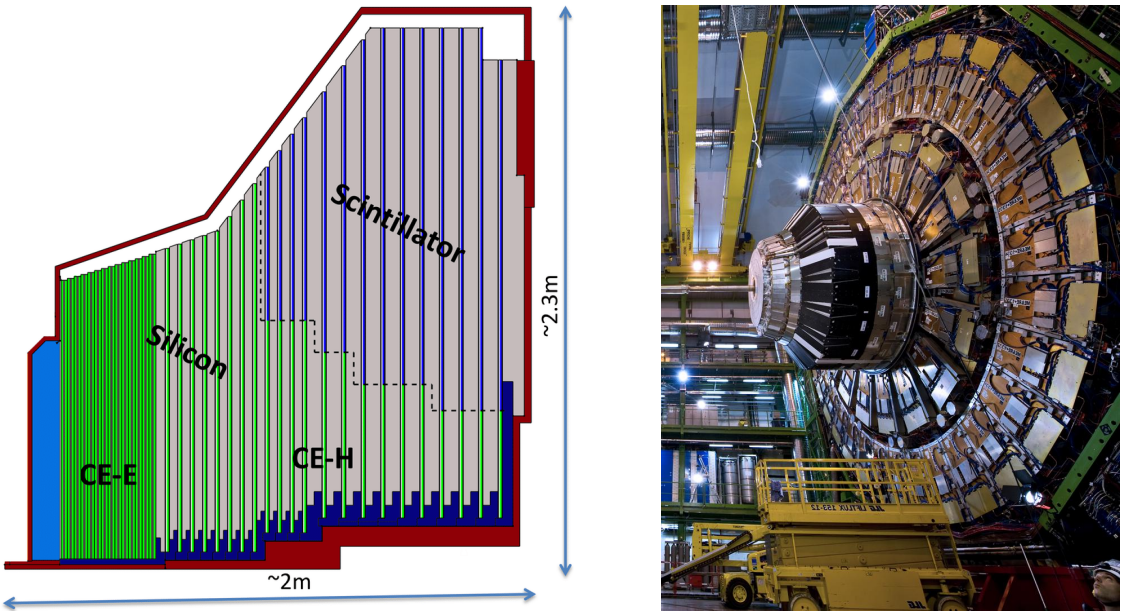
HGCAL is commonly referred to as a 5-D calorimeter because it enables the measurement of energy and time as a function of the position due to the integration of three defining technologies:

- Large dynamic energy range by ADC and time-over-threshold (TOT) method.
- Precise time of arrival measurements with timing resolution of ∼25 ps.
- Precise tracking of shower developments due to high granularity.

3.1 Calorimetry

Calorimetry is a detection principle in particle physics to measure four-momenta and energy flow of particles [61]. Calorimeter data contributes to the data analysis mainly by identification of incident particles (mostly electrons, positrons, photons and hadrons) and by measuring the energy these particles deposit when interacting with matter.

While the incident particles are usually at energies in the GeV range, the performance of a calorimeter is determined by processes happening at the MeV, keV, and sometimes also eV level. A particle traversing a calorimeter induces a shower of secondary particles with



(a) (b)

Figure 7: (a) Schematic cross section of an endcap [47]. (b) Current calorimeter endcap inside the CMS detector [22].

decreasing energies which eventually can be measured. Showers initiated by hadrons[4] are more complex and distinctly different from ones initiated by electrons or photons [61]. Since the focus of this thesis is solely on electromagnetic showers, these will be discussed in more detail.

Electromagnetic showers develop via a few well-understood processes:

- Electrons and positrons lose energy by ionization at low energies and by bremsstrahlung at high energies. If the energy is at a critical point ϵ_{crit} both of these processes play an equally important role. Since bremsstrahlung transforms one particle into two ($e^{\pm} \rightarrow e^{\pm}\gamma$) while ionization involves merely an energy transfer of one particle to another, the critical energy marks a turning point in shower development, namely the point where the average energy of shower-carrying secondary particles is at its maximum followed by shower decay. The critical energy is roughly inversely proportional to the atomic number Z of the calorimeter material used for particle absorption,

$$\epsilon_{crit} = \frac{610\,\text{MeV}}{Z + 1.24} \tag{5}$$

  The fractional energy loss of both these processes in the high-Z material lead (Pb) can be seen in Fig. 8a.

- Photons interact through the photoelectric effect at low energies, electron-positron pair production at high energies and via Compton scattering in between. As for electrons and positrons, only pair production contributes to the shower growth by turning one particle into two ($\gamma \rightarrow e^{+}e^{-}$). In Fig. 8b the cross sections of these processes can be seen as a function of the respective photon energy in the high-Z

[4]Such as protons and pions.

material lead. Figure 9 shows the likelihood of these processes as a function of the atomic number Z. In low-Z materials, Compton scattering is by far the most dominant process in a large energy range. As an example for the energy- and material-dependence of the processes' cross sections the one for photoelectric effect $\sigma_{p.e.}$ scales by $\sim Z^5 E^{-3}$, thereby is highly material-dependent. Furthermore, the angular distribution of e^+e^- pairs is highly directional while photo- and Compton-electrons scatter isotropically. Therefore, electromagnetic showers begin concisely at high energies and widen as the shower particles' energies decrease [61].

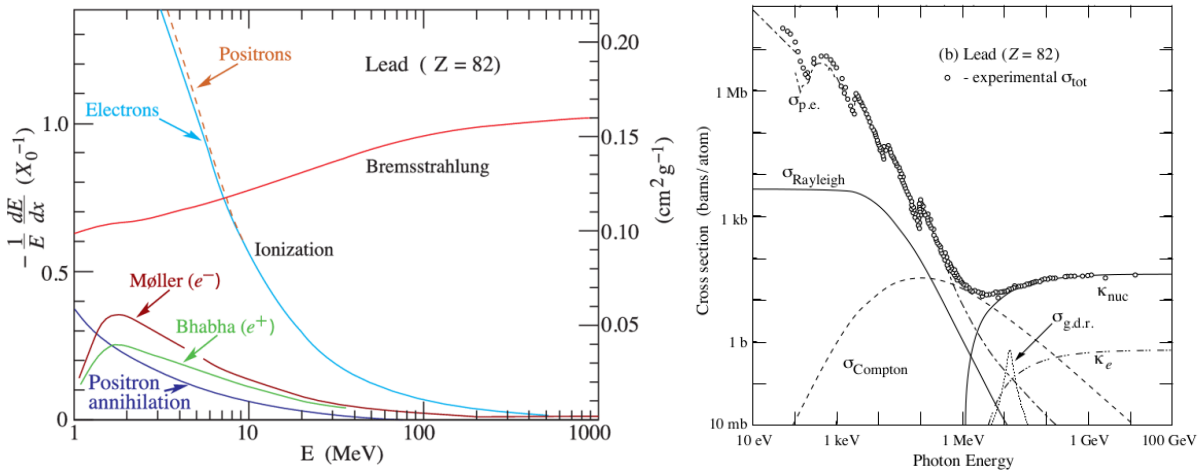


Figure 8: (a) Fractional energy loss in lead by electrons and positrons as function of the energy. (b) Cross sections in lead by photons as function of energy. The shown processes are cross sections for pair production in electron field (κ_e), pair production in nuclear field (κ_{nuc}), photoelectric effect ($\sigma_{p.e.}$), Rayleigh scattering ($\sigma_{Rayleigh}$), and photo-nuclear absorption ($\sigma_{g.d.r}$). Figures reproduced from Ref. [57].

Showers start developing in the absorber material for incident particles in the GeV range and higher. The longitudinal shower development can be characterized by the radiation length X_0 which is defined as the depth of material an electron (or positron) has to penetrate in order to lose $\sim 63\%$ (1-1/e) of its energy:

$$X_0 = 716.4\,\mathrm{g cm^{-2}} \frac{A}{Z(Z+1)\ln(\frac{287}{\sqrt{Z}})}. \tag{6}$$

The mean free path of photons with energies above 1 GeV can generically be expressed in terms of the radiation length as $\lambda_0 = \frac{9}{7} X_0$.

Due to the particle multiplication process, the average shower energy can be expressed in a decay law of the shower-inducing particle's original energy E_0 and the current penetration depth x, as

$$\langle E(x) \rangle = E_0 e^{-x/X_0}. \tag{7}$$

By setting the boundary condition $E(t_{max})=\epsilon_{crit}$ the shower maximum t_{max} can be found by

$$t_{max} \approx ln\left(\frac{E_0}{\epsilon_{crit}}\right) + t_0, \tag{8}$$

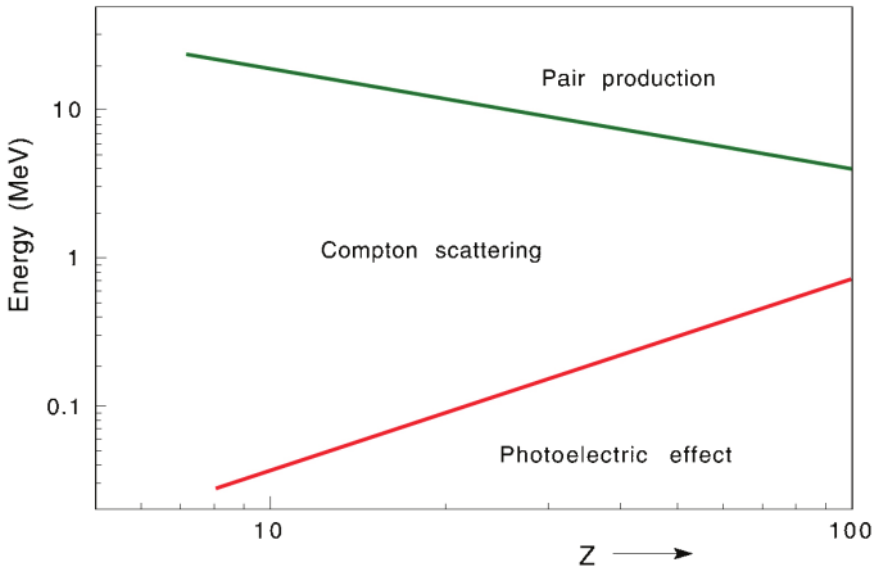


Figure 9: Photon energy domains in which photoelectric effect, Compton scattering and pair production most likely occur as function of the atomic number Z. Image reproduced from Ref. [61].

with t_0 being a correction factor[5] and t_{max} defined in radiation lengths X_0. It becomes apparent that the shower maximum only grows logarithmically with the incident particle's energy. For this reason electromagnetic calorimeters can cover a large range of energies while being constructed rather compact in size.

Besides longitudinal development the transverse extent is characterized by two processes:

- Electrons and positrons diverge from the shower axis via multiple scattering with the atoms in the absorber material.
- Photons, electrons and positrons diverge via the isotropic processes of Compton scattering and photoelectric effect.

While the first effects dominate for early shower stages the second ones dominate after the shower maximum is reached. The empirical quantity describing the transverse extent is the Molière radius R_M, defined as a cylinder around the shower axis containing 90% of the shower's energy.

$$R_M \approx 21MeV \frac{X_0}{\epsilon_{crit}(MeV)}. \tag{9}$$

Both R_M and X_0 are defined for the energy regime $> 1GeV$ [35]. It is noteworthy that R_M and X_0 are just macroscopic approximations and should not be taken at face value.

The definition of X_0 must be considered fundamentally different for photons and electrons. In Fig. 10 the distribution of deposited energy fraction in the first five radiation lengths of a 10 GeV electron and photon showering in lead can be seen. Electrons start interacting in the first few millimeters, while photons may or may not interact in the

[5] t_0 is $-0.5\,\mathrm{X}_0$ for electrons and $+0.5\,\mathrm{X}_0$ for photons as shower-inducing particle [35].

same depth of material. On average, electrons deposit a larger amount of energy (21.0%) with a lower spread (±6.4%) than photons (14.8%) with a larger spread (±8.6%).

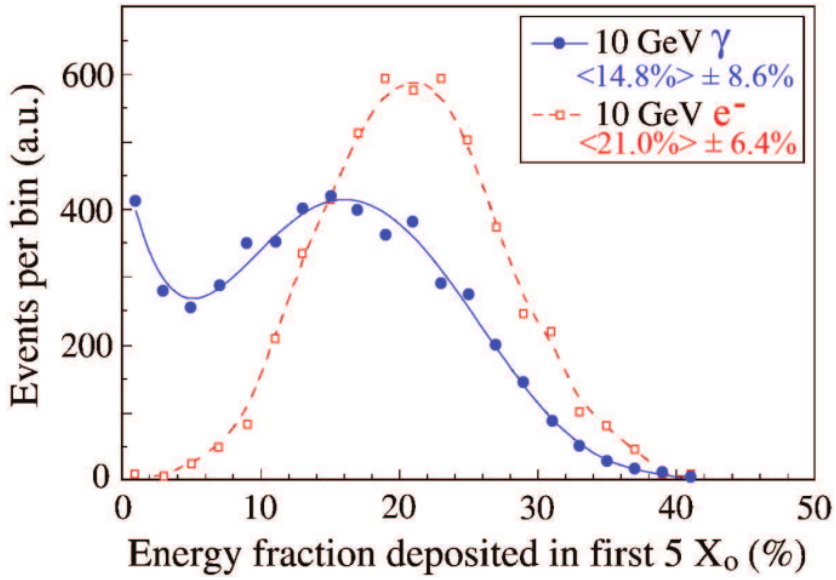


Figure 10: Distribution of energy fraction deposited in first 5 X_0 by 10 GeV electrons and photons showering in lead [61].

In contrast to electromagnetic showers, hadronic showers are more complex because they contain electromagnetic and hadronic components. The majority ($\sim$ 90%) of particles inducing an electromagnetic shower component are due to π^0s decaying into two photons, which in turn are subject to pair production. The hadronic shower component on the other hand is caused by nuclear reactions where neutrons and protons are released from the atomic nuclei of the absorber material's atoms when hit by hadrons. The nuclear binding energy which has to be overcome for nuclear reactions to occur will be lost in this interaction and cannot be measured anymore. This energy is referred to as invisible energy and makes it in general harder to infer incident energies of hadronic showers [61].

Calorimeters can be constructed either as homogeneous calorimeters or sampling calorimeters. Homogeneous calorimeters are built from a single material that causes the generation of secondary particles (absorber material) and produces the measurable signal (active material). An example of a homogeneous calorimeter is the current CMS ECAL which uses $PbWO_4$ scintillators as homogeneous material. Sampling calorimeters, on the other hand use alternating layers of absorber and active material, such that they only measure a sample of the generated secondary particle population. The current CMS HCAL and the HGCAL are examples for sampling calorimeters.

3.2 Silicon Sensor

The active materials of HGCAL are silicon sensors and scintillator tiles.[6] The silicon sensors, seen in Fig. 11, are produced from 8-inch silicon wafers and cut into hexagonal shape in order to optimize the usable wafer surface area and thereby reducing production

[6] In this thesis only silicon sensors are discussed.

costs. The bulk of the material is p-doped with smaller n-doped implants on the surface of the material, such that PN-junctions are formed. The sensors are produced in two variants, one with a higher cell count (444 cells) and pitch size of 0.5 cm (high density) and one with a lower cell count (198 cells) and pitch size 1 cm (low density). High density sensors are only used for high η-regions, close to the beam pipe.

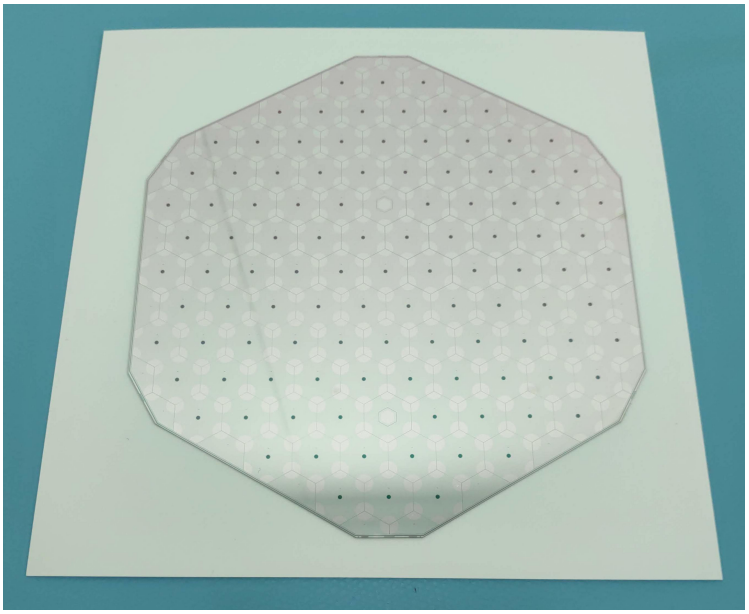
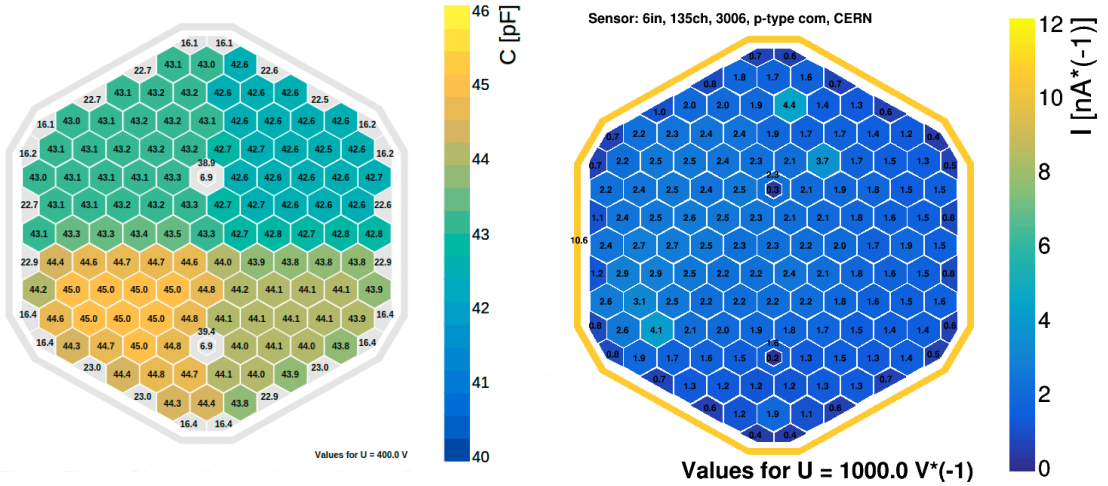


Figure 11: 8" low density prototype HGCAL silicon sensor. The two calibration cells are located on a central vertical line in the upper half and lower half of the sensor.

The sensors will be operated in reverse bias with a voltage between 150 V to 800 V, depending on sensor thickness and the amount of radiation received, which fully depletes the silicon over the thickness of the bulk material. The two characteristic properties of a sensor are the leakage current and the capacitance as function of voltage. While the capacitance depends on inhomogeneities in sensor thickness, cell size and doping, the leakage current is sensitive to ambient temperature, light, and variations in doping. A major contribution to the leakage current is due to radiation damage to the bulk of the silicon. The amount of leakage current at a given voltage is an important characteristic of a sensor, as it is the main source of power dissipation and electrical noise during operation with large bias voltage. In order to reduce leakage current, the HGCAL is planned to be operated at low temperatures of $-30\,^{\circ}\mathrm{C}$. Results of electrical sensor characterization can be seen in Fig. 12.

When charged particles traverse the sensor thickness they lose energy due to ionization in the silicon. For each 3.6 eV in energy loss (via ionization) on average one electron-hole pair is created which forms the signal that is read out. A MIP (minimum ionizing particle) with 81 keV in ionization loss can therefore create about 22 500 electron-hole pairs in 300 μm of silicon. Photons rarely interact with the silicon due to the high Z-dependence ($\sim Z^5$) of the cross-section for photoelectric effect $\sigma_{p.e.}$[7]. However, the photons transfer

[7]Photons more likely interact with the absorber material by the photoelectric effect, creating photoelectrons in the process.



(a) (b)

Figure 12: (a) Capacitance at 400V for LD 6" prototype sensor. (b) Single cell leakage current at 1000V for LD 6" prototype sensor. Figures reproduced from Ref. [23].

energy via Compton scattering with the conduction electrons.

Due to the strong electric field induced by the applied bias voltage causes electrons and holes are quickly collected and conveyed to the readout chip. When no electron is traversing the sensor there is still a small leakage current present, which is the product of spontaneous electron-hole productions due to excitation through ambient temperature and impinging low energetic photons[8]. A low leakage current is desired since it has a significant impact on the power dissipation and signal-over-noise ratio of the measurement.

Sensors include two types of cell that are used for different purposes. While the regular readout cells make up the vast majority, there are also 6 (12) calibration cells for low density (high density) sensors. Calibration cells are smaller in size such that they have lower capacitance, hence also lower noise, and are sensitive to smaller charges than the regular cells. The purpose of these cells is to enable a re-calibration of the sensor on minimum ionizing particles (MIP) which regular cells are not able to distinguish from noise after some time of operation due to irradiation damage.

The sensors are mounted under hexagonal PCBs (hexaboards) which have several ROCs that are connected to the sensor cells via wire bonds. In order to mitigate the negative effects of common-mode noise on an event-by-event basis the readout chips include special channels for common-mode measurement that are not connected to any sensor cell. Using these channels, fluctuations in the pedestals (cf. Sec 4.3.3) are recorded and later subtracted from the measurements.

3.3 Readout Chip

The readout chip (ROC) of HGCAL is called HGCROC and is responsible for energy and timing measurement of secondary shower particles ionizing and exciting the sensor

[8]Visible light has a large cross section for photoelectric interaction.

material. One HGCROC can connect to sensor cells via 72 channels that each have their individual on-chip amplification and measurement circuitry. By design, the chip is separated in two identical halves that equally divide the number of channels. In order to readout all 198 (444) sensor cells of low density (high density) sensors 3 (6) ROCs need to be mounted on a hexaboard and connected to the cells.

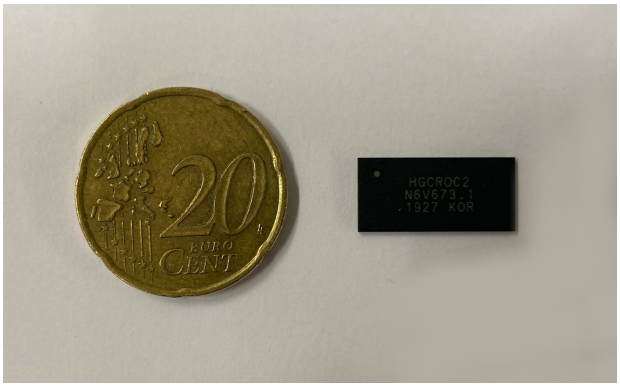


Figure 13: The (packaged) HGCROC at scale. The silicon inside the package is even smaller.

The HGCROC provides energy and timing measurements via three mechanisms.

1. Smaller energies are measured directly via a 10-bit analog-to-digital converter (ADC) after amplification and shaping.

2. Larger energies are measured indirectly via a 12-bit time-to-digital converter (TDC) by using the time-over-threshold (TOT) technique. This technique utilizes a discriminator comparing the input signal to a constant reference voltage while controlling an adjustable current source to feed back a linearly decreased signal scaled by the input. Thereby, the TDC measurement is started once the input signal exceeds the reference voltage and stopped when the input charge is reduced to a value below this threshold. The measured time interval is, due to the linearity of signal degradation itself, linearly proportional to the input signal. Ideally, the time-over-threshold measurement is initiated once the ADC is unable to measure a given input signal, e.g. once the signal exceeds the ADC's dynamic range.

3. The time of arrival (TOA) of particle(s) hitting a sensor cell is measured via a 10-bit TDC using a discriminator. However, since only a timestamp is required the TDC measurement is triggered merely once the input signal exceeds the discriminator's reference voltage. This voltage is set to a relatively low threshold of about 100 ADC counts in order to record inputs of all energies. Yet, it should not be too low to avoid a spurious TDC activation by electronic noise (up to $\sim$ 10 ADC counts in prototype systems).

The input signal from the silicon sensors comes from the amount of charge collected, which in variable in a large dynamic range of a few to hundreds of thousands of femtocoulomb. This charge is converted to a voltage by the preamplifier and further redirected to the three above mentioned measurement circuits. The energy deposited by a particle in the active sensor material is linearly proportional to the amount of charge it creates.

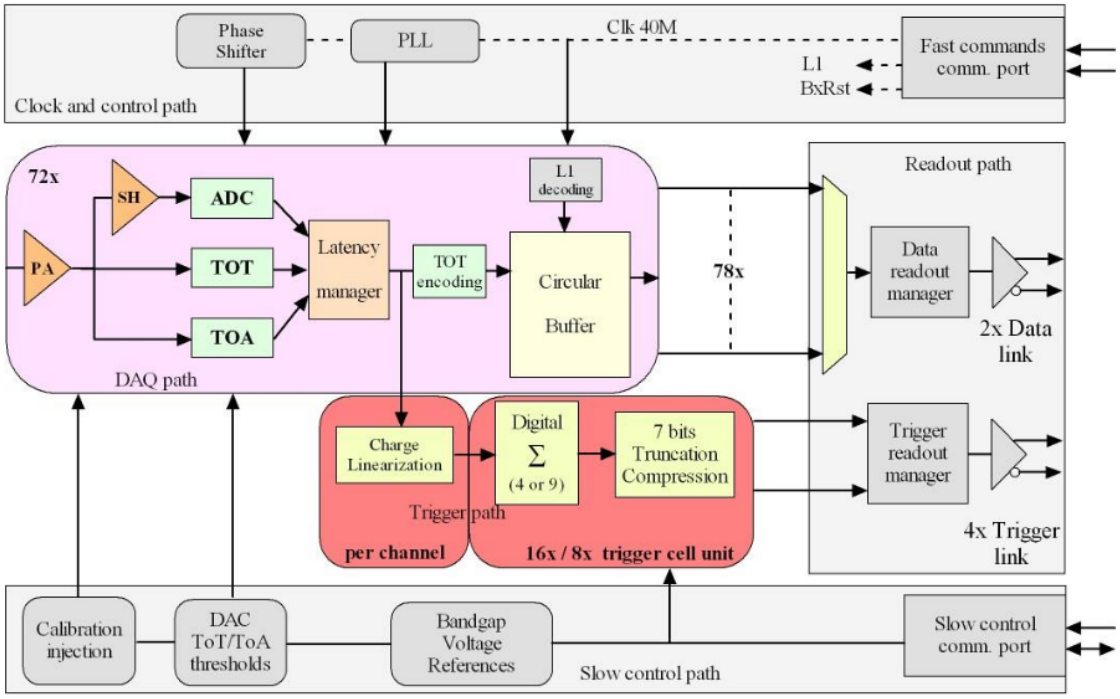


Figure 14: Schematic architecture of HGCROCv2 [59].

The chip can be read out and configured via two separate communication ports called fast commands (for data acquisition) and slow controls (for chip configuration). Besides the raw sensor data, the HGCROC also provides trigger data, a coarser and more compressed version of the sensor data that is used by the HGCAL trigger primitive generator. For this purpose the amount of measured charge of four (optionally nine) neighboring sensor cells is summed to build a trigger cell. The data from a trigger cell is converted into the 7-bit fixed point data format 4E3M (4 exponent, 3 mantissa) which can be read out together with the raw data for every bunch crossing. Only after the CMS trigger system has decided to keep a bunch crossing the full ADC, TOT and TOA data is moved to permanent storage. For this purpose, the full data is temporarily buffered until the trigger decision is made.

3.4 Trigger Primitive Generator

HGCAL's trigger primitive generator (TPG) will be part of the CMS Phase-2 Level-1 Trigger (L1T) system seen in Fig. 15 as *HGCAL* under the *Calorimeter trigger*. The purpose of the TPG is to produce coarse-grained three-dimensional energy clusters, referred to as *trigger primitives* for the global calorimeter trigger and the correlator trigger. Inside the correlator trigger the trigger primitives of TPGs from all sub-detectors of CMS are combined and analyzed in order to generate a trigger decision, e.g. to decide if a bunch crossing, contained interesting (with respect to the objectives of the CMS physics program) events. If the L1T decides to keep a bunch crossing the high level trigger (HLT) is notified in order to examine the accepted bunch crossing more carefully by using the full resolution data and previously omitted tracker information with more sophisticated

algorithms. Only after the HLT decides to keep the data it is written to long-term storage. As mentioned in Sec. 2.3, the purpose of the trigger system is to drastically reduce the bandwidth of data storage by retaining only interesting events.

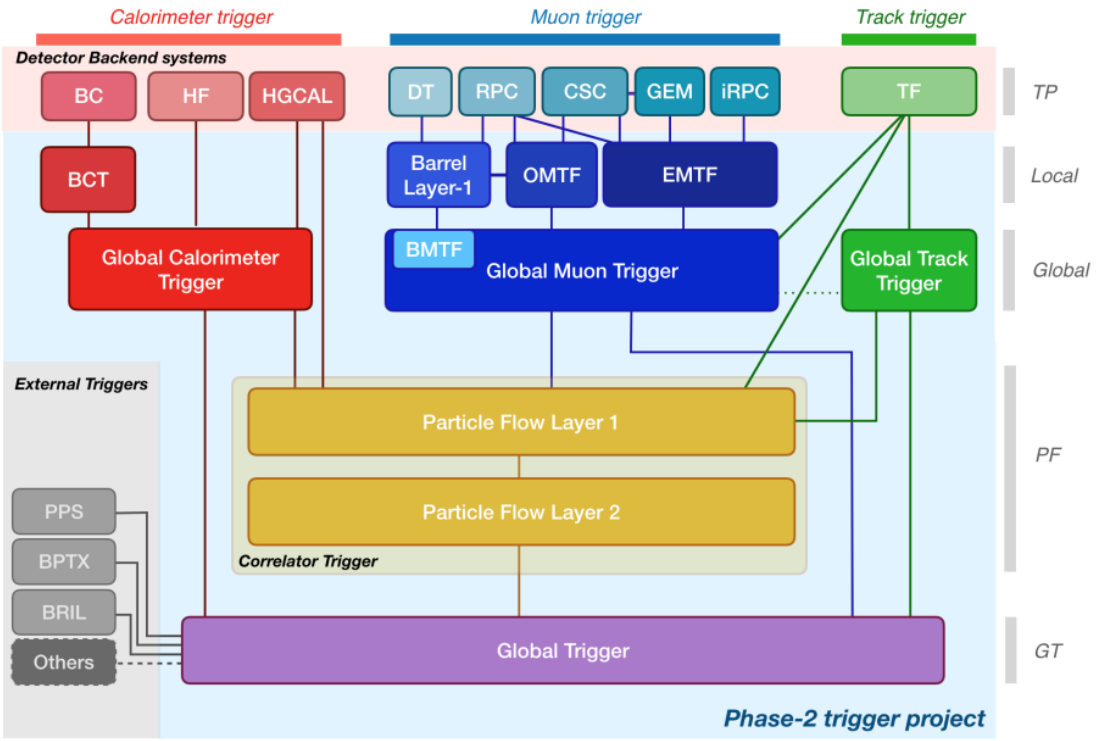
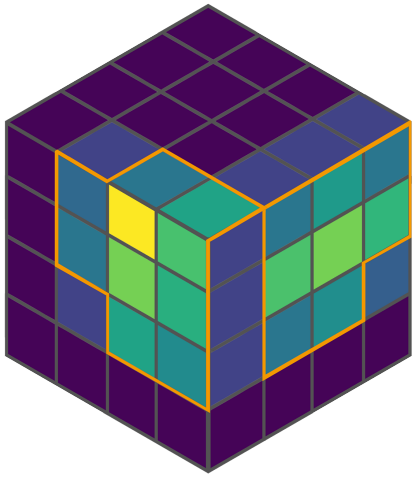


Figure 15: The CMS Phase-2 Level-1 Trigger system [18].

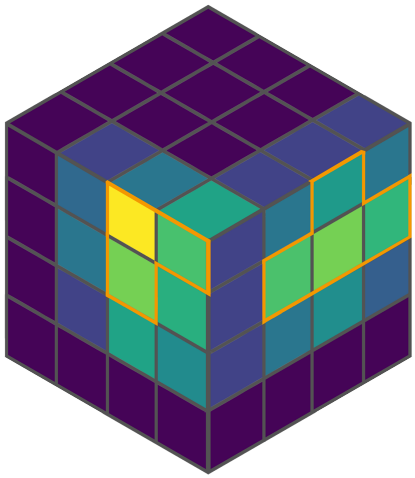
The trigger primitive generator has to achieve a data reduction of the high grained calorimeter data by a factor of 400. The TPG accomplishes this by means of processing the calorimeter in four successive stages: ROC, trigger concentrator chip (ECON-T), Stage-1 and Stage-2 trigger processors. The last two are implemented on FPGAs in a close-by computing cavern, while the ECON-T resides on the frontend electronics implementing four algorithms for compressing the sensor data:

- **Threshold** excludes trigger cell data below a certain threshold (Fig. 16a).
- **Best Choice** includes only N cells with highest values (Fig. 16b).
- **Super Trigger Cell** sums neighboring trigger cells (Fig. 16c).
- **Generic encoder** is an algorithm that allows to perform data compression.

Studies have shown that none of the first three algorithms performs well on all types of event data [51]. The fourth, generic encoder block, has recently been implemented to explore the application of machine-learning techniques in order to get stable trigger performance on all types of physics objects. Besides the ECON-T, according to the TDR [18] the Stage-1 and Stage-2 processors reduce the data bandwidth by clustering into 2-dimensional and 3-dimensional clusters, respectively. The three compression stages in the HGCAL TPG will be the subject of chapter 5.1.



(a)



(b)

(c)

Figure 16: Illustration of ECON-T algorithms for an example of trigger cell energy values in a module: (a) Threshold, (b) Best-Choice, and (c) Super Trigger Cell.

4 Readout Chip Characterization

The design of the two HGCAL endcaps includes a total of about 30 000 sensors containing about 6 000 000 Si channels that each needs to be read out. Therefore, an amount of about 100 000 HGCROCs must be produced, mounted and tested [15, 40]. The final chip and hexaboard testing will take place at six mechanical assembly centers (MACs). By testing a significant amount of chips (223 so far) individually, on PCBs and on PCBs attached to sensors data for characterization has been taken.

Characterization of electronics devices (i.e. chips) refers to determining statistically characteristic values for key properties. By measuring these properties together with the impact on the performance of the devices, a quality control framework can be established accepting only devices within a certain range of characteristic values. Additionally, it enables a post-production tuning of these properties, per chip, in order to bring them into the accepted range. To create such a framework is the overarching task towards which the work in this chapter is directed.

4.1 Process variation

Process variation describes the variation of physical properties such as length, width and oxide thickness of semiconductor devices [44]. This variation causes measurable differences in performance between components that have been produced in the same way (e.g. same batch). Although digital circuits are affected as well, the effect on analog components is more significant due to their operation in a continuous range [32].

The HGCROC has the task to amplify naturally weak occurring signals caused by particles interacting with sensor material to an accurate and measurable one. To achieve this goal the ROC includes for each read-out channel several analog components that are termed *the analog front-end*. The components of the analog front-end together with the calibration circuit (which is shared amongst channels) can be seen in Fig. 17. Process variation, the requirement of precision and the amount of analog components inside the ROC motivate the characterization efforts.

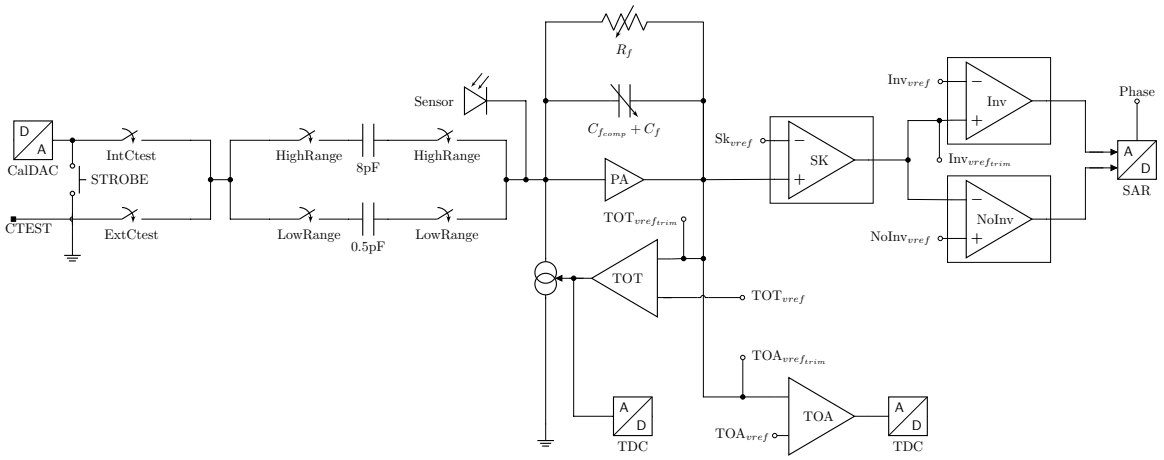


Figure 17: Analog front-end of HGCROC

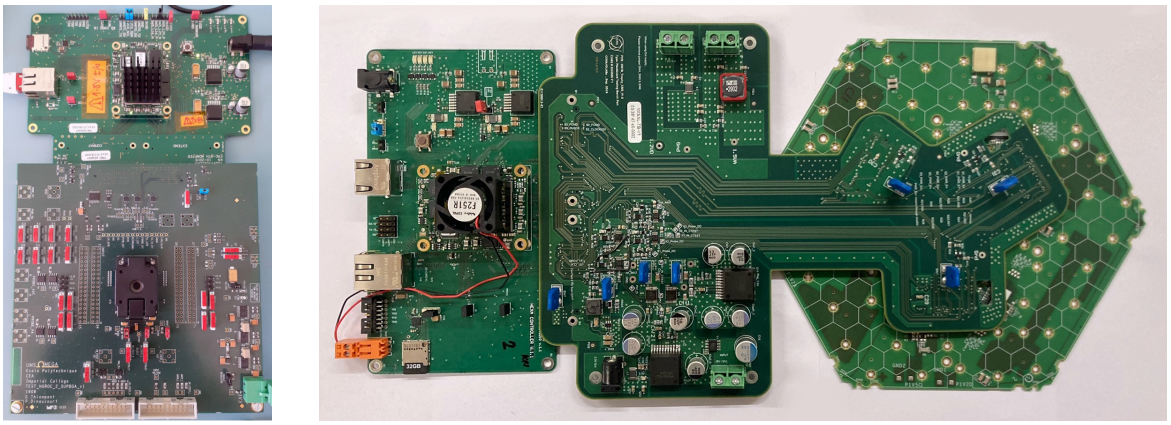
4.2 Strategy

Besides taking data for characterization, another major focus lies on the accumulation and transfer of knowledge attained from online analysis (at the time of testing) to the chip designers. As the HGCAL project foresees three HGCROC generations to obtain the final readout chip, the generated knowledge from testing has a direct impact on design decisions of the subsequent generation. At the time of writing the second generation of HGCROCs was subject to tests. The online analysis is performed to investigate certain chip components that are of immediate interest and allow a direct evaluation about the functioning of a chip. In general, single chip, hexaboard and module tests form three distinct stages of testing to assure quality along the different production steps.

Additionally, the chip designers built several digital-to-analog converters (DACs) on critical analog components into the ROC in order to allow for post-production tuning. Some of these handles can be set per half ROC (global) while others are set per channel (local). The DACs can be programmed via specific configuration commands that usually change the voltage supplied to a specific circuit element. The handles postfixed by *trim* in Fig. 17 represent local handles while all others are global.

4.2.1 Hardware

Due to the three testing stages, two different test systems have been developed. For single chip tests the single-chip characterization (ZIF socket) board, shown in Fig. 18a is used. The central piece of the single-chip board is the black ZIF (zero insertion force) socket sledge that allows to insert a ROC and to press the balls onto metal contacts that establish an electrical connection. Furthermore, the board includes two connectors (at the bottom of the figure) that breakout specific pins of the ROC to external devices.



(a) ZIF-Socket board.

(b) Trophyboard with hexaboard.

Figure 18: Test boards attached to hexacontroller.

For hexaboard and module tests the test system depicted in Fig. 18b is used. The hexaboard is attached via three connectors (not seen here) to an intermediate board referred to as *Trophyboard*, which allows to read out the three (six) chips mounted on low density (high density) hexaboards. The Trophyboard houses six ADCs for reading out the same pins that the ZIF socket board provided by the two break-out connectors.

Both Trophyboard and single-chip board are in turn connected to a control board called *hexacontroller*. The hexacontroller uses a Trenz TE0820 System on Module integrating a Xilinx Zynq UltraScale+ FPGA and processing unit. The processing unit runs the Linux distribution CentOS while the FPGA is programmed via a custom-designed firmware to enable high-speed communication to the testing boards. The FPGA configures the ROCs, controls the data acquisition, collects the data from the ROCs and makes them accessible to the OS.

4.2.2 Software

The HGCROC provides two independent interfaces for chip configuration and data acquisition. To configure the parameters of the analog front-end (and the digital part of the ROC) so called *Slow Control* messages are send via the inter-integrated-circuit (I^2C) bus to the chip. For further information on the I^2C bus and protocol the reader may refer to [4]. For setting up and initializing a data acquisition so called *Fast Commands* can be sent via the Advanced eXtensible Interface [62] (AXI). The ROCs are continually sending data (consisting of a bit pattern that represents an idle status) through a high-speed connection (eLink) with 1.28 Gb/s. Only after sending an L1A signal to the ROC, the FPGA firmware expects readout data on the eLinks after some latency and starts the acquisition. Due to the separation and independence of the two interfaces software development was carried out in parallel. A schematic overview of the software setup can be seen in Fig. 19. Measurement, data storage and online analysis is triggered by an external client computer via ethernet.

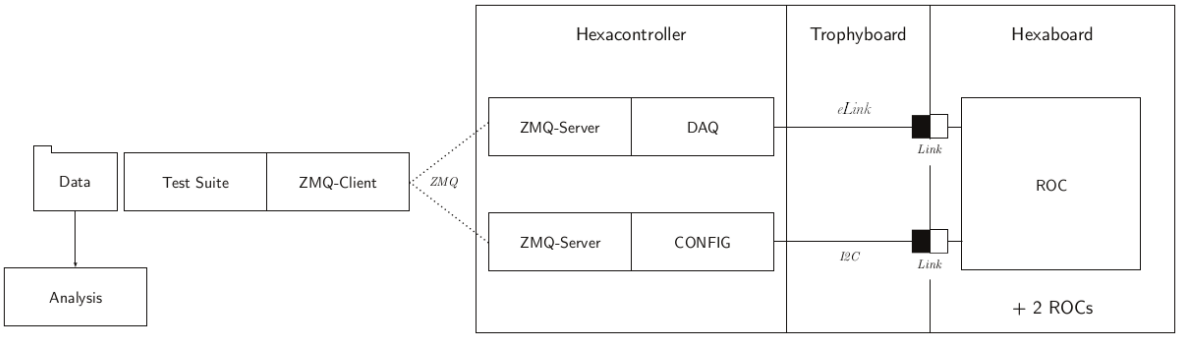


Figure 19: Software architecture of test systems. The client computer (left) sends configuration files to the hexacontorller server (right), which configures the ROCs and sends the data back to the client which stores it for later analysis.

In order to characterize and tune components of the analog front-end internal DACs need to be set. The ROC provides eight I^2C addresses of which four (R0 to R3) grant access to setting the values of the DACs. As seen in Fig. 20, writing into registers R0 and R1 form a 16-bit address pointer that references a specific register in HGCROC memory. After the pointer is set, writing the desired value byte to R2 fills the referenced register with the specified value. Similarly, writing the value byte to R3 writes to the same location and increments the address pointer by one after the write, such that another write to R3 would fill the register at the successive address. Thus, taking data for characterization consists of successive configuration and data acquisition.

I2C @	Register	Comments
R0	ASIC parameter address (LSB)	Indirect @
R1	ASIC parameter address (MSB)	Indirect @
R2	Data	
R3	Data with auto @++	Increment indirect @ after each access
R4-R5-R6	Direct access SC-register	
R7	Status register (error, parity)	Read-only

Figure 20: Table of I^2C registers and their function [59].

Software development efforts were directed by three overarching requirements:

1. Remote operation,
2. Compatibility across all testing systems, and
3. Performance and fault tolerance.

As mentioned before, one of the author's main contributions was the development of software for configuration. Therefore, the next sections give a brief overview of the software engineering decisions made in order to fullfill the above mentioned requirements.

Remote operation is facilitated via a TCP socket established by the embedded networking library ZeroMQ (ZMQ) [10]. The hexacontroller running the configuration software acts as server while the remote computer initializing tests is the client (Fig. 19). Control flow between client and server is handled via the request/reply pattern. The client sends a request for configuration which the server handles by starting a new thread for each ROC to configure. When each thread finishes successfully, the server sends a reply signaling that it is ready for configuration again. If an error occurs during the handling of the request, the client is notified with an error code in the reply.

The ROC understands a configuration as a sequence of bytes written to certain registers. However, in order to create a more user-friendly interface to configuration the software includes a conversion table of human-readable parameter names to register addresses, referred to as the *Register Map*. This map enables the user to write structured documents as YAML-files which are then sent to the server. YAML is a human-readable data-serialization language commonly used for configuration files. As certain parameters span more than one register while others occupy only a fraction of a register the conversion routine was crafted in a generalized way considering all edge cases. In case a register's value cannot be inferred from the information available at a given time, the value was read from the ROC in order to not overwrite information.

Compatibility with all test systems is assured by an automatic detection routine, scanning all available I^2C busses for responding ROCs. As depicted in Fig. 20 one chip will show up as eight consecutive I^2C addresses on a bus. The amount of ROCs found on the busses determines the platform and thereby the test system that is used: One ROC for single-chip, two ROCs for scintillator tileboards, three ROCs for low density hexaboards and six chips for high density hexaboards. All test boards are designed in a way that the first of the eight consecutive I^2C addresses has a unique value. This allows for example in hexaboard systems to detect and correct for the state of rotation. This is accomplished by defining a mapping of *correct* rotation between ROC address and bus ID which in turn depends on the pin assignment defined in the firmware and thereby does not change. However, since the configuration software is independent of the software for data acquisition, the latter needs to be notified about the state of rotation as well in order to establish the correct mapping to the readout links.

The requirement of **performance** has been achieved by avoiding redundant I^2C transactions and by caching. The most significant performance boost in terms of redundant bus transactions came by saving the most recent written address and to reuse it in subsequent read/write operation. Many operations use partially overlapping or identical registers R0 and R1 which can be exploited by partially (*lazily*) updating the state of these registers. This mechanism was further enhanced by sorting the operations of a configuration request by similar addresses before writing and using the burst-write register R3 for truly consecutive addresses. The results of these enhancements on I^2C transactions can be seen in table 1. Another bottleneck was discovered in converting between parameter and reg-

Table 1: Amount of bus read/write operations for different configuration files. The init.yaml file contains the initial configuration of the ROCs, while all other yaml-files are used to configure the ROC for certain tests. Because these files are executed with each run, a full test requires configuring the ROC with the almost identical configuration file several times in a row. The amount of successive configurations for testing purposes is indicated in parenthesis behind each configuration file.

Config file (configurations)	**Regular R/W**	**Enhanced R/W**	**Speedup**
init.yaml (1×)	708	155	4.6×
scan_pedDAC.yaml (32×)	16512	5250	3.1×
scan_calibdac.yaml (41×)	984	249	4×
config_samplingscan.yaml (16×)	16224	397	41×
scan_vrefinv.yaml (30×)	1440	361	4×
deconfig_samplingscan.yaml (1×)	1008	104	9.7×

ister information. Since the register map used for conversion contains about 2500 entries, continuous table look-ups turned out to consume a significant amount of processing time. By closer inspection of commonly used configuration files and their respective registers, it became clear that only a small subset of registers occur often. By implementing a cache of the conversion, the total runtime for configuration was reduced by a factor of about two.

In terms of **fault tolerance** another type of cache was developed, termed *Write cache* (Fig. 21). Besides storing all address-value pairs that have been read or written since initialization, the cache avoids redundant write operations. If an error in one of the I^2C transactions occurs, a reset of the I^2C state machine and the ROC registers is issued

before the chip is reconfigured to the last known state saved in the Write cache. After this procedure, the last configuration sent from the client is repeated.

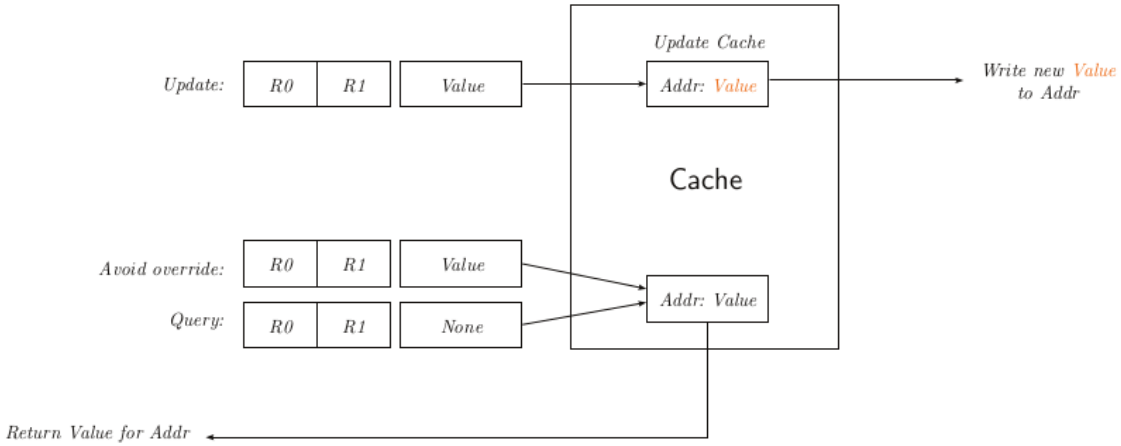


Figure 21: Write cache saves state and avoids redundant writes. It is written to or queried via an address–value pair. The address consists of a tuple (R0,R1) and the value is a byte to be written to the specified ROC register.

4.3 Tests and results

In order to characterize the different components of the ROC's analog front-end, several procedures have been developed and tested. Besides generating characterization data, these procedures include online analysis to give fast feedback which proved to be advantageous in understanding the general behavior of the components as well as the interdependence between components of the analog front-end. The procedures can be controlled via a client computer that communicates with the software suite described in section 4.2.2. The procedures are planned to be used in the final chip, hexaboard, and module tests in the MACs. As the focus of this thesis lies on the implementation of procedures, analysis will only be discussed briefly.

4.3.1 Power consumption

Measuring power consumption gives quick insight into the functioning of a chip. After powering on, a list of default parameters is loaded from permanent memory into the digital and analog parts. In this state a ROC (on a hexaboard) consumes about 2.9 W. Chips that showed a drastic deviation from this value (by about 0.3 W to 0.5 W) did not respond to I^2C communication and were therefore declared malfunctioning - in total 4 of the 223 tested chips behaved like this. Characterization of power consumption allows to reject chips in an early testing stage which is advantageous when, at a later point in time, thousands of chips need to be tested.

For single-chip and multi-chip systems there are two different ways to measure power consumption:

- After connecting the single-chip characterization board (ZIF socket board) to a power supply unit, a constant voltage of 3.3V is being supplied and the drawn current is measured via the connected power supply. The power consumption is calculated by $P = U \cdot I$, which includes the current drawn by the board itself.

- For multi-chip test systems the current drawn is read out via six current sensing amplifiers (INA250A2 [58]) positioned on the interfacing Trophyboard. Each sector[9] of the hexaboard is connected to two current sensing amplifiers for measuring the consumption of the analog and digital parts separately. The INA250A2 is designed with a gain of $G = 500\,\mathrm{mV/A}$ and supplied with a voltage of $V_{sup,dig} = 1.2\,\mathrm{V}$ for the digital part and $V_{sup,ana} = 1.5\,\mathrm{V}$ for the analog part. The output signal of the sensing amplifiers is subsequently converted into a digital signal by ADCs [11] which are read out by the hexacontroller via I^2C. Power consumption is calculated from the ADC output V_{INA} (automatically converted by the Linux I^2C ADC Kernel driver) as $P = V_{sup} \cdot V_{INA}/G$. The total power consumption of the chip is the sum of consumptions from the digital and analog parts. Typically $P_{ana} \approx 2.5\,\mathrm{W}$ and $P_{dig} \approx 0.5\,\mathrm{W}$ for low density hexaboards (one chip per sector) and twice these amounts for high density hexaboards (two chips per sector) after power-on.

4.3.2 Probe points

The ROC is designed to allow reading out several direct current (DC) probe points located at critical components in the circuitry. Characterizing these DC levels across chips (and their halves) helps the chip designers better understand the impact that process variation has on the chip design. Due to different architectures for single- and multi-chip systems there are two different ways to measure the probe points:

- Single-chip boards expose two ports (one per half) that provide two pins each (ProbeDC1 and ProbeDC2) for an external device to read from. The probe points must first be configured and can then be read out, one at a time by a connected digital multimeter via GPIB.

- Hexaboards break-out the ROC's ProbeDC pin per sector to the Trophyboard. The Trophyboard includes one ProbeDC ADC per sector, accessible to the hexacontroller via I^2C. Since the halves cannot be read out in parallel as in the case of single-chip boards, care must be taken when configuring the chip. For low density hexaboards the probe points must be read out sequentially first for one half, then deactivated and afterwards read out for the other half. For high density hexaboards (two chips per sector) this procedure must be done sequentially for both chips of a sector.

The results of measuring the probe points on different chips can be seen in Fig. 22. The variation of some points is stronger than for others. Especially the TDCs *VD_*TDC** and the reference voltage of the pre-amplifier *Vref_pa*, *VrefCf* and *Vcp* seem to be particularly effected by process variation.

Another probe point called $CTEST$ is used to characterize the internal charge injection DAC termed $CalDAC$. $CalDAC$ is used in most of the characterization tests and has

[9] A third of a hexaboard is defined as a sector. Low density modules mount one ROC per sector and high density modules mount two ROCs per sector.

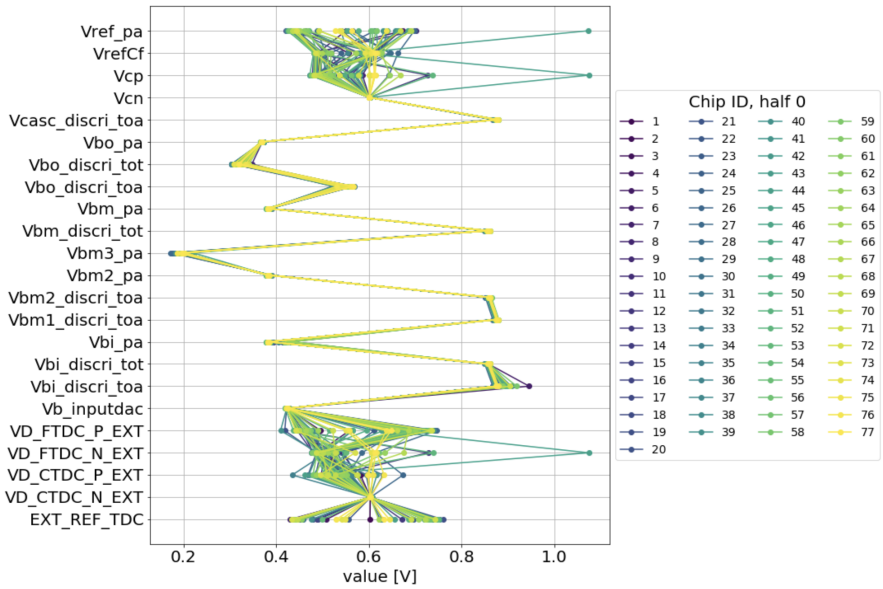


Figure 22: ProbeDC measurements for one half-chip across chips [42].

thereby an implicit impact on the measurements. Therefore it makes sense to characterize $CalDAC$ as well by measuring voltages that are produced after setting particular DAC values. Voltage as a function of the DAC values are expected to increase linearly with a spread in slope and offset between chips as result of process variation. The results in Fig. 24 show a more coherent behavior for lower $CalDAC$ values. The offset at $CalDAC = 0$ is almost identical between chips, however a dispersion in slope becomes apparent when looking at larger $CalDAC$ values.

To accomplish the measurement of $CalDAC$, the switches $IntCtest$ and $ExtCtest$ must be closed, while switches $HighRange$ and $LowRange$ are open (Fig. 23). By triggering the $STROBE$ (Fast Command) signal, $CTEST$ can be read out either via an external device over the $CTEST$ pin (for single-chip) or via the $IntCtest$ ADC on the Trophy-board.

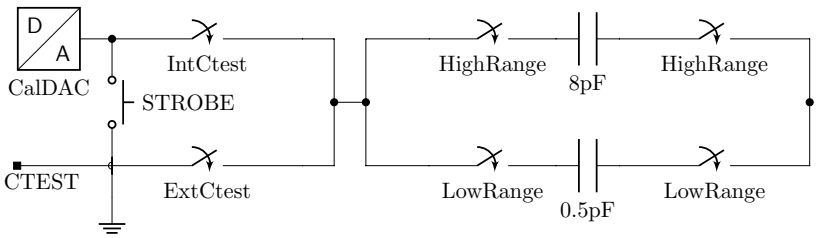


Figure 23: HGCROC calibration circuit. Figure is a zoomed excerpt of Fig. 17.

4.3.3 Pedestals

A pedestal is a DC level that is constant over time. Since real signals are always subject to noise the pedestal is defined as the average (equivalent) DC level of a signal. The baseline pedestal is the DC level that is measured when no charge is injected. It is noteworthy

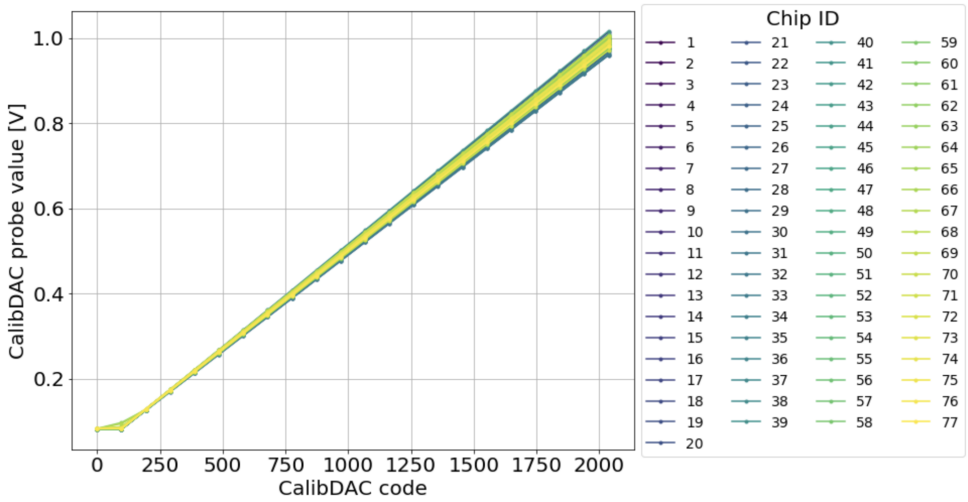
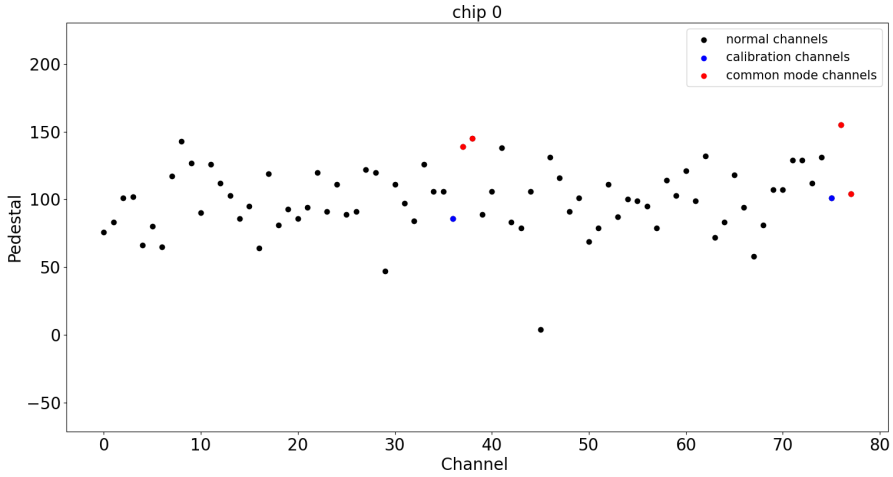


Figure 24: Voltage at $CTEST$ as function of $CalDAC$ value [42].

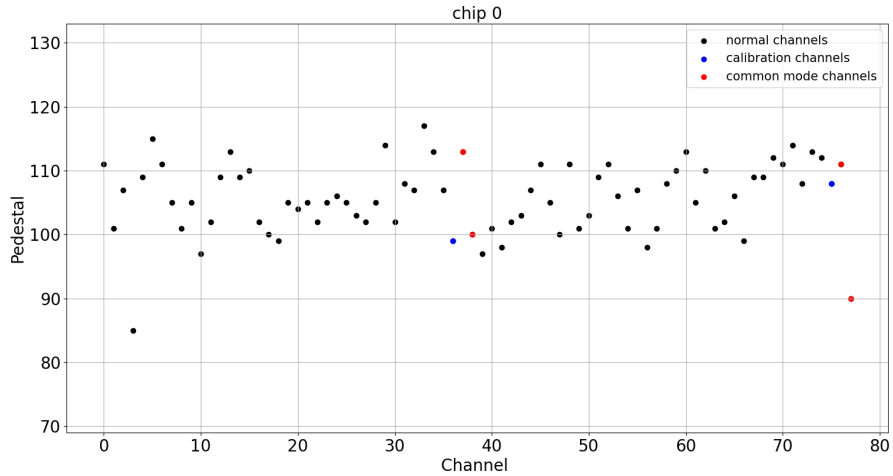
that this level changes when the calibration circuit is connected to a channel. Measuring the pedestals is referred to as *pedestal run*. Since the two halves of a chip are physically separated, the pedestals of channels in the halves tend to accumulate around different baseline pedestals. As can be seen in Fig. 25a there is a large inter-channel dispersion of about 100 ADC counts and a relatively high average pedestal of about 150–175 ADC counts with power-on default settings. Both of these outcomes are not desired. Desired would be a low average pedestal, in order to obtain a larger dynamic range, and an as small as possible inter-channel dispersion, in order to obtain a most similar channel response.

To mitigate such large dispersion and offset, the chip designers included the local shaper handle $Inv_{vref,trim}$ and the global shaper handles Inv_{vref} and $Noinv_{vref}$. While the two global handles are discussed in 4.3.4 the local handles allow each channel's pedestal being adjusted in 32 steps. The act of adjusting local pedestals is referred to as *pedestal trimming* and the encapsulating procedure for finding the optimal pedestal per channel is called *pedestal scan* (Fig. 26). Starting from each channel's baseline pedestals and plotting the ADC response as a function of $Inv_{vref,trim}$, a horizontal line can be found that minimizes the pedestal dispersion by using one particular value for $Inv_{vref,trim}$. Instead of solving this problem analytically, the horizontal line is set to begin at the channel with the highest baseline pedestal (at $Inv_{vref,trim} = 0$). This is often a good enough approximation and requires less computation. A chip's channel is defined as *untrimmable* if its function does not cross the horizontal line. In practice, untrimmable channels have not been observed in any of the tested chips.

After the pedestals have been trimmed, a pedestal run shows the effect (Fig. 25b). It is apparent that the inter-channel dispersion has been reduced from 100 ADC counts to about ten. It is desired to reach a similar pedestal level for each channel per half with low inter-channel dispersion. This allows for using simple algorithms – pedestal subtraction, zero suppression, common-mode noise correction, etc – to post-process the sensor data.



(a) Before pedestal trimming.



(b) After pedestal trimming.

Figure 25: Measurement of baseline pedestals. Note the difference in the range of the vertical axes.

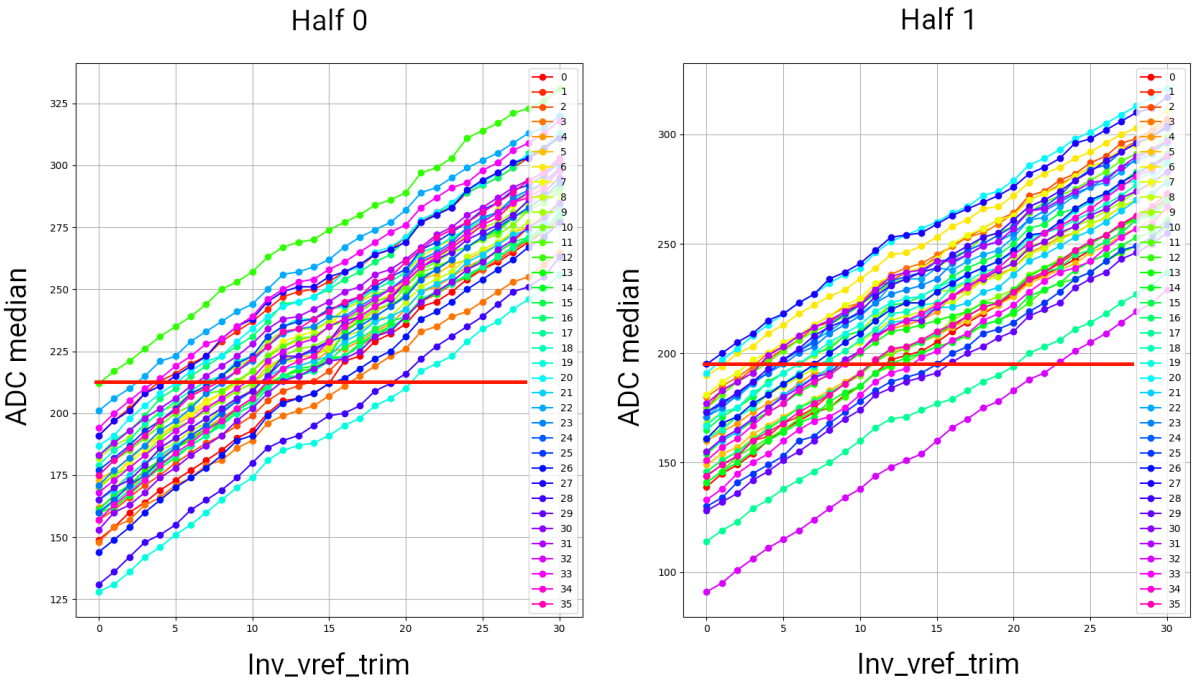


Figure 26: Median of ADC values (pedestal) as a function of $Inv_{vref,trim}$.

4.3.4 Shapers

The shaper block is composed of three stages: A Sallen-Key shaper, the inverting and non-inverting shapers, and unity gain amplifiers (buffers). While the Sallen-Key shaper and the buffers optimize the signal-to-noise ratio [59] and provide a stable signal for ADCs to sample from, the inverting and non-inverting shapers condition its dynamic range. After the signal leaves the Sallen-Key shaper it is fed to both inverting and non-inverting shapers where it is positively (negatively) amplified and further propagated as a differential input to the ADC. Since the shaper gain is susceptible to process variation, it must be characterized. To tune the gain and mitigate inter-half variation in dynamic range, the chip designers built in two global DACs Inv_{vref} and $NoInv_{vref}$. Together with the local shaper handle $Inv_{vref,trim}$ as mentioned in 4.3.3 and the voltage level V_{PA} provided by the pre-amplifier the magnitude of the output signal $S = |S_- - S_+|$ follows from:

$$S_+ = 3 \cdot (Inv_{vref} - Inv_{vref,trim}) - 2V_{PA} \tag{10}$$

$$S_- = 2V_{PA} - NoInv_{vref} \tag{11}$$

To better understand the behavior of the shaper response in practice one can set all possible combinations of Inv_{vref} and $NoInv_{vref}$ as a function of the differential output signal measured by the ADC. The result of such a scan on baseline pedestals can be seen in Fig. 27a. A preceding pedestal trimming has been performed in order to mitigate the impact of inter-channel dispersion on the median ADC value calculated per half. As shown, a horizontal shift between the two distributions becomes apparent which can be attributed to the physical separation of the two halves inside a chip's package. In order to maximize the dynamic range, a low output signal for baseline pedestals (light blue) is desired which would correspond to different Inv_{vref} and $NoInv_{vref}$ settings for the different halves. To see the upper end of the dynamic range, the same scan is performed

but instead of baseline pedestals a charge is injected. The charge is chosen to be so large that it will saturate the ADCs, e.g. $CalDAC = 1000$ using $LowRange$ capacitor. The existence of optimal global shaper settings (where all ADCs will saturate before $CalDAC = 1000$) is assumed. The result is shown in Fig. 27b. The difference between the plots 27a and 27b is the dynamic range (Fig. 27c). Only values of dynamic range larger than 800 ADC counts are plotted to illustrate the existence of a region of optimal global shaper settings which is situated at similar positions for both halves. In order to maximize the dynamic range, a global shaper setting from a central position of this blob is desired.

4.3.5 Analog-to-Digital Converters

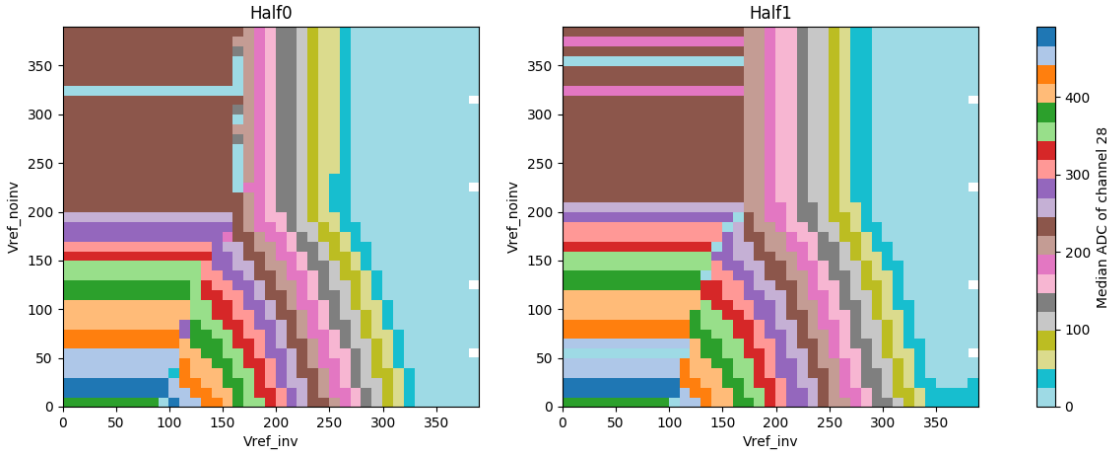
The successive approximation register (SAR) ADC samples the voltage differential supplied by inverting and non-inverting shaper outputs with a 40MHz sampling frequency into a 10-bit digital signal. Due to delay introduced by the preceding circuit the ADC might start converting the analog signal at an unfavorable point in time. To avoid this, the ADC has a phase handle that can offset the start of conversion. The phase parameter is tunable in 16 steps of about 1.5 ns. The 16th step corresponds to one full BX (delay of 25 ns). It is desired that the ADC starts the conversion at the maximum of the input signal. The phase parameter is a global setting. The procedure that detects the ideal phase parameter is called *sampling scan*. In Fig. 28 a phase offset of one, corresponding to 25 ns, would start the conversion at the best timing.

After the optimal phase is set, the ADC's response to input signals can be characterized. By increasing the injected charge, a linear increase in ADC counts is expected, saturating at some point. This test is referred to as *injection scan* (Fig. 29a). The slope of the injection scan and the location of the saturation point is desired to be similar between channels and chips. A non-optimal shaper reference voltage leads to earlier ADC saturation as can be seen in Fig. 29b or no ADC response at all. This is one of the main results from the first testing session of single chips and led to the investigation of dynamic range with respect to the Inv_{vref} and $Noinv_{vref}$ parameters discussed before.

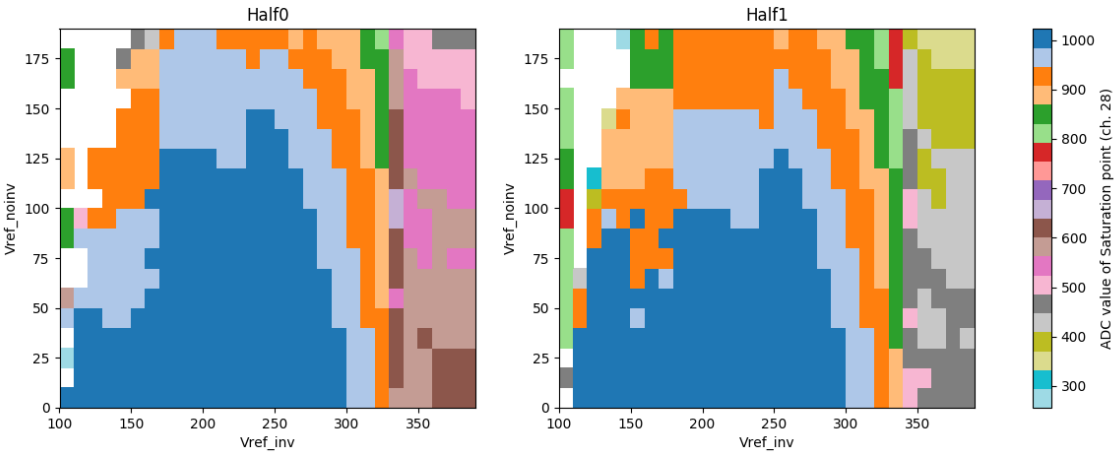
4.3.6 Discriminators

The Time-Over-Threshold (TOT) and Time-Of-Arrival (TOA) discriminators (comparators) each trigger a Time-To-Digital converter (TDC) when the input voltage exceeds a set reference voltage. The TDC used for TOA measurements is activated at the rising edge of the discriminator output to capture the time the signal arrives, while the TDC for TOT measurement triggers on rising and falling edges of the discriminator output to capture the time the comparator was activated, i.e. the time spent over the reference voltage threshold. The TOT measurement allows the measurement of large charges that exceed the dynamic range of the ADC. Therefore, a reference voltage about the amount of charge it takes to saturate the ADC is ideal. For the TOA comparator a reference voltage as low as possible is ideal to record the timing of the smallest amount of input charge without being too low to be triggered by noise. For that reason, the chip designers included global and local DACs on the reference voltage of the discriminators.

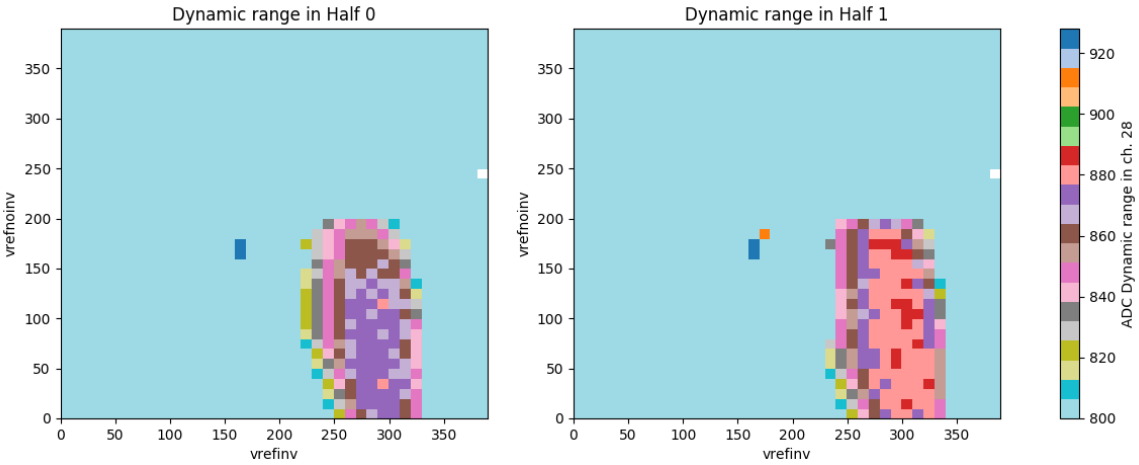
Since it is desired that the chips' TOA and TOT measurements are uniform for identical input signals, the behavior of the discriminators must be characterized. This is



(a) No injection (baseline pedestals).



(b) With injection (ADC saturated).



(c) The ADC dynamic range obtained the difference between Figs. (b) and (a).

Figure 27: Median ADC values as function of the global shaper parameters.

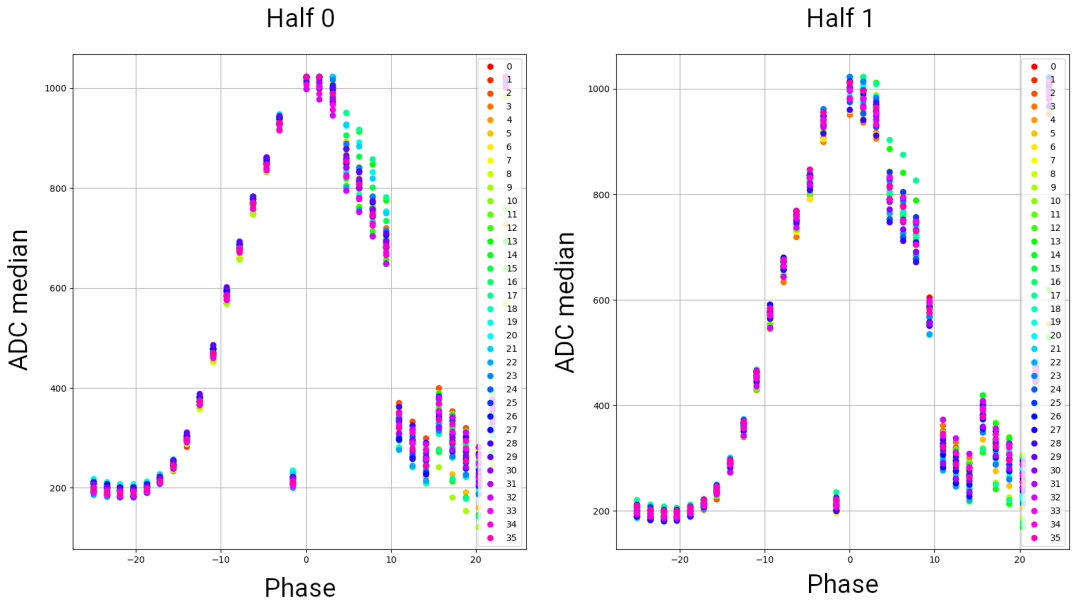
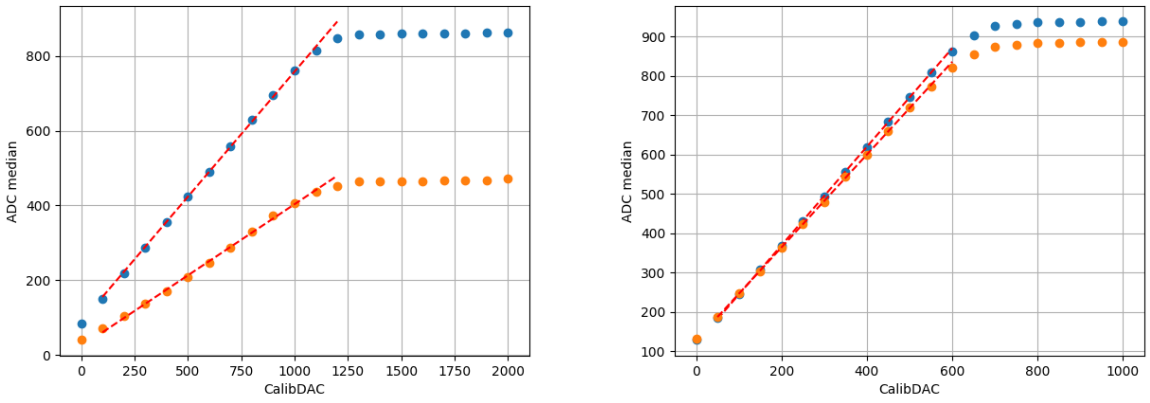


Figure 28: Median ADC values as a function of the Phase parameter when injecting a fixed charge. The two panels show the results in the two halves of one chip.



(a) Non-optimal shaper settings.

(b) Optimal shaper settings.

Figure 29: Injection scans for a given channel in the two halves of a chip. The blue curve corresponds to the first half and the orange to the second. Note the change in the vertical axis range.

done in two ways. Firstly, by varying the global thresholds TOT_{vref} and TOA_{vref} at a set injected charge and recording the threshold for 50% efficiency (e.g. 50 of 100 events activated the TDC), incrementing the amount of charge and repeating the procedure. This is referred to as *global TOT/TOA threshold scan* or *S-curve*. When plotting the global thresholds at 50% efficiency as a function of the amount of injected charge a linear relation is expected. Secondly, the local pedestals are determined in a similar manner to section 4.3.3 after which a trimming can be applied to mitigate inter-channel dispersion. The first testing session of single chips showed inconclusive results of TOA and TOT measurements. In the hexaboard testing sessions a bug in the PCB design was discovered that prevented the study of the TDCs. At the time of writing, the discriminators and their contribution to noise are under investigation.

4.4 Conclusion

In summary, 223 single chips were tested, of which 186 passed the preliminary selection criteria that were set up and subsequently used in the assembly of 21 LD and the very first 12 HD hexaboards. The testing was critical, especially for the HD hexaboards as only 12 "good" PCBs existed at that time and it could not be afforded to waste any boards due to non-working chips being mounted. The outcome was that all boards worked fine after mounting. Furthermore, 54 hexaboards (42 LD and 12 HD) have been tested in two testing campaigns. At the time of writing the testing data are being analyzed and compared across chips and platforms which will yield a list of final rejection criteria and tolerances. Besides the produced raw data needed for characterization, online analysis at the time of testing opened new areas of investigation, e.g. optimization of global shaper parameters, interference caused by discriminator and TDC activity and noise studies on hexaboards and modules. Each measurement campaign helped to improve software and firmware - one example for this is the optimization of the configuration software discussed in section 4.2.2. Finally, a recent effort towards usability led to developments of a server-side web application and a client-side graphical user interface program as interfaces to the developed programs, which the author is involved in.

5 Convolutional Autoencoder Trigger Concept

In the second part of this thesis a concept of trigger primitive generation for the HGCAL detector is presented. The concept is based on the theory of neural networks, a subdomain of artificial intelligence, exploring mathematical structures loosely inspired by the brain. The first section of this chapter introduces the general concept and working principles of neural networks (NN), followed by the presentation of two specific types of NNs, autoencoders and convolutional networks, together with the popular MNIST dataset used as surrogate data. These three ingredients are of central importance to the concept developed in the subsequent part, after which studies on conceptual aspects with respect to the HGCAL environment are presented. This chapter is completed with a discussion in which all results are reviewed in view of their application to the concept.

5.1 Neural network

A neural network is a mathematical structure embedding a functional form. In a process called neural network training, certain network parameters, i.e. coefficients of the functional form, are gradually adjusted such that the network function approximates a desired function, which is usually not known analytically. In principle, any function can be fitted arbitrarily close [45] depending on the network capacity (amount of parameters), its architecture, the training time and amount of available (and relevant) data. Thus, the goal of training is to find a function underlying a problem such that the network is able to predict the correct solution for new data that belongs to the same problem domain, i.e. the network is able to generalize beyond the training data. However, neural networks are not always guaranteed to reach generalization power after training.

Neural networks are composed of neurons (nodes) and weights (edges) that are arranged in consecutive layers (see Fig. 30) such that each layer's neurons receive their inputs from the previous layer's neurons and send their outputs to the successive layer's neurons. Data is fed to the network via the first (input) layer, undergoes transformations in the intermediate (hidden) layers until it arrives at the final (output) layer that can be read out and interpreted. The amount of hidden layers separates shallow neural networks from deep networks. Deep networks can easily span tens or hundreds [26] of hidden layers.

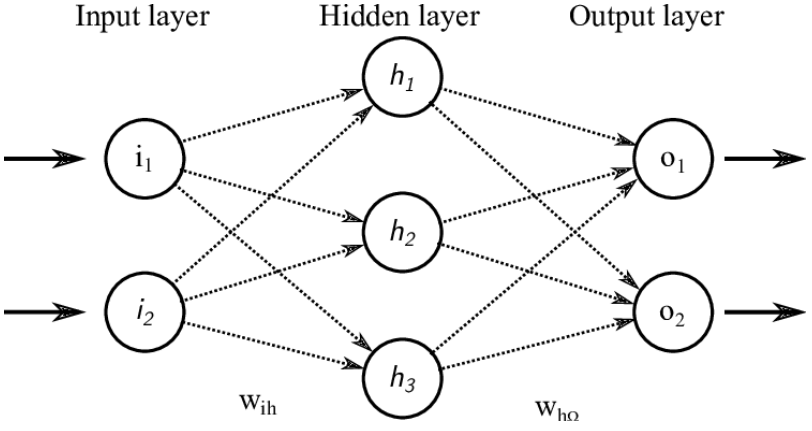


Figure 30: Simple neural network architecture. Figure reproduced from Ref. [38].

Data propagating through a neural network is successively transformed by the neurons of each layer. Each neuron transforms its inputs into a single output value, called *activation*. In a fully connected neural network architecture (Fig. 30) the activation of a single neuron is defined by

$$a(\vec{x}) = \sigma\left(\sum_i w_{ih} x_i + b\right), \tag{12}$$

where $\vec{x}$ represents the inputs (i.e. previous layer's outputs), w_{ih} the connecting weights and b the bias term. The weights and biases are the free parameters of the network, usually initialized to random values before adjustment due to network training. σ is a non-linear function (tanh, sigmoid, ReLU, softmax, etc.) that allows the model to capture non-linear relationships in the input data. Geometrically, a neuron activation transforms the input in three ways:

1. Transform the input linearly: $\sum_i w_{ih} x_i$.
2. Translate to a different position by b.
3. Warp by σ. This effectively maps the input into a non-linear manifold.

Successive non-linear layers allow complex transformations [46]. These networks are referred to as deep neural networks (DNNs). Non-linearity is a decisive criterion for depth since a stack of linear hidden layers, on the other hand, can be collapsed to a single layer containing linear combinations of the former. Therefore, DNNs are much more powerful than shallow (linear) nets.

Since successive hidden layers obtain their inputs from previous layers, $\vec{x}$ can be replaced by $\vec{a}^{l-1}$, where the subscript l refers to the depth index of a layer. Ordering neurons in layers further allows to elevate Eq. 12 to a vector-matrix equation:

$$\vec{a}^{l}(\vec{a}^{l-1}) = \sigma\big(\mathbf{W}^{l} \cdot \vec{a}^{l-1} + \vec{b}^{l}\big), \tag{13}$$

with the weight matrix $\mathbf{W}^l$ between two consecutive layers and $\vec{b}^l$ being a layer's biases. In this form the (forward) propagation of data through the network can be computed rather easily by basic linear algebra subprograms that are highly optimized for matrix multiplications. A graphical processing unit (GPU) can further accelerate the computation by introducing many computational cores that allow for massive parallelization.

As network training is a curve fitting process involving many free parameters, care must be taken to not overfit (or underfit) the model on the training data. Overfitting happens when the parameters are tuned too much. The desired function is for example linear while the network learned a quadratic fit due to the presence of noise in the training data, i.e. the network learns a fit to random fluctuations caused by noise which do not capture the underlying problem. Underfitting is the reverse process of a linear fit to a desired quadratic function resulting from too few network parameters or insufficient training time. To prevent overfitting or underfitting, the training process should be monitored and halted at the right time. Additionally, so-called regularization techniques can be used to prevent overfitting (and also accelerate the training, i.e. the convergence of the fit). More details on networks training can be found in Appendix A.

The training process is guided by a so-called loss (or objective) function $\mathscr{L}$. Since neural networks can be trained for different purposes (classification, regression, reconstruction, data generation, etc.) a multitude of loss functions can be found in the literature. For a given input, the loss function takes the activations of the network's output layer and a corresponding desired output (ground truth) as input and generates a single real number to be minimized. For reconstruction, for example, the mean-squared error function

$$\mathscr{L}_{MSE}(\vec{x}, \vec{y}) = \sum_{n} (x_i - y_i)^2 \tag{14}$$

is often used. If the desired output (truth) has been generated in advance by experts or external sources one speaks of supervised learning. One variant of supervised learning is self-supervised learning in which the ground truth and input data are identical.

In the following sections special network architectures are discussed, that aide in the subsequent presentation of a neural network based trigger primitive generator for the HGCAL detector.

5.1.1 Autoencoder

An autoencoder is a type of neural network which is often used for dimensionality reduction. By introducing a bottleneck in the architecture of the network, a lower dimensional representation (encoding) of input data is achieved, that is subsequently transformed back into the original input space (decoding). The decoded reconstruction of the input is compared to the original input via a reconstruction error, often the mean squared error $MSE = \frac{1}{n}\sum_i^n (x_i - \hat{x}_i)^2$, for n features x_i of a data point and its corresponding reconstruction $\hat{x}_i$. By minimizing the reconstruction error, the network seeks for the best possible representation of the input in its bottleneck (also referred to as latent space).

The simplest type of autoencoder is the *undercomplete* autoencoder, depicted in Fig. 31. The term undercomplete refers to the latent space, which is smaller in dimensions than the input space. Other types of autoencoders, on the other hand, can be *over*complete. They initially contain a latent space larger than the input space that is subsequently reduced by training under regularization constraints that penalize the activation of latent neurons and thus creating the bottleneck. These types include the sparse, contractive, and denoising autoencoders. Variational autoencoders, on the other hand, are a network architecture for data generation. All of these can be further complemented with a deep stack and/or convolutional layers giving rise to two further categories: Deep and convolutional autoencoders. Deep undercomplete convolutional autoencoders are considered in this thesis.

When using deep autoencoders with nonlinear activation functions, the network is able to learn a more powerful generalization of the (linear) dimensionality reduction technique Principle Component Analysis (cf. Appendix B), since nonlinearities allow for warping as discussed in the beginning of Sec. 5.1 which can account for complex relationships in the input data by allowing transformations to more complex manifolds. Furthermore, Ref. [36] reported that depth can exponentially reduce computational cost and the amount of training data needed to learn certain functions. In order to achieve generalization in deep autoencoders overfitting must be prevented: Too much capacity

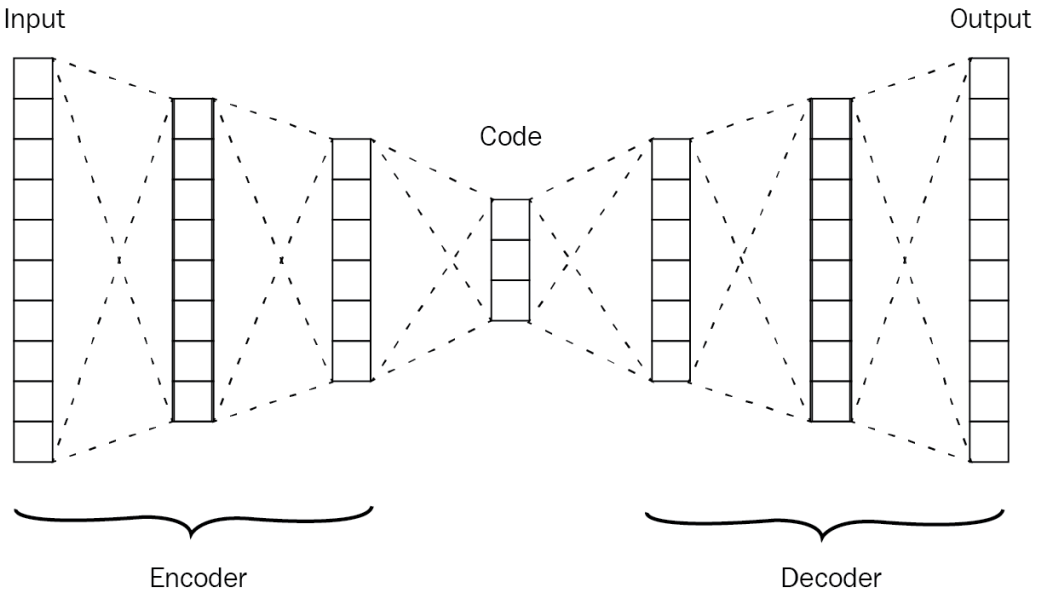


Figure 31: Schematic of a five-hidden-layer deep undercomplete autoencoder network. Figure reproduced from Ref. [43].

(layer depth and width) could lead to learning an identity function by mapping each sample of the training dataset to an index in latent space (a single hidden neuron would be sufficient) from which the input is perfectly reconstructed – a reconstruction error of zero. In this case, the network would extremely overfit on the training data and yield no useful generalization [39]. Therefore, a common practice for training deep autoencoders is *greedy pretraining*, in which the layers of the encoder and decoder parts are trained successively in isolation on each others latent representations before combining them to the deep model.

5.1.2 Convolutional Neural Network

Convolutional neural networks (CNNs) are neural networks containing (a stack of) convolutional layers, which are particularly well-suited when processing image-like data. Real-world images often exhibit an inherent bias – a spatial relationship between the features in a scene. A digit, for example, can be interpreted as a composition of simple features: straight lines, round edges and circles arranged in a certain way. CNNs use (small) filters to learn these features by processing partial areas of the input data at a time. By sliding the filter (also called kernel) across the image distinct features can be learned, and later (at inference time) detected equally well in all parts of the image. This technique makes CNNs invariant to the position where a feature appears in the image.

Figure 32 shows the working principle of the convolutional layer. The pixel values of an image patch are weighted by a filter and afterwards added together to form the output activation of a resulting image (called feature map). Subsequently, the window over the input image is advanced and another weighted sum is calculated to build the second activation of the feature map. This process is repeated until all patches of the image have a corresponding activation on the feature map. The process of "sweeping" a kernel K along an image I is closely related to the convolution operation $K * I$. Since K is not

flipped before sweeping this operation is, more precisely, the cross-correlation operation.

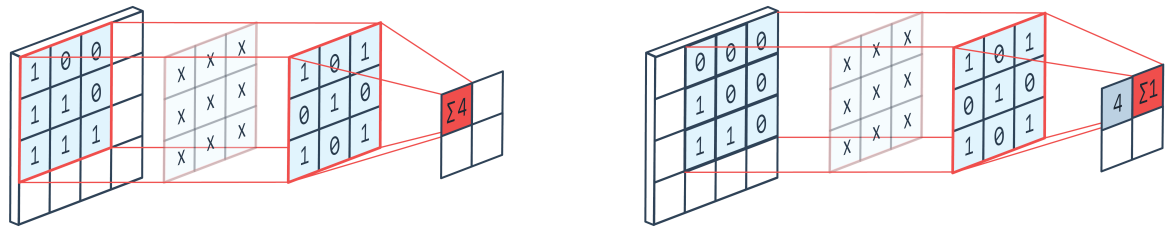


Figure 32: Working principle of convolutional layers. One pixel value of the resulting feature map is a weighted sum of an image patch with the kernel values. Figure reproduced from Ref. [2].

The size of the feature map is determined by three parameters: Filter size, step size (stride), and the amount of additional padding. In some cases where it is desired to preserve the original input image dimensions in the resulting feature map, a stride size of one in x- and one in y-direction can be chosen, while zero padding the image according to the kernel size, which is referred to as *same* padding. *Valid* padding on the other hand does not pad the image and also only allows windows that are fully inside the input image. The output (feature map) dimensions n_{out} of a convolutional layer can thus be calculated from the input dimensions n_{in}, the amount of padding to one side P, the step size (strides) S and the kernel size K as

$$n_{out} = \left\lfloor \frac{n_{in} + 2P - K}{S} \right\rfloor + 1, \tag{15}$$

In CNNs several convolutional layers are used in succession, thus subsequent feature maps take preceding maps as inputs to look for local features. To illustrate the effect, consider a dataset of handwritten digits. While the first convolutional layer learns (low-level) features like edges and shapes, showing their presence as high activations in the corresponding feature map, the second convolutional layer has the ability to combine the presence of these features to find more general (higher-level) features, i.e. to combine for example the activations of a straight line and a circle in distinct parts of the input image to match activations corresponding to the digit *9*. Usually, several filters are learned in parallel for one convolutional layer.

The trainable parameters of convolutional layers are the weights of the kernel together with a bias. Since kernels are usually small and only the weights of the kernel are learned, the number of network parameters is much smaller compared to fully connected networks, which makes CNNs both fast (if parallelized) and memory efficient.

In the last decade, CNNs have been extensively researched, so that they became a de facto standard when dealing with image-like data in neural networks.

5.1.3 MNIST dataset

The MNIST dataset is a collection of 70 000 scanned images of handwritten digits, together with their correct classification. The name of the dataset comes from the fact that it is composed of a **m**odified subset[10] of two datasets collected by the United States' **N**ational **I**nstitute of **S**tandards and **T**echnology. Some of the images contained in the dataset can be seen in Fig. 33. The 70 000 images are split in two parts of 60 000 for training and 10 000 for testing. Training and test set are composed of writing samples from two different groups of people in order to allow testing the network independently sampled data is not used during training [45].

Figure 33: Subset of the MNIST handwritten digit dataset. Figure reproduced from Ref. [7].

The data points of the MNIST dataset are 28×28 greyscale images, in which each pixel has an integer intensity between 0 (black) and 255 (white). The correct classification of training and test data comes in two companion datasets, which are composed of integers between zero and nine that represent the class of the digit shown on the image. In this thesis, the MNIST dataset was used as a surrogate for calorimetric shower data because of the following reasons:

- Granular calorimeter data is in principle image-like, since shower shapes exhibit spatial information.
- Like handwritten digits, shower shapes have semantic meaning, i.e. their energy content, type of shower-originating particle, and its incident location.
- The sensor cell reading is, like greyscale images, single-channel data.
- Interpretations of classification and reconstruction of handwritten digits is closer to the domain of human experience than doing the same on shower data. Thereby, experimentation results can be generated faster and are easier to interpret.
- Also, MNIST is a complete and well-studied dataset that can be used directly.

[10] Size-normalized and centered in a fixed-size image frame [41].

5.2 Concept studies

While the previous sections introduced the basics of neural networks, special architectures and a popular dataset, the following paragraphs discuss how these techniques can be used inside HGCAL's trigger primitive generator that was introduced in Section 3.4. In particular, it will be illustrated how neural networks can be exploited for the purpose of compressing data and extracting relevant information in order to supply the Level-1 trigger system with the necessary information to make a trigger decision.

When bunches of particles cross at the intersection point of the CMS experiment, a subset of these particles are subject to hard collisions, producing a range of other particles in the process, of which a few interact in the end-cap calorimeters. Particles produce electromagnetic or hadronic showers, described in Section 3.1, which traverse the detector. The shapes of these showers often differ for different types of incident particles and energies. In order to generate a trigger decision, i.e. to determine if a bunch crossing contained collisions that are relevant for physics processes the CMS physics program is looking for, the trigger system must distill three types of information from the sensor data for each shower that has been recorded:

1. Type of incident particle that induced a shower, e.g. γ, $e^{\pm}$, π, τ, etc.
2. Energy of the incident particle (inferred from shower energy).
3. Hit position and direction of incident particle.

Since the Level-1 trigger decision must be generated for every BX in a short amount of time the data rate that the L1T algorithms can process is limited. Therefore, sensor data is compressed before processing, with the goal to conserve as much information as possible. Current implementations of compression algorithms in the first of three stages of compression in the HGCAL TPG, the **e**lectronic **con**centrator chip for **t**rigger data (ECON-T), shows varying performance depending on the type and incident position of particles. This situation is summarized in Fig. 34, which shows the triggger rate as a function of the offline threshold[11] for different incident particles. A low trigger rate corresponds to better classification accuracy on the given data. An increase in rate means that an algorithm needs more data in order to reach the same classification accuracy. Thus, there is no single algorithm which performs well in all cases. Furthermore, the large amount of expected pile-up in the high luminosity environment of Run 4 worsens the situation significantly due to overlapping showers and an increase in ambiguities in shower data.

These facts lead to the question of whether there exists a better algorithm which combines the advantages of the above algorithms, yielding good classification performance on all types of incident particles. As discussed in Section 5.1.1, autoencoders are a higher-level generalization of conventional compression algorithms that allow more powerful representations by exploiting more complex relationships in the input data. The idea, introduced in this chapter is to develop a neural network based system spanning the three levels of compression (ECON-T, Stage 1, and Stage 2) of the HGCAL TPG to mitigate the problem of particle type dependent trigger rates. In this thesis, several neural network models have been developed in order to investigate the impact of architectural

[11] The offline threshold is defined as the threshold that achieves 95% trigger efficiency.

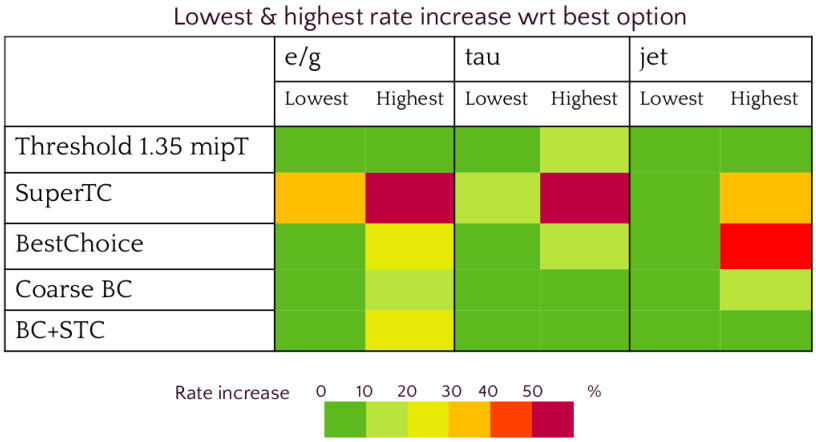


Figure 34: Increase of trigger rate rate as a function of offline threshold for the three different compression algorithms of the ECON-T chip: Threshold, Super Trigger Cell, BestChoice. These algorithms have been presented in Sec. 3.4. Two further variants Coarse BestChoice first applies sorting and then selects the N largest values, BC+STC uses BestChoice in the ECAL and Super Trigger Cells in the HCAL, based on the observation that BestChoice works best for EM showers (small objects) and STC best for jets (larger objects). Table reproduced from Ref. [51].

boundary conditions imposed by the HGCAL TPG to support the development of a proof-of-concept. These boundary conditions are:

- **Elementary encoder unit**: The recently integrated generic encoder block in the ECON-T allows an implementation of a concise hardware-based neural network. As discussed in the previous sections, autoencoders are particularly well suited for finding meaningful encodings that are a generalization of other encoding techniques.

- **Split encoding**: Shower data is distributed among several ROCs – each ROC only sees an excerpt of the information it is meant to preserve. Thus, the question arises as to how effectively encoders can compress parts of a scene while still allowing for the desired information to be recovered by combining the corresponding encodings, i.e. how much does a global encoding differ from the combination of local encodings?

- **Invariance in ϕ**: Physical processes inside the detector are identical for same η, in concentric rings around the beam axis. Thus, showers induced by same particles with same energies are expected to follow the same statistics. By this virtue, encoding units in these rings can be expected to behave very similarly.

- **Partial encoding**: Shower-inducing particles can hit the detector at different positions. Thus, the occupancy of sensors can look vastly different from the viewpoint of the encoder units. In order to account for this, the units must be trained on images that exhibit this randomness in location.

- **Trigger primitive information**: As discussed in Sec. 3.1 the longitudinal and lateral extent, as well as the internal structure of a shower contain three relevant types of information: energy, particle type, and incident position. This is the primary information that is desired to be preserved.

- **Information bottleneck**: Because of the timing constraint for L1T decisions the sensor data needs to be reduced by a factor of $5 \times 30 \times 8 = 1200$ in the three[12] compression stages of the HGCAL trigger primitive generator (cf. Sec. 3.4). Thus, to make a trade-off decisions, a relationship between compression ratio and information capacity would be useful.

Based on these constraints a neural network architecture was developed, schematically depicted in Fig. 35. It shows three stages of feed-forward encoder networks. The corresponding decoders will be implemented in software for training and monitoring, thereby allowing for sparser hardware implementations of just the encoder part. The network as a whole is trained such that relevant information is maximally preserved. In this thesis the key conceptual ideas are explored by using a smaller scale architecture on the toy problem of recognizing handwritten digits from the MNIST dataset. In the following, these studies will be presented togehter with their results.

5.2.1 Elementary encoder unit

As shown in Fig. 35, the concept is built on multiple elementary encoder units successively propagating encoded latent representations through a cascade of encoding stages, starting from calorimetric (trigger cell) data and ending with trigger primitives. The following studies focus on the behavior of encoders in a single stage (ECON-T) by considering all of the aforementioned boundary conditions. The cascading behavior of multiple consecutive stages is subject of future studies. The subject of this section is the basic building block of the concept, the elementary encoding unit, and its reconstruction performance under ideal conditions.

Calorimetric data is inherently image-like – sensor cell readings can be interpreted as pixels (of energy intensity) forming an image of a shower. As discussed in Sec. 3.1, the length, width, and overall shape of showers carries macroscopic information that allows to infer information about its inducing particle. Thus, the encoder unit should be capable of extracting structural information from the shower image. Sec. 5.1.2 highlighted that convolutional neural networks are the usual choice when it comes to such data due to their ability to extract a hierarchy of features by correlating features in a layer with themselves (inductive bias).

Additionally, each encoding stage has the task to compress the calorimetric data by a certain ratio without losing too much information. As discussed in Sec. 5.1.1, the goal of PCA is to preserve dimensions (features) in the data that show the most variance, thus carry the most information and that autoencoders are a more powerful generalization of PCA. Thus, an autoencoder architecture is the preferred choice for the encoder networks. Furthermore, since the compression stages have a fixed amount of incoming and outgoing data links, the implementation of the networks must be undercomplete.

Deep convolutional layers and an undercomplete autoencoder architecture can be combined into a deep convolutional and undercomplete autoencoder, which will be the model

[12]There are two more compression stages in the summing of sensor data to trigger cells and in the change of datatype from 10 to 7 bit. However, these stages are finalised in the design and are therefore not further considered in this discussion.

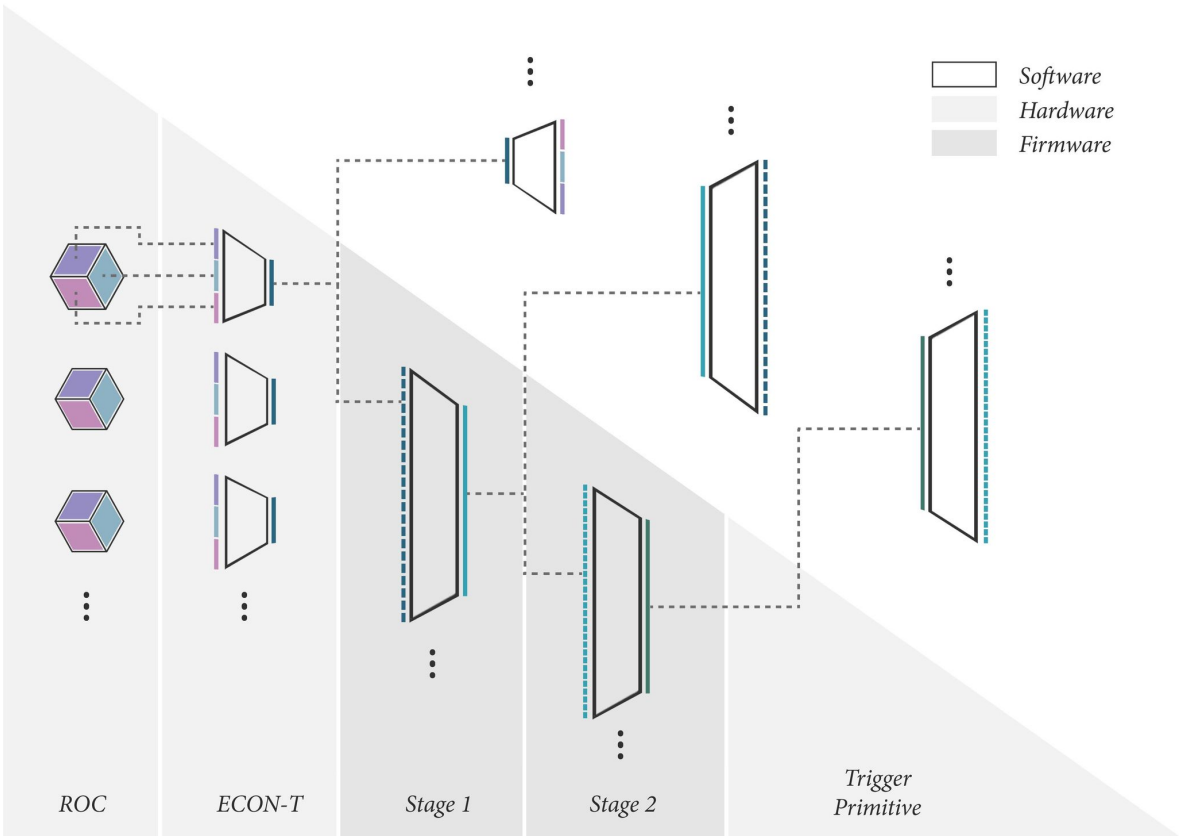


Figure 35: Illustration of a neural network based HGCAL trigger primitive generator. Trigger cell data from three ROCs is sent to one ECON-T, where the data is concatenated and encoded to the latent representation of a module. This latent space is sent together with 20 other ECON-T encodings to the Stage-1 encoder, which, again, concatenates its inputs and encodes it. Stage-1 encodings are sent together with 29 other Stage-1 encodings to Stage-2, which concatenates its inputs and encodes it. The latent space of Stage-2 is the trigger primitive. ROC and ECON-T compression will be implemented in hardware, while Stage-1 and Stage-2 is implemented in firmware. Corresponding decoders will be implemented in software in order to train the network and monitor the outputs of each stage during inference.

under investigation in this section. Keeping the real application in the ECON-T in mind, the network depth has been kept at a reasonable size to ensure implementation in limited (ASIC hardware) resources. An illustration of the architecture can be seen in Fig. 36. In order to compare the performance of neural network architectures after introducing different constraints, the saturated reconstruction (MSE) loss is used as a measure of reconstruction performance and visual fidelity of the reconstructed image.

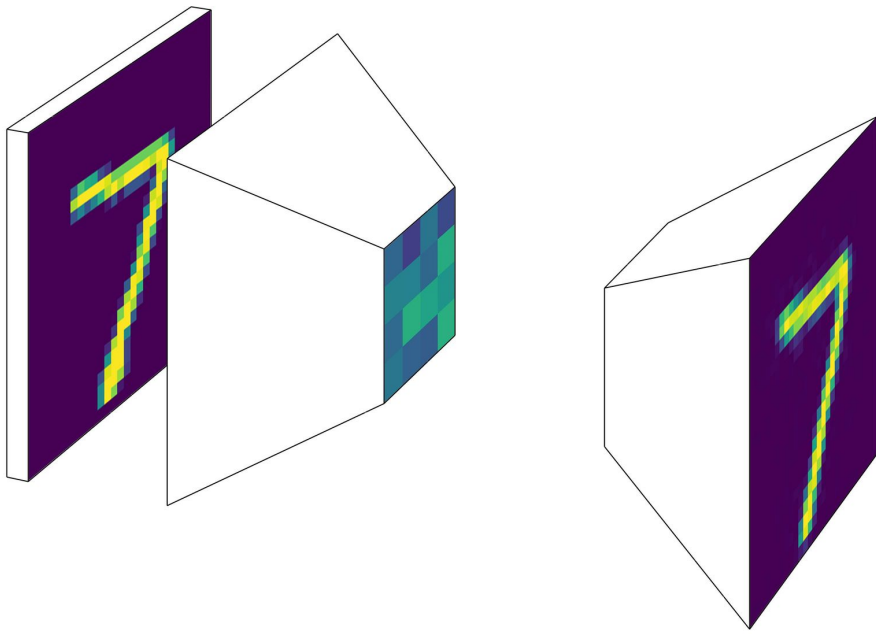


Figure 36: Illustration of convolutional autoencoder model. The encoder takes a 28×28 input image, produces a 4×4 latent space from which the decoder tries to reproduce the original image.

The basic convolutional autoencoder consists of a stack of convolutional layers, which reduce the input image dimensions with each layer, according to Eq. 15, until a sufficiently small output feature map is produced. As it is typical for convolutional layers to increase the amount of filters with depth, the resulting feature maps need to be combined at the end of the encoder in order to form the bottleneck. There are two ways to achieve this:

- By using a convolutional layer at the end of the encoder that uses a single filter and thereby collapses all preceding feature maps into a single output map, which can either be used directly as a two-dimensional latent space or flattened to one dimension. The resulting encoder structure is fully convolutional. Therefore, this model is referred to as fully convolutional autoencoder (FCAE).

- By using a fully connected (dense) layer at the end of the encoder which connects all pixels from all preceding feature maps to a one-dimensional layer of arbitrary size in order to build the bottleneck. Due to the dense layer this model produces a different latent space and is thus referred to as convolutional autoencoder (CAE). This architecture is similar to what has been so far implemented in the ECON-T.

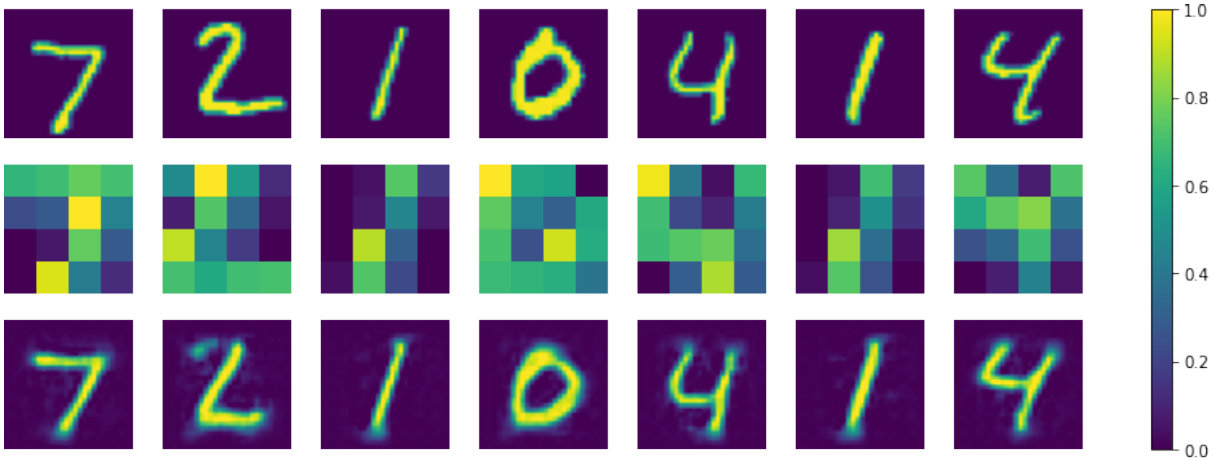
After the bottleneck, transposed convolutional layers expand the representation back into the original input space in the decoder part of the autoencoder.

While the FCAE, trained on reconstruction, produces latent representations that resemble pixelized versions of the input images, the latent space of the CAE has a more code-like nature, as seen in Fig. 37. The reason for the difference in representations can be explained by expressing the dense layer by an equivalent convolutional layer with a kernel size matching the feature map dimensions of the last encoder (convolutional) layer, no strides and an amount of channels (filters) that is equivalent to the amount of units in the dense layer. Per channel, a one pixel feature map is produced by using the same filter on all input feature maps, yielding a one-dimensional output in the channel dimension which is equivalent to the output of a dense layer. By this transformation the effect of the dense layer on the latent space can be seen more clearly – by scanning all feature maps for the same pattern, the dense (or dense-equivalent) layer looks for global features (common to all feature maps) and yields activations based on their combined presence, whereas the point-like convolutional layer combines the presence of all local features (same pixels along the channel axis) into the latent representation.

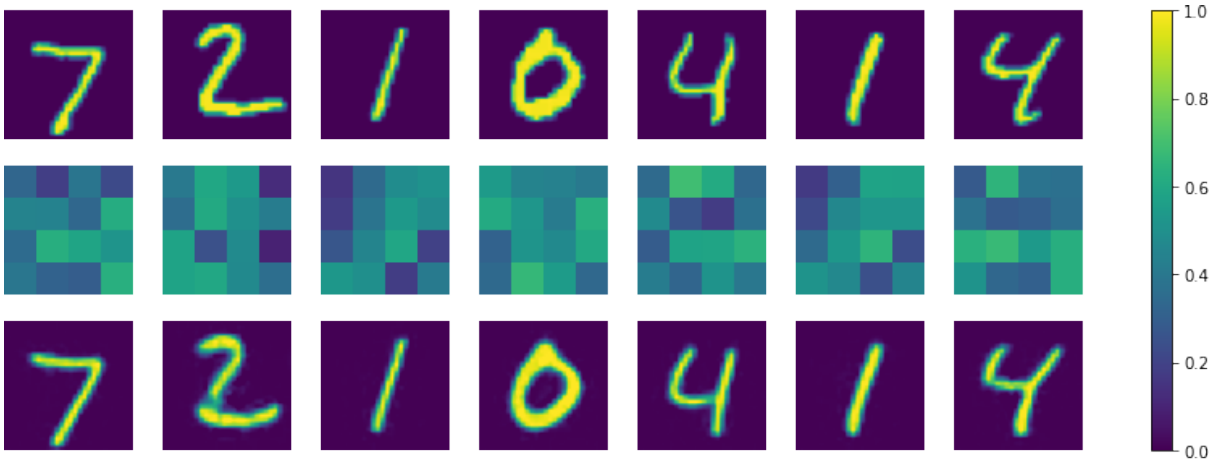
Also, the network capacity of the CAE is larger than the FCAE's, which allows modeling of more complex functions. This is confirmed by observing the reconstruction error, which is twice as large for the FCAE than when using a dense layer in the encoder and decoder. The dense layer of the decoder is not necessary to illustrate the working principle described here, however it drastically improves performance due to increased network capacity. Because the decoder in the concept is implemented in software and not subject to any size constraint, it was chosen much larger than the encoder's dense layer. Because of the superior performance of the autoencoder including dense layers and the similarity to the current implementation in the ECON-T, this architecture (CAE) was chosen as a baseline for all further studies.

A further advantage of using a dense layer in the latent space is that its dimensions can be easily changed, whereas the fully convolutional model requires significant changes to the architecture in order to produce a change in dimensions. By varying the latent dimensions while keeping the rest of the architecture unchanged, a relationship between reconstruction quality and compression ratio can be established. As reconstruction quality is not defined quantitatively, the saturated reconstruction loss on the validation set was chosen as a representative measure for information loss due to the bottleneck. An example of the progression of this loss during training for latent dimension of 16 can be seen in Fig. 38. For all latent dimensions the validation loss converges early on. However, to guarantee saturation, the best value after 1000 epochs was selected for comparison.

For different bottleneck dimensions, the loss on the validation set saturates at different values. Figure 39 shows saturated validation losses as function of the inverse compression ratio $1/C$, bottleneck size divided by the 28×28 input dimensions of the image. If the validation loss is near zero, the reconstruction is perfect. The loss seems to approach zero already before an inverse ratio of 1 (encoder-decoder identity mapping) is achieved. The reason for this is that the handwritten digits only occupy a small subset of the 28×28 input space, thus a representation of only ~ 160 dimensions (corresponding to $1/C \approx 0.2$) is sufficient to represent all information of the digits. In other words, the current rep-



(a)



(b)

Figure 37: Examples of input images (first row), their latent space representation (middle row) and their reconstruction from the latent space (last row) for: (a) the fully convolutional autoencoder (FCAE) and (b) the convolutional autoencoder (CAE). The training progression of (b) can be seen in Fig. 38.

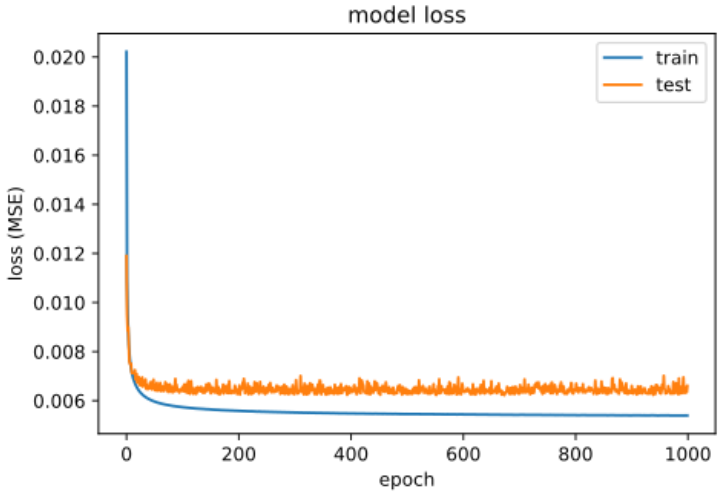


Figure 38: Progression of the MSE loss on the train set (blue) and the test/validation set (orange) for the CAE with 16 latent dimensions and 1000 training epochs.

resentation of digits in 28×28 images is a rather inefficient way to store the contained information. For inverse ratios below 0.2 the network starts to lose information. However, with $1/C \approx 0.02$ (corresponding to 16 latent dimensions) the reconstruction still yields a high visual fidelity. This ratio has been consistently used in the following studies to achieve comparability.

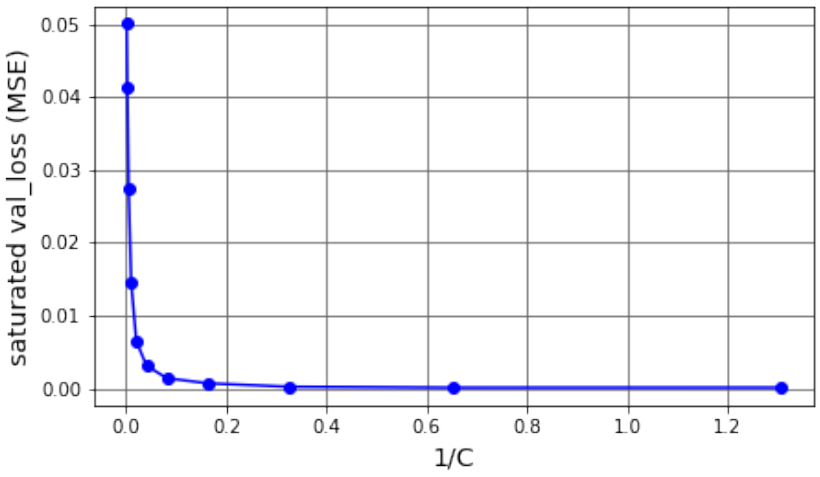


Figure 39: Reconstruction loss after 1000 training epochs as a function of inverse compression ratio of the CAE. Each data point corresponds to a five-hidden-layer deep undercomplete convolutional autoencoder trained on the MNIST handwritten digit dataset.

5.2.2 Split encoding

Shower data is usually distributed over several sensors in the calorimeter. Therefore, the encoder units only see an excerpt of the shower at any given time. Because of this, the global shower information which is desired to be preserved needs to be recovered from multiple encodings of partial information. This study investigates the feasibility and performance of such an encoding scheme. In order to compare visual fidelity with the CAE baseline model, only the structure of the encoder units was altered, while the decoder architecture was kept unchanged. This way, any changes in performance can be attributed directly to changes made to the encoder.

To achieve a limited field of view, the 28×28 input images have been split in four 14×14 patches (quadrants), which are fed in parallel to four independent encoding units that each computes a latent space. These four latent spaces are concatenated and fed to one decoder. An illustration of the architecture, referred to as *4-Split* (4S) can be seen in Fig. 40. The encoders are each composed of the same convolutional layers as the CAE encoder. However, the terminating dense layer has only four units, such that the four units together produce a latent space of 16 dimensions, equivalent to the CAE discussed in the previous section. Therefore, the dense layer at the end of each encoding unit cannot correlate global features of the entire image, but rather correlates features per quadrant. This scheme achieves the previously mentioned encoding of partial information. On the other hand, due to its individual set of network parameters, each unit has the capability to learn its own, quadrant-specific filters. The total capacity of the four units together is about 33% larger than the encoder of the baseline model (cf. Table 2). Thus, the question arises: Is the performance of the 4-Split model better because of a larger overall capacity or because of smaller dense layers and per quadrant, instead of global, correlations?

As it turns out, the 4-Split architecture (4S) performs slightly worse on the reconstruction task, despite its larger capacity. The loss saturates at a value about 20% larger than the baseline model. Examples of reconstructions and corresponding latent spaces can be seen in Fig. 41. The fully convolutional version of the 4-Split (F4S), on the other hand, performs significantly worse than the fully convolutional model (FCAE). Fig. 41a shows that the encoder splitting seems to prevent the network from building a pixelized latent representation, as seen in in the examples of the fully convolutional autoencoder (Fig 37a).

This study showed that it is feasible, without much loss in reconstruction performance to encode image patches which contain only partial information of an image, from which the global information was recovered after decoding their concatenation.

5.2.3 Invariance in ϕ

Due to the symmetry of the CMS detector, physics processes in areas of concentric rings, of $\theta = [0, 2\pi]$, around the beam axis have no preferred direction, i.e. they are invariant in ϕ. Therefore, encoder units positioned in these regions should be expected to also behave the same. For neural networks that means identical architecture and identical network parameters, which guarantees identical results at inference time. This study explores how identical units can be implemented in practice and how they compare in performance to the 4-Split with individual encoder units and to the baseline model (CAE).

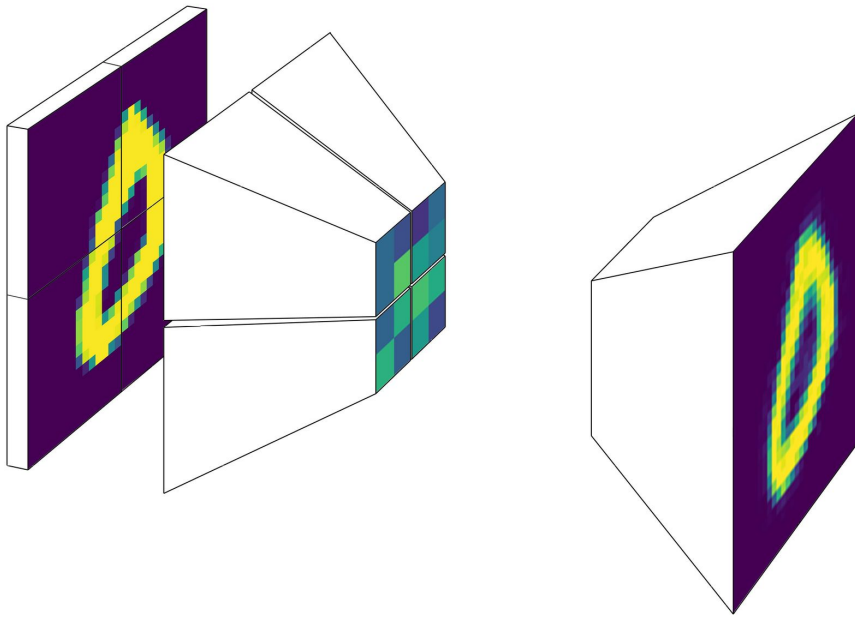
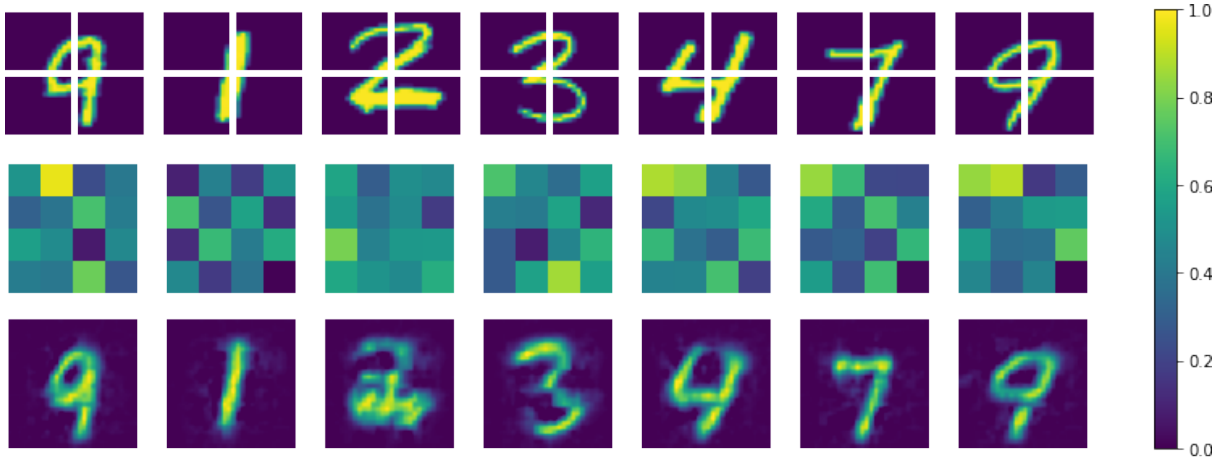


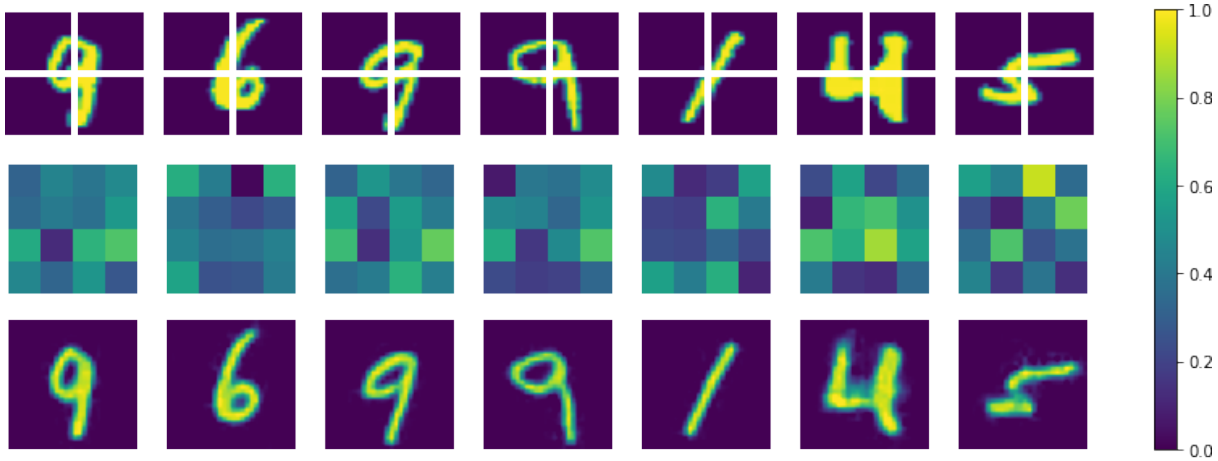
Figure 40: Illustration of 4-Split model. Four similar encoding units with individual parameters produce four 2×2 latent spaces, which are concatenated and fed to the same decoder architecture from the baseline model described in Sec. 5.2.1.

Identical units can be realized by sharing (tying) the parameters of architecturally identical encoder units. Then, the gradients of all encoder units are implicitly averaged and a shared set of parameters is updated during training time. This technique not only requires (in theory) less memory to store the parameters, but also greatly simplifies the training of the 4-Split model (4S). Instead of training four individual units in parallel, only one unit is trained sequentially on four different inputs, such that the generated latent spaces are first concatenated and then further directed to the decoder. Fig 42 shows the latent space activations of four encoder units (a) with individual and (b) with shared weights exposed to the same 14×14 input image fragments. As one can see, the latent spaces of the encoder units in (b) are identical as expected.

The reconstruction performance of the 4-Split with shared weights is slightly worse than the model with individual weights. The loss saturates at a value $\sim 8\%$ larger, which can be attributed to the much larger capacity of the model with individual parameters and its capability to learn quadrant-specific filters, whereas the one with shared weights can only draw from a common set of filters for all quadrants. However, it is surprising that the loss is not much worse, considering that the encoders with shared weights contain about three times fewer network parameters (cf. Tab. 2) than the ones with individual weights. Fig. 43 shows examples of inputs images, their corresponding latent spaces and reconstructions. In comparison to Fig. 41b, the visual fidelity does not look much worse. Only the reconstruction of the digit 5 in the last column exhibits a significantly lower level of detail.

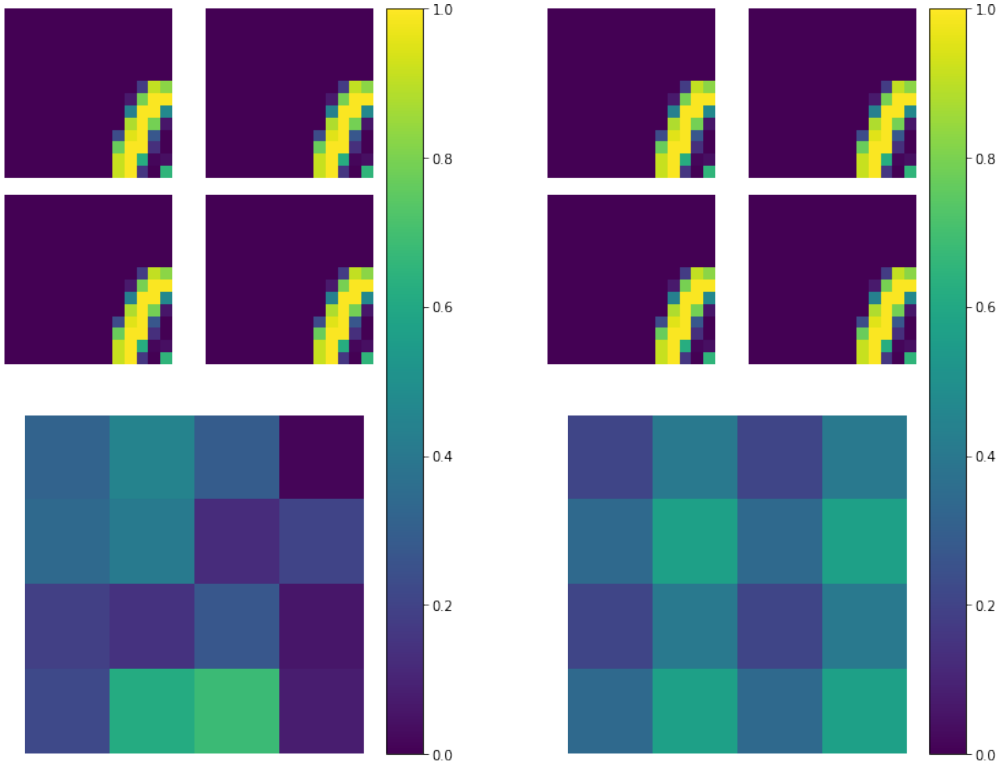


(a)



(b)

Figure 41: Examples of input images (first row), their latent space representation (middle row) and their reconstruction from latent space (last row) for: (a) Fully convolutional 4-Split (F4S) and (b) 4-Split (4S) models.



(a) (b)

Figure 42: Latent space activation for identical image fragments for 4-split with (a) individual weights and (b) shared weights.

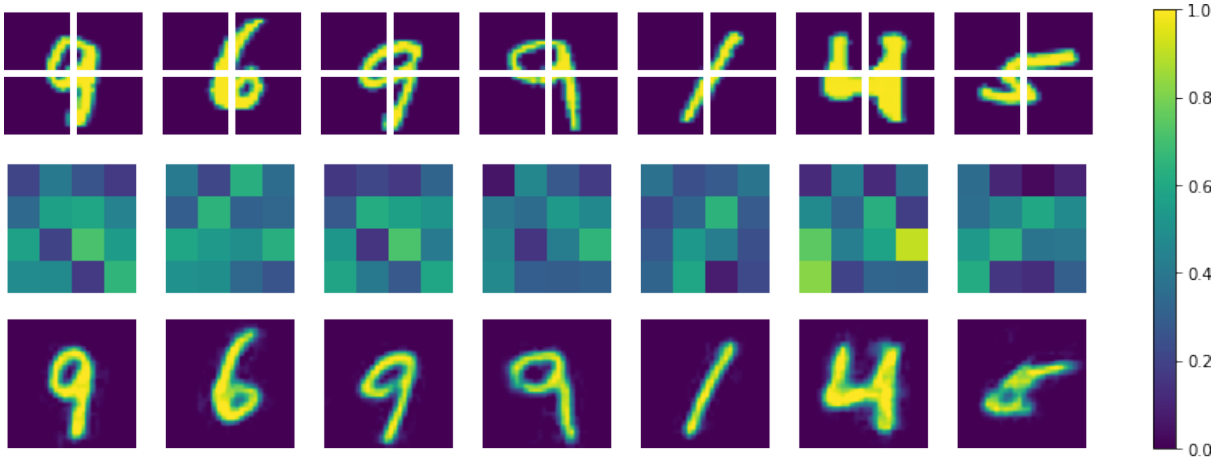


Figure 43: Examples of input images (first row), their latent space representation (middle row) and their reconstruction from latent space (last row) for the 4-Split model with shared weights.

This study showed how identical encoding units, exposed to parts of an image, can be implemented. These encoding units perform a little bit (8%) worse than encoding units with individual parameters on the reconstruction task. However, the decrease in performance is rather insignificant compared to the amount of network parameters necessary for the shared-parameter-model (3 times lower).

5.2.4 Partial encoding

In the last two sections the construction of a network architecture was discussed that encodes quadrants of an input image which are subsequently combined in order to recover the information of the full image. However, in the detector, shower data is not always distributed in equal parts on neighboring sensors. Instead, there are many possible splittings of shower data between sensors, and therefore many more encodings that need to be considered. Furthermore, since the encoder units share a common set of weights, each unit must know about the encodings for all encoders at the same time. This study focuses on simulating different shower positions, including empty sensors to test the performance under conditions that are more realistic than previously discussed.

In order to simulate different sensor occupancy configurations, the 28×28 digit is embedded in a larger 56×56 empty frame, and translated in the x- and y-axes in a range of $0 \rightarrow 28$, such that the digit can reach all possible positions inside the larger frame, as illustrated in Fig 44. In this way, the full information is guaranteed to be inside the input image.

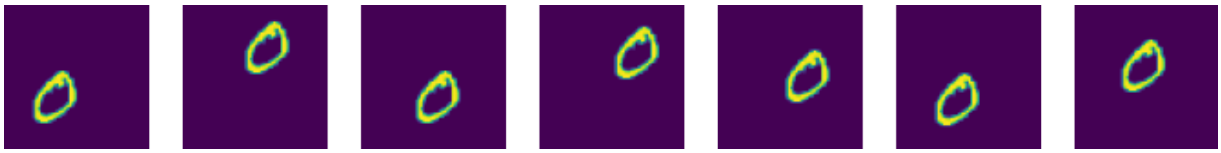


Figure 44: Example of 28×28 digits randomly positioned inside a larger 56×56 image of zeros. The digit is translated in integer steps in the range $0 \rightarrow 28$ for x- and y-axes.

The 4-Split architecture with shared weights is extended to include four times more encoder units (16 in total) and the decoder is scaled up accordingly to produce 56×56 reconstructions. Each encoder covers 1/16 of the full input image and an illustration of the model architecture, referred to as 16-Split (16S), can be seen in Fig 45.

If the 16-Split is trained without shifting the digit in the larger input image, the model is very similar to the 4-Split. Due to the random positioning of the digits, the shared set of parameters needs to account for more variation in the input data, i.e. there are more 14×14 image framgments in comparison to a splitting by quadrants.

To compare the 16-Split to the other models, the reconstruction loss is evaluated only on the part of the output image which contains the 28×28 digit. Otherwise, the mean of the squared errors between output pixels and truth would pull the loss down, since most of the activations are zero. Evaluating the 16-Split's reconstruction performance leads to an increase larger than a factor of two in reconstruction loss with respect to the 4-Split

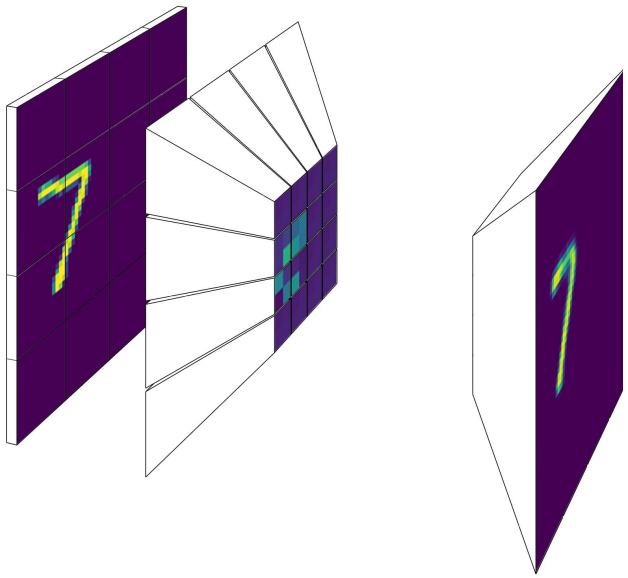


Figure 45: Illustration of the 16-Split model. The 16 encoder units have the same architecture as the 4-Split encoders, generating a four unit latent space from a 14×14 input image patch. Thus, the decoder input latent space consists of 64 (16×4) values, from which the original 56×56 image is reconstructed. The inverse compression ratio of the 16-Split is the same (0.02) as those of the 4-Split (4S) and the baseline (CAE) model.

with shared weights and to a 2.7× increase in comparison to the baseline (CAE) model. Examples of reconstructed images and their corresponding latent space representation can be seen in Fig. 46. Since four times more encoder units are used, the latent space is composed of 16×4 encoder output latent spaces. It can be seen that most of the latent space activations are near 0, while only the encoding units that see parts of the digit in their input show an activation.

This study showed a significant decrease in reconstruction performance when the encoders need to account for different digit positions and thereby a larger variety of input patches containing part of the same global information. This is summarized in Tab. 2 which shows the validation losses for all models so far discussed after 10 000 epochs of training. The 16S model was not trained for so long as it has a much larger amount of parameters and presumably requires a more careful tuning of the training process. This could be an interesting consideration for future studies.

5.2.5 Trigger primitive information

Reconstruction quality (visual fidelity), which has been discussed so far, is not the primary information needed for trigger primitives – particle type, position, and energy, on the other hand, are. Therefore, this study discusses additions to the 16-Split architecture introduced in the last section which allow to train on objectives other than reconstruction.

First, in order to classify which digit is in the image, given the combined latent spaces of the 16 encoder units, a new decoder architecture was created, referred to as classification

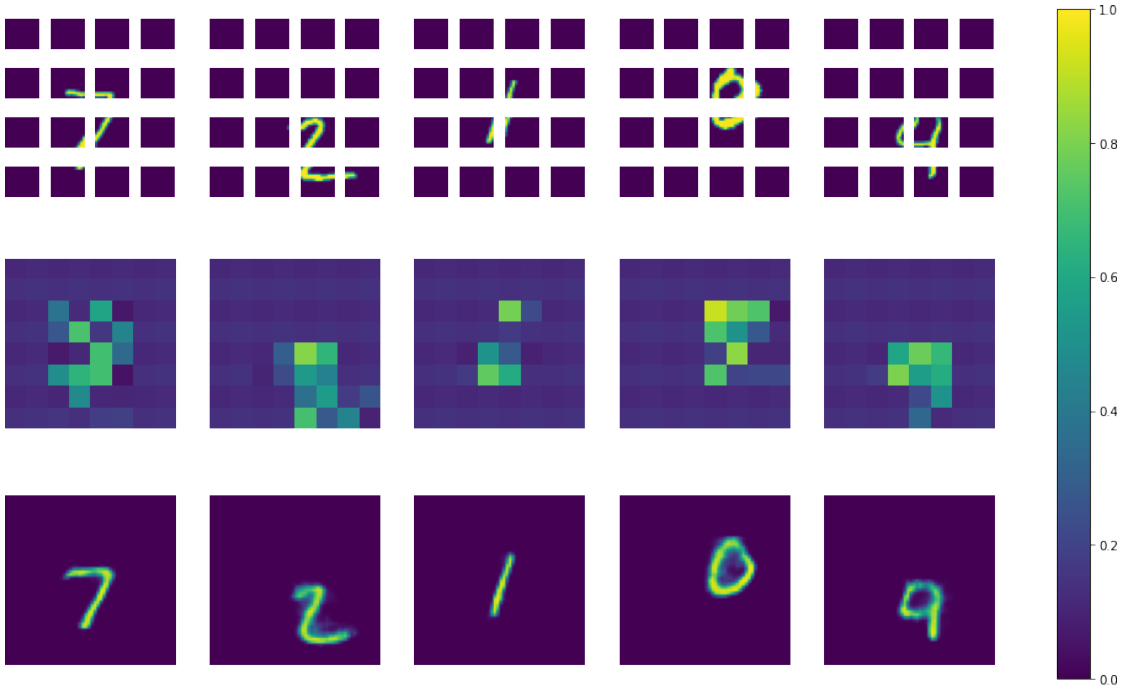


Figure 46: Examples of input images (first row), their latent space representation (middle row) and their reconstruction from latent space (last row) for the 16-Split model with shared weights.

Table 2: Training summary for different models, including number of trainable parameters for the entire model (encoder model), the training time per epoch and the saturated reconstruction loss (MSE) on the validation set after 10 000 epochs. The corresponding model architectures are detailed in Appendix C and their training loss curves are shown in Appendix D.

Model	**Input shape**	**Parameters**	**Time per epoch**	**MSE (sat.)**
FCAE	$1 \times 28 \times 28$	39 251 (19 841)	7s	0.0156
CAE		99 097 (69 008)		0.0073
F4S	$4 \times 14 \times 14$	96 982 (77 572)	9-11s	0.0311
F4S (shared)		38 803 (19 393)		0.0332
4S		121 753 (91 664)		0.0088
4S (shared)		53 005 (22 916)		0.0092
16S	$16 \times 14 \times 14$	492 001 (366 656)	*	*
16S (shared)		148 261 (22 916)	24-26s	0.0226

head (16S-C). The classification head contains four dense layers with 256, 128, 64, and 10 nodes, respectively. The last layer's activations are transformed by a softmax function into a probability distribution. During training, this probability distribution is compared to a one-hot vector encoding the true class corresponding to an input image using the categorical cross-entropy (CCE) loss function. In this way, the encoders and classification head are trained to minimize the difference of the last layer's activation to the true label. Fig. 47 depicts latent spaces and probability distribution outputs corresponding to different input image examples, where the class labels are distributed on the x-axis, together with the true label highlighted in blue. The representations of the encodings show a prominent checkerboard pattern. These patterns are an artifact of the strided convolutions of the encoders, which could be avoided by using a stride of one and max-pooling layers in between. Although this pattern is also visible in Fig. 46, it is noteworthy that for classification it is much more prominent, which could be an indication of a more concise latent space in comparison to the space trained on the reconstruction task. In other words, for the classification task only part of the possible capacity of the latent space is used.

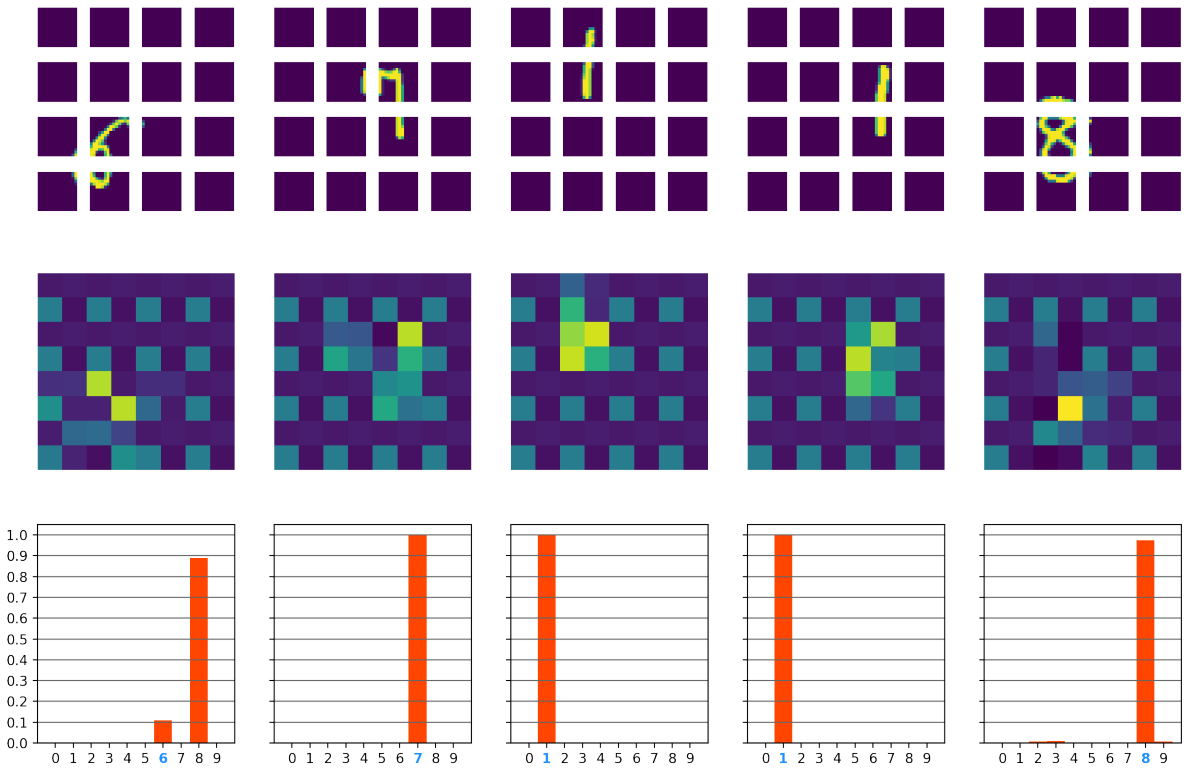


Figure 47: Examples of input images (first row), their latent space representation (middle row) and their the predicted classification (red) from latent space compared to the true class (bold and blue) of a digit (last row) for the 16-Split with classification head (16S-C).

Second, for the purpose of regressing the position of the incident particle, given the digit image and the true position, another decoder architecture, referred to as the regression head (16S-R), was created. The regression head is very similar to the classification head – it also contains four dense layers of which the first three have the same dimensions as the 16S-C. The fourth dense layer, however, contains only two units with linear activation; for a given example with a corresponding truth consisting of a (x,y)-coordinate pair is compared to the two linear output activations via the mean absolute percentage error (MAPE). For each image the truth was defined as the center of a digit. Resulting latent spaces and regression outputs for certain example images can be seen in Fig. 48. The quantity Δr is the euclidean distance $\Delta r = \sqrt{\Delta x^2 + \Delta y^2}$ between the regression output (red circles) and the truth (blue circles). The same checkerboard pattern shows up. It seems that for the regression task even less latent capacity is needed – besides a larger contrast between the units, even units in the area of encoding seem to be mostly unused in the codes.

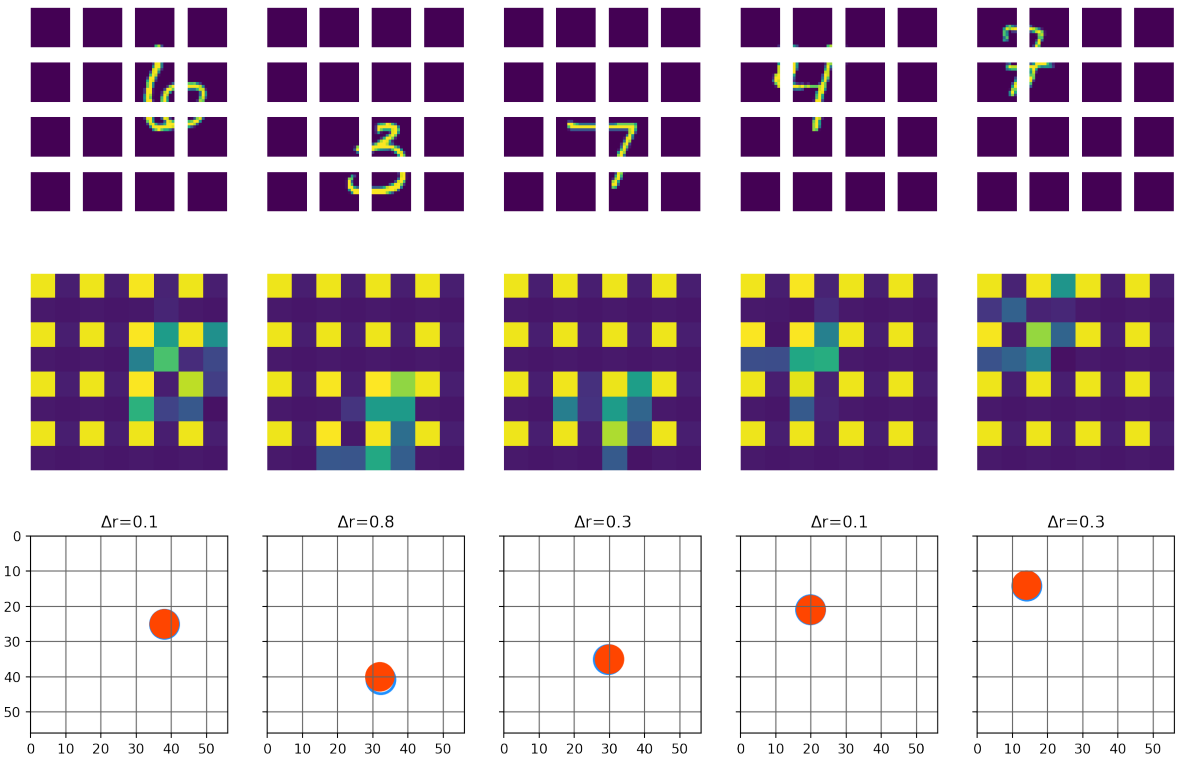


Figure 48: Examples of input images (first row), their latent space representation (middle row) and their the predicted regression of position (red circle) from latent space compared to the true position (blue circumference) of the center of a digit (last row) for the 16-Split with regression head (16S-R). Above the prediction, the euclidean distance, Δr, between truth and prediction is given.

After having seen how classification and regression can be realized within the same architecture, the next question is: How can one train the encoders in order to produce a latent representation that can be used by all decoder heads to recover different types of information from the same input image? By observing the latent representations of the classification and regression tasks one could assume that unused latent space capacity can be used in order to find such an encoding. Fig. 49 shows the latent space representation for encoding units trained on classification and regression at the same time. One can see that the contrast of the checkerboard pattern is fainter and that the latent codes seem to use more space than seen in Figs. 48 and 47. This supports the hypothesis that the pattern is related to used latent space capacity. Both heads have been trained at the same time with equal weight on the loss function. As it turns out, training with several objectives always involves a trade-off, with best possible performance only in case a head is trained in isolation. This can be seen more clearly in Fig. 50. It illustrates how the saturated loss for a certain head varies with its loss weight LW on the global loss $\mathscr{L}_{glob} = LW_{class} \cdot \mathscr{L}_{CCE} + LW_{reg} \cdot \mathscr{L}_{MAPE}$ used to update the network parameters. Table 3 shows an overview of training reconstruction (MSE), classification (CCE) and regression (MAPE) heads in different order, fixing the encoder weights and training the other remaining heads on the (fixed) encoder. It shows that the best performance can be achieved when the encoder is trained for a single task. The second best performance can be observed when all heads are trained simultaneously. However, the loss weights, which specify the impact of a certain task on the total loss, are hyper-parameters that would need to be fine-tuned in order to maximize the performance. Furthermore, this study yielded several other insights into the training processes:

- **Row 1 vs. row 5** shows a significant decrease in classification accuracy when the classifier is trained on the encoder pretrained for reconstruction and regression. If the encoder is pretrained on reconstruction alone, the classifier can be trained (together with the regressor) to yield significantly better performance. The representations for reconstruction must be more favorable for classification than the representations for regression and reconstruction together.

- **Row 1 vs. row 6** compares encoders pretrained on reconstruction to encoders pretrained on reconstruction and classification. The latter seems to generate representations that are unfavorable for the regressor to learn a meaningful mapping. This is a similar behavior to the first point, as in both cases the task trained on pretraining with two other tasks looks significantly worse. However, this cannot be generalized since row four shows no significantly worse reconstruction performance when the encoder was pretrained on classification and regression.

- **Row 3 vs. row 7** shows a surprising improvement of MAPE loss when all objectives are trained together. During training it was noteworthy that the MAPE loss was largely fluctuating. It might be that the model was not trained long enough and that the MAPE value would, after more training, reach a better value for the training on single task, which would be expected. It can also be seen in Tab. 4 that the MAPE loss does not always behave as expected. Overall, the training of all heads at the same time yields the best performance.

This study showed the performance on the tasks relevant for trigger primitive generation considering the most significant architectural constraints imposed by the real-world system. Noise (similar to pile-up) has not been discussed so far, which would be interesting

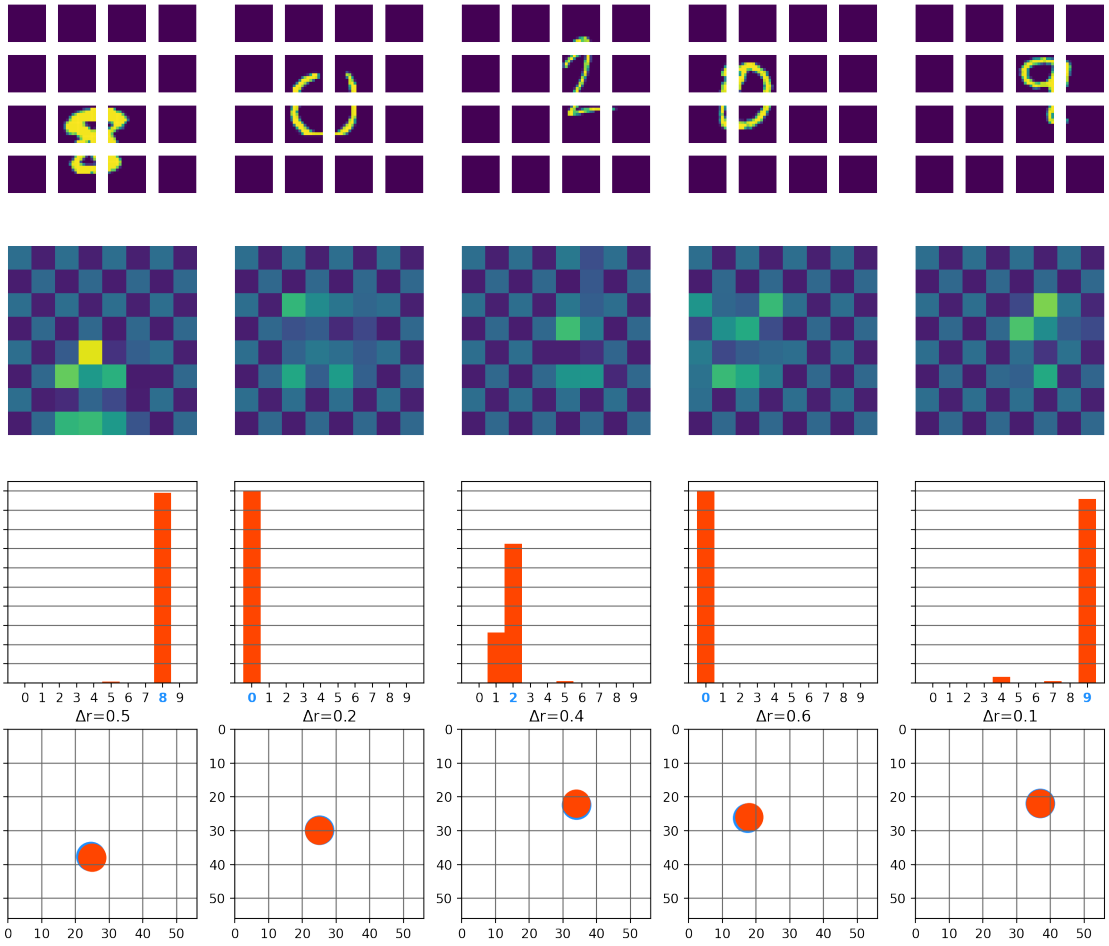


Figure 49: Examples for 16-Split trained with classification and regression heads at the same time. Examples (first row), latent spaces (second row), classification probability distribution (third row), and regression results (last row).

Table 3: Comparison of performance for different ways to train the 16-Split model for three different types of objectives: reconstruction (MSE loss), classification (CCE loss, binary accuracy metric), and regression (MAPE loss). The tasks without brackets in a row have been trained first, after which the encoder parameters were not changed anymore during training. The task in brackets comes from training the decoder on this (fixed) encoder. When more than one objective is trained at the same time, relative loss weights LW had to be specified which have been chosen somewhat arbitrarily, and could probably be tuned to get better performance.

Encoder loss weights			**Decoder loss performance**		
MSE	**CCE**	**MAPE**	**MSE**	**CCE, Acc.**	**MAPE**
1	(1)	(1)	0.0226	(0.5811, 83.29%)	(1.5123)
(1)	1	(1)	(0.0494)	0.3248, 92.66%	(2.9215)
(1)	(1)	1	(0.0352)	(0.8090, 73.13%)	1.2721
(1)	**1**	**1**	**(0.0302)**	**0.3986, 88.49%**	**1.5217**
1	(1)	0.01	0.0269	(2.3115, 9.4%)	1.2870
1	0.05	(1)	0.0249	0.4767, 88.59%	(27.4358)
1	0.05	0.1	0.0340	0.5136, 83.18%	1.2143

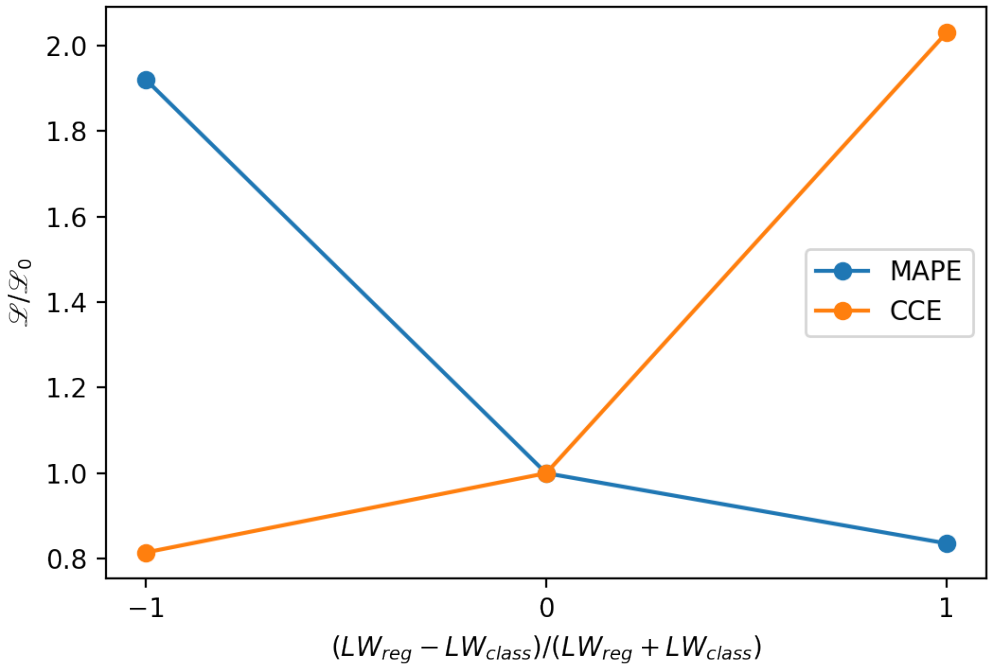


Figure 50: Asymmetric progression of two different losses (MAPE, CCE) as function of their contribution (encoder loss weight LW) to the global loss. A value of -1 corresponds to pure classification loss, $+1$ to pure regression loss.

to explore in further studies. The denoising application of autoencoders encourages such studies.

5.2.6 Information bottleneck

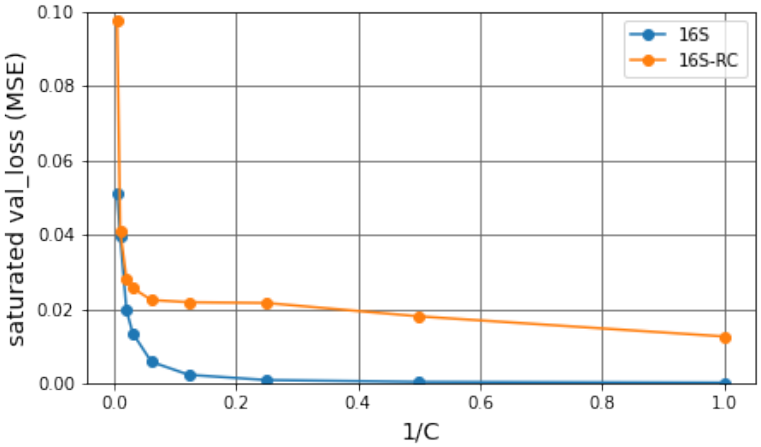
In the last experiment, the prediction performance of the three objectives with respect to the size of the latent space was investigated. For this purpose, the dense layer of the 16-Split, that defines the size of the latent space, has been varied while the rest of the architecture was kept unchanged. The saturated loss values as a function of the inverse compression ratio between input and latent dimensions was determined and can be seen in Fig. 51.

It can be seen that the losses on the validation set follow the same falling exponential trend as the CAE (Fig. 39) when changing the bottleneck size. Saturation seems to take place at a relatively early stage ($1/C = 0.02$), suggesting that already small amount latent dimensions satisfy all three objectives reasonably well, and that further increase of size does not increase the performance much, without changing the training strategy or augmenting the datasets.

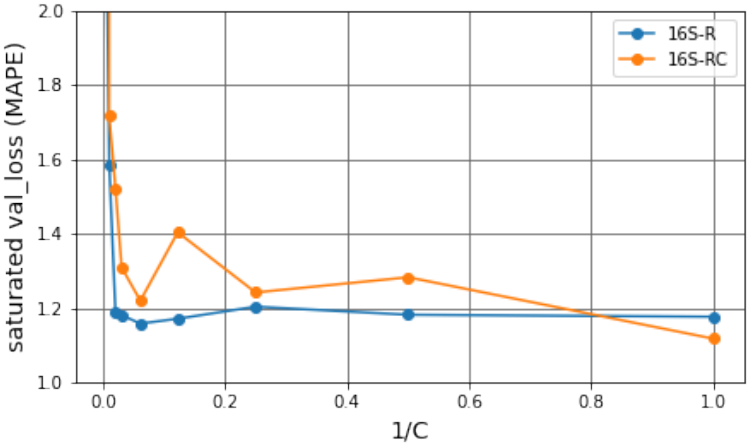
Tab. 4 shows the corresponding measurements. Classification accuracy and regression loss seem to saturate very early on, while reconstruction loss is still improving. This supports the hypothesis made in the previous section – classification and regression tasks need less latent capacity and therefore saturate earlier. The reason why they are still (slightly) improving can be linked to a larger dense layer per encoder, thus a more informative code per image fragment to compensate for the random positioning of the input digit.

Table 4: Loss performance of different objectives after varying the inverse compression ratio $1/C$ (latent dimensions/input dimensions) of a model while keeping the rest of the architecture unchanged. Models are trained until saturation on either reconstruction, classification or regression alone (16S, 16S-C, 16S-R) or the encoders are trained on regression and classification, after which the encoder parameters are fixed and the reconstruction head is trained on top of this (16-RC). For each model the values corresponding to best classification accuracy are selected and summarized in the table. Models with larger capacity needed to be trained with an exponentially larger number of epochs in order to reach a saturation level.

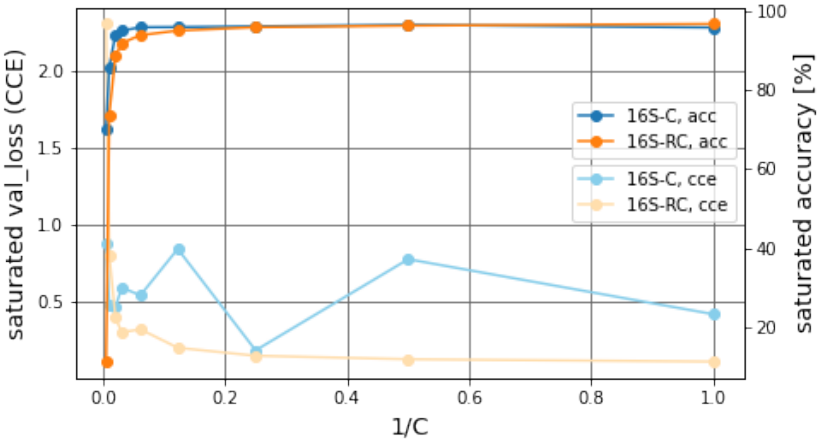
1/C	**MSE**		**CCE, Acc.**		**MAPE**	
	16S	**16S-RC**	**16S-C**	**16S-RC**	**16S-R**	**16S-RC**
0.005	0.0509	0.0976	0.8782, 70.15%	2.3011, 11.34%	2.3214	28.2130
0.010	0.0394	0.0409	0.4783, 85.79%	0.8049, 73.68%	1.5861	1.7168
0.020	0.0197	0.0281	0.4653, 93.88%	0.3986, 88.49%	1.1892	1.5217
0.030	0.0135	0.0256	0.5932, 94.88%	0.3026, 91.75%	1.1805	1.3084
0.061	0.0058	0.0224	0.5422, 95.76%	0.3224, 93.76%	1.1580	1.2194
0.122	0.0023	0.0218	0.8431, 95.80%	0.2026, 94.91%	1.1710	1.4032
0.250	0.0009	0.0216	0.1859, 95.99%	0.1509, 95.77%	1.2034	1.2416
0.500	0.0004	0.0180	0.7766, 96.37%	0.1284, 96.17%	1.1817	1.2824
1.000	0.0002	0.0126	0.4204, 95.70%	0.1129, 96.60%	1.1764	1.1173



(a)



(b)



(c)

Figure 51: Saturated loss value as a function of decoder input latent dimensions for the 16-Split encoder model trained with different heads and with different learning objectives. The curves correspond to the values summarized in Tab. 4.

This study showed that above a certain threshold of latent dimensions, the performance on all tasks does not improve much. Classification seems to saturate earlier on in performance. Regression of total energy, as the third objective of trigger primitive information, was not discussed so far, which should be included in future studies.

5.3 Discussion

In summary, the six studies above show the feasibility and performance of a neural network for the purpose of trigger primitive generation by considering major architectural detector properties and constraints.

The theoretical information content desired to preserve for trigger primitive generation amounts to 131 bits: three bits for the type, three 32 bit floats for the position, and one 32 bit float for the energy of the shower-inducing particle. Fig. 52 shows test beam data for a shower caused by a 250 GeV electron in eight consecutive modules convering 27 X_0.

In the final system, eight (low-density) modules will mount 8×3 ROCs, each accessing 16 trigger cells with 7 bits per cell which gives an input space of 2688 bits. If one only considers 99 bits for predicting the particle class and its position, an inverse compression ratio of $99/2688 \approx 0.037$ should theoretically be sufficient to recover the information of this particular shower.

As the studies in Sec. 5.2 showed, a neural network trained on classification and regression can recover the desired information on a similar toy problem with fairly accurate precision. Fig. 53 shows classification accuracies per digit class on the entire validation set in a confusion matrix for a 16-Split model of 128 latent dimensions. The compression ratio between input space and bottleneck of this model amounts to 0.04, comparable to the ratio of 0.037 for the example shower above. It is noteworthy that the toy dataset contains about twice as many classes as the real shower data.

The true positives (along the diagonal of the matrix) show a maximum difference of 14%. Although this result cannot be directly compared to the performance of the ECON-T algorithms in Fig. 34, there exists a parallel between the increase in trigger rate at constant offline threshold and classification accuracy of a neural network based approach. This parallel could allow to answer the question as to whether the neural network performance is better on different particle types than the current ECON-T algorithms and is therefore worth exploring in future studies.

Fig. 54 shows the median residuals inside upper and lower quartiles for regression of position in the x- and y-coordinates. The closeness of the median residuals to zero show that there is no pathologic deviation between the regressed value and the true coordinate.

Furthermore, the impact of pile-up in the detector has not been considered so far. Autoencoders are often used for the purpose of noise reduction, i.e. to produce a noise-free reconstruction from a noisy input image. The performance on classification and regression could be determined in a future study. Furthermore, these studies considered only a single stage of encoding, whereas the proposed concept is composed of three such stages. The effect of chaining multiple stages should also be subject of a future study.

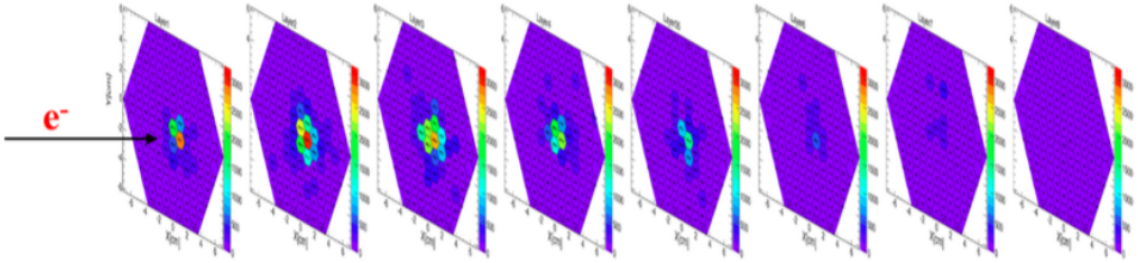


Figure 52: Example of test beam data for sensor occupancy due to a shower induced by a 250 GeV electron [55].

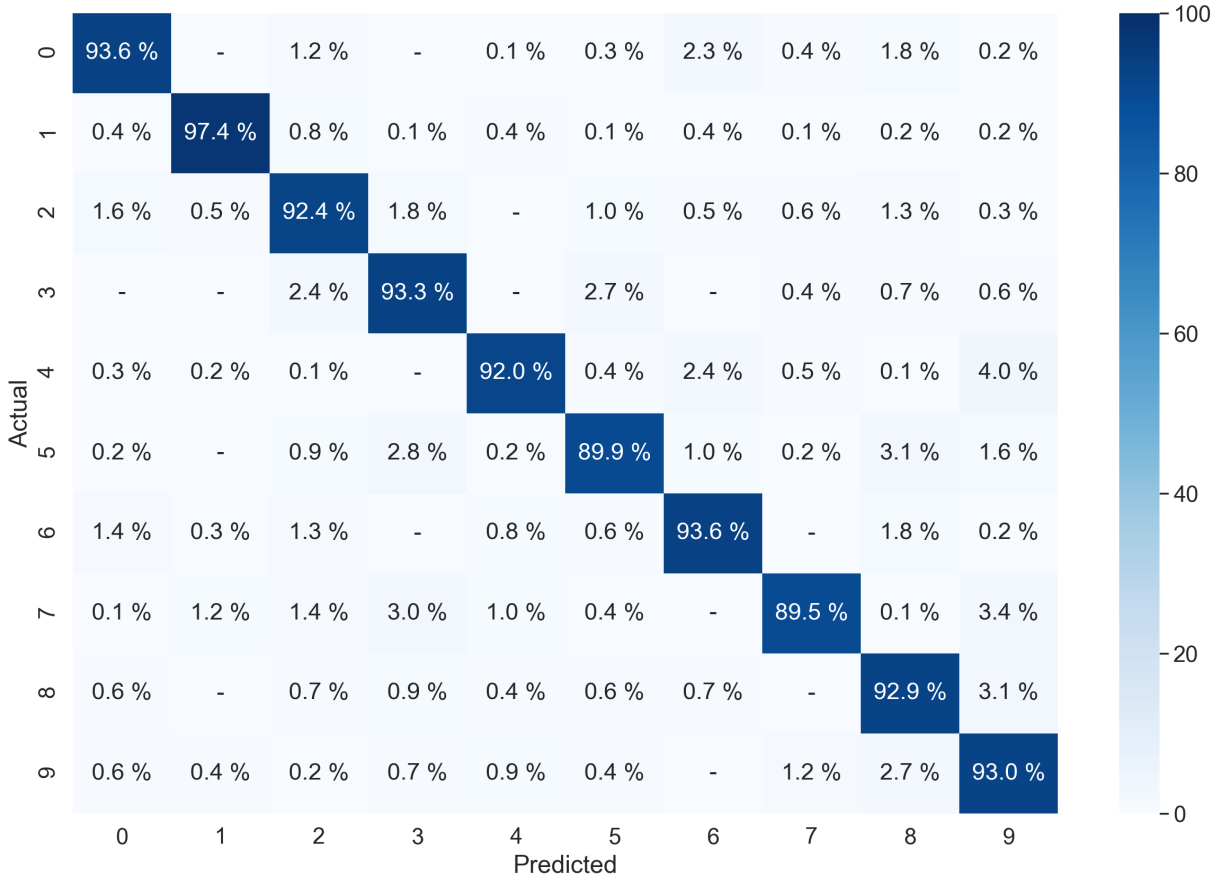
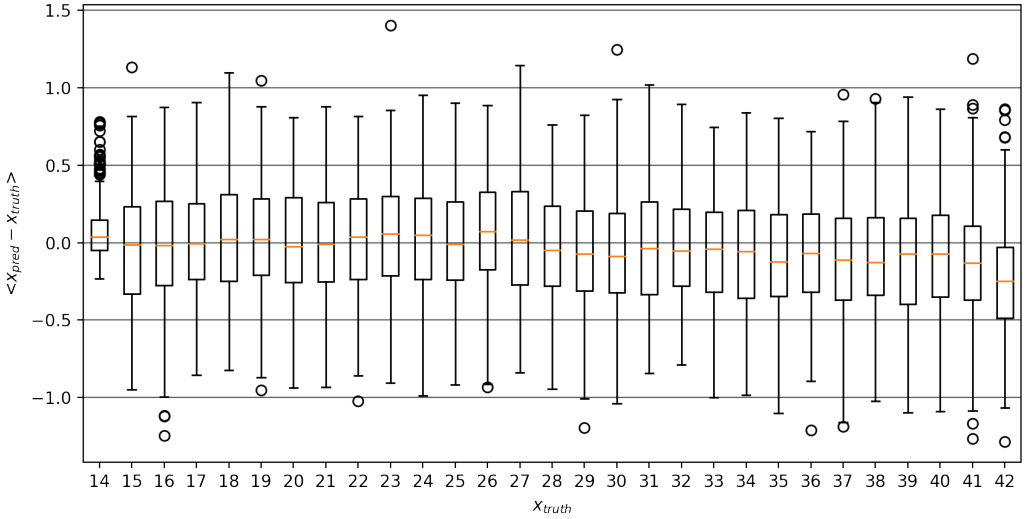
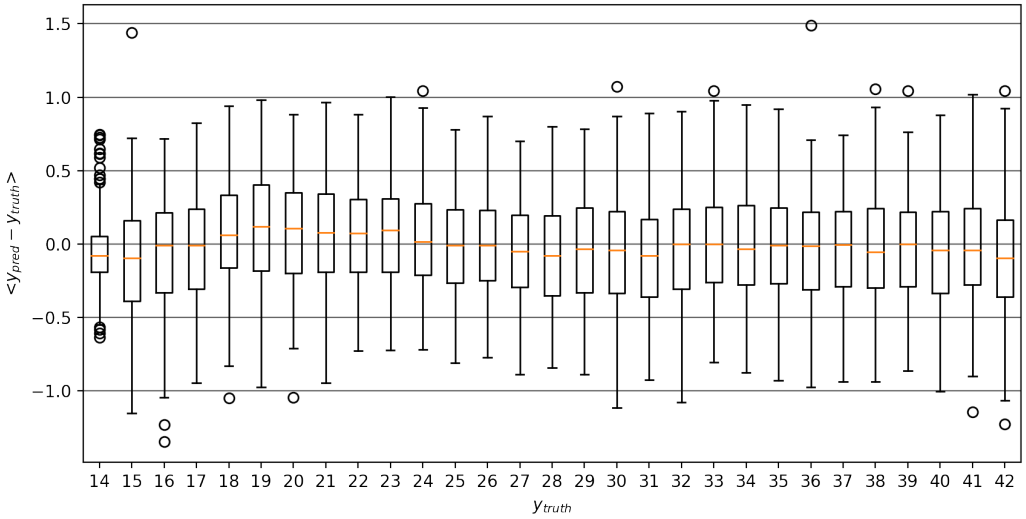


Figure 53: Confusion matrix showing prediction accuracy of true positives along the diagonal and accuracy of false positives in all other positions. The plot summarizes the classification results on the validation set for 16-split trained on regression and classification simultaneously. The x-axis shows the predicted digit (class) labels and the y-axis the actual labels.



(a)



(b)

Figure 54: Box plot showing the median (orange) in its interquartile range (box) and outliers are marked by circles for 16-Split (trained on regression and classification simultaneously, and with a latent space of 128 decoder input dimensions) regression on the validation set: (a) The x-coordinate and (b) the y-coordinate. The positions are calculated from the center of the 28×28 digit embedded in the 56×56 image and cover the range (14,14) to (42, 42).

6 Summary & Outlook

As the development on the HGCAL approaches the production phase, quality control during the fabrication of detector components and suitable implementation concepts of its subsystems need to be in place. The main contributions to these ends presented in this thesis are the development of software for readout chip configuration, the creation and analysis of online chip testing procedures, as well as the implementation of a neural network based concept architecture involving the HGCAL trigger primitive generator.

The original motivation for the first main contribution of developing software for chip configuration was to enable running testing procedures on different test systems from a remote computer in different testing facilities. The software[13] developed in this thesis satisfies these requirements by exposing a common chip API, while handling the detection of different test systems internally and by providing an ethernet based server/client interface which accepts human-readable configuration files that are converted to register level and subsequently written to the chips. Furthermore, the software introduces caching and certain I/O optimizations that enable faster configuration and a layer of fault tolerance. Due to its simple installation and ease of use, the software has been and is a critical component for chip tests in the past and in the future.

The second main contribution is the testing of 223 single chips and 54 hexaboards in four testing campaigns (June–August 2019). These tests contributed to the development of HGCAL in two significant ways. First, they provided the data, which is being analyzed at the time of writing, in order to determine the characteristics of chip populations. Secondly, they informed the development process of the frontend electronics components by providing feedback to the board- and chip-designers. The test results led to a better understanding of the chip parameters, for example the connection between the shaper reference voltage setting and the overall dynamic range, as well as to further investigations into revealing issues, like noise and unexpected TDC measurements, and to the redesign of the hexaboard.

The final main outcome of this thesis is the development of a prototype to deliver a preliminary proof-of-concept showing the feasibility of a multi-stage, end-to-end trained neural network based trigger primitive generator on a toy problem. The presented concept studies[14] successively introduced realistic architectural detector constraints on the prototype and evaluated the impact on its performance. The produced models have not been tuned for optimal performance, which should be subject of future studies. In parallel to the work presented so far, the concept architecture has been implemented and trained also on simulated shower data[15], by making use of the insights delivered by the studies of this thesis.

In conclusion, the work presented in this thesis made important contributions to the HGCAL upgrade by providing a critical part of the software testing suite, by performing chip characterization measurements, and by presenting an alternative neural network based approach to trigger primitive generation inside HGCAL.

[13] `https://gitlab.cern.ch/hgcal-daq-sw/zmq_i2c`
[14] `https://gitlab.cern.ch/dwinter/micae`
[15] `https://gitlab.cern.ch/tquast/hgc_l1_trigger_autoencoder`

Appendices

A Gradient Descent and Backpropagation

Calculating the gradient of $\mathscr{L}$ with respect to each network parameter shows the impact that changing the parameters has on the loss. Thus, the training process is the optimization problem of minimizing the loss by varying the network parameters. The most effective way to do so is to change a parameter p in the opposite direction of the loss's gradient with respect to p, i.e. in the direction of steepest (gradient) descent:

$$p' = p - \eta \nabla_p \mathscr{L}, \tag{16}$$

with a (small) step size η, called learning rate. According to this update rule, the network is guaranteed to find a (local) minimum of the loss function. The process is illustrated in Fig. 55.

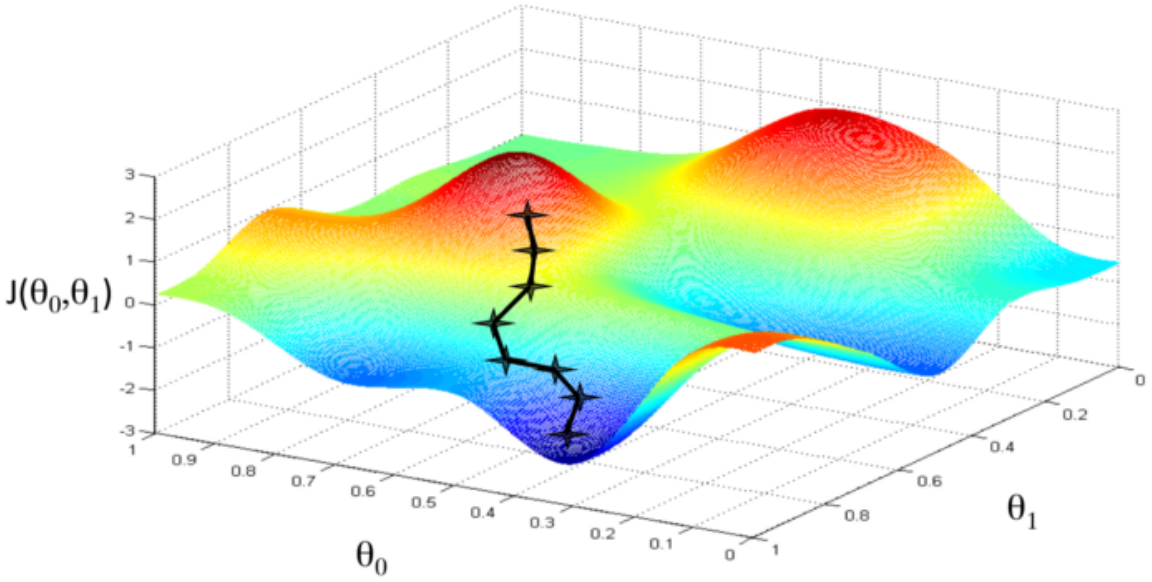


Figure 55: Visualization of the gradient descent algorithm for a model with parameters Θ_0 and Θ_1. Figure reproduced from Ref. [60].

As it is desired for neural networks to generalize on a population of data describing a problem, the learning process needs to take a sufficiently large (representative) subset of the population into account. The network parameters are then updated via the average of gradients for all samples in this (training) set. Thereby, the network has to consider the whole population for each training step. Since this updating procedure requires a lot of calculation and memory usage, a variant called *stochastic* gradient descent (SGD) is commonly used. SGD considers small, randomly selected, subsets of the training data (usually 32 or 64 samples per subset), called mini-batches to calculate the average gradient and update the parameters. Therefore, the network is able to learn faster, but also less precisely when converging towards the minimum.

The gradients of the loss function are calculated by tracking the data transformations of the forward pass and subsequently calculating partial derivatives $\delta\mathscr{L}/\delta b$ and $\delta\mathscr{L}/\delta w$ at each node. As the transformation of a data point $\vec{x}$ in the network can mathematically be represented by function applications $a(\vec{x})$ at each layer, a succession of layers is simply a composition of functions $a_n \circ ... \circ a_1 \circ a_0(\vec{x})$, starting from the output layer, exploiting the chain rule of multi-variate calculus and the compositional nature of the network layers. This procedure, referred to as backward propagation is illustrated in Fig. 56; the loss is labeled E, the activation y_i and weights denoted as $\delta E/\delta w$. Note that biases are ignored.

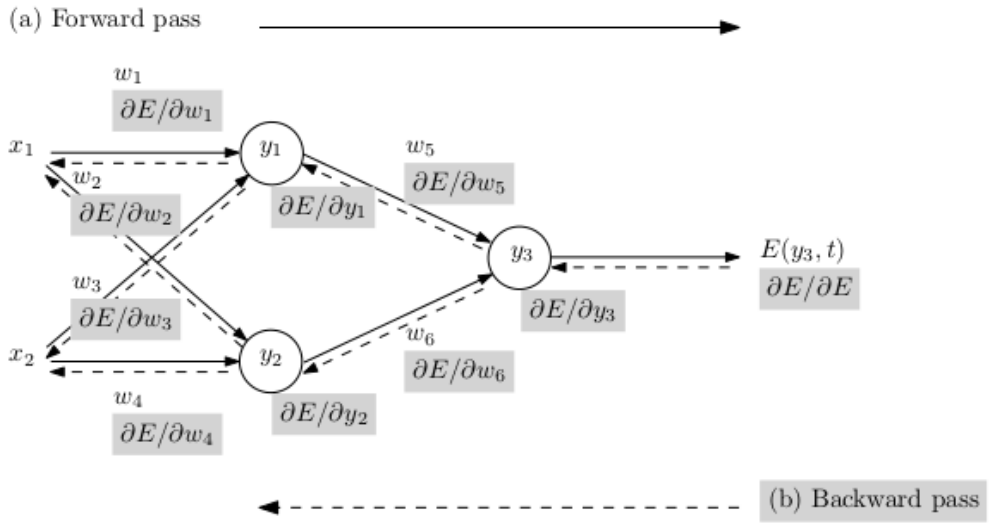


Figure 56: Illustration of forward and backward pass and the components of the computational graph used during neural network training process. Biases are ignored. Figure reproduced from Ref. [20].

The backpropagation algorithm can be developed by starting from the loss function $\mathscr{L}$ and successively applying the chain rule. At the beginning, the error term of a neuron in the output layer L is defined as

$$\delta_i^L = \frac{\partial \mathscr{L}}{\partial z_i^L} = \frac{\partial \mathscr{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}, \tag{17}$$

for the non-linear activation $a_i^l = \sigma(z_i^l)$ and $z_i^l = \boldsymbol{W}^l \vec{a}^{l-1} + b_i$. Thus, the rule for computing the previous layer's error can be found:

$$\delta_i^{L-1} = \frac{\partial \mathscr{L}}{\partial z_i^{L-1}} = \underbrace{\frac{\partial \mathscr{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}}_{\delta_i^L}\frac{\partial z_i^L}{\partial \vec{a}^{L-1}}\frac{\partial \vec{a}^{L-1}}{\partial \vec{z}^{L-1}} = \delta_i^L \boldsymbol{W}^L \sigma'(\vec{z}^{L-1}). \tag{18}$$

Thus for any layer l one obtains:

$$\delta_i^l = (\boldsymbol{W}^{l+1})^T \vec{\delta}^{l+1} \sigma'(z^l). \tag{19}$$

The partial derivatives of the loss with respect to the biases and weights of any layer can be found in a similar manner:

$$\frac{\partial \mathscr{L}}{\partial w_{ij}^l} = \underbrace{\frac{\partial \mathscr{L}}{\partial a_i^l}\frac{\partial a_i^l}{\partial z_i^l}}_{\delta_i^l}\frac{\partial z_i^l}{\partial w_{ij}^l} = \delta_i^l a_j^{l-1} \tag{20}$$

for the weights and

$$\frac{\partial \mathscr{L}}{\partial b_i^l} = \underbrace{\frac{\partial \mathscr{L}}{\partial a_i^l}\frac{\partial a_i^l}{\partial z_i^l}}_{\delta_i^l}\underbrace{\frac{\partial z_i^l}{\partial b_i^l}}_{1} = \delta_i^l \tag{21}$$

for the biases. Eq. 17 and Eq. 19 can be written in vector form using the Hadamard product $\odot$ (element-wise multiplication) and the gradient of the loss with respect to an activation $\nabla_{a^L}\mathscr{L}$:

$$\vec{\delta}^L = \nabla_{a^L}\mathscr{L} \odot \sigma'(\vec{z^L}) \tag{22}$$

$$\vec{\delta}^l = ((\mathbf{W}^{l+1})^T\vec{\delta}^{l+1}) \odot \sigma'(\vec{z}^l). \tag{23}$$

B Principal Component Analysis

While Fig. 31 schematically shows a deep undercomplete autoencoder, there exists an even simpler type with a single hidden layer. By considering linear neuron activation $\vec{a}(\vec{x}) = \mathbf{W}\vec{x}+\vec{b}$, it turns out that this simple network learns an encoding function similar to a dimensionality reduction technique called principal component analysis (PCA). In PCA the original feature axes of a data set are swapped with new ones (as linear combinations of the originals) that are aligned with the directions of greatest variance in the data [9]. Mathematically, this is accomplished by eigendecomposition $\mathbf{C} = \mathbf{P}\boldsymbol{\lambda}\mathbf{P}^T$ of the unbiased covariance matrix $\mathbf{C}$ associated with data points $\vec{x}_i$ contained in the data set $\mathbf{X}$. $\mathbf{C}$ is defined as

$$\mathbf{C} = \frac{1}{m-1}\mathbf{X}\mathbf{X}^T. \tag{24}$$

The eigenvectors $\vec{v}_i$ corresponding to the largest eigenvalues λ_i resemble the n principal components, where m is the number of input/output dimensions and $n \leq m$. The eigenvectors and eigenvalues can be found according to

$$\mathbf{C}\cdot\mathbf{P} = \mathbf{P}\cdot\boldsymbol{\lambda}, \tag{25}$$

with $\mathbf{P}$, the matrix containing the eigenvectors $\vec{v}_i$ in its columns and $\boldsymbol{\lambda} = \mathrm{diag}(\lambda_i)$:

$$\mathbf{P} = \begin{pmatrix} v_{1,1} & v_{2,1} & \cdots & v_{n,1} \\ v_{1,2} & v_{2,2} & \cdots & v_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1,m} & v_{2,m} & \cdots & v_{n,m} \end{pmatrix} \text{ and } \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}. \tag{26}$$

Thus, the transformation from m inputs to n latent features is accomplished by the transformation $e(\vec{x}) = \mathbf{P}^T\vec{x}$, and the reverse transformation into input space via $d(e(\vec{x})) = \mathbf{P}\mathbf{P}^T\vec{x}$. When $n<m$, then $\mathbf{C} \neq \mathbf{P}\boldsymbol{\lambda}\mathbf{P}^T$ and $d(e(\vec{x}))$ cannot preserve all information anymore: $d(e(\vec{x})) \neq \vec{x}$. It is noteworthy to mention that for PCA to succeed principal features

must exist in the data to begin with, i.e. correlation in the features of the input data is presumed.

Both PCA and the linear undercomplete autoencoder are looking for the same linear subspace. The only difference is that the basis vectors of autoencoders are not necessarily orthogonal, while the eigenvectors of the (symmetric) covariance matrix for PCA are by principle always orthogonal [36]. Furthermore, PCA requires the eigenvectors to be normalized, thus yielding a unique solution while for autoencoder training there exists no such precondition.

C Neural network model architectures

Table 5: **CAE**: Baseline convolutional autoencoder

Layer/Operation	**Kernel**	**Features**	**Stride**	**Padding**	**Activation**
Input shape $28 \times 28 \times 1$					
Convolution 2D	3×3	32 maps	2×2	same	ReLU
Convolution 2D	3×3	64 maps	2×2	same	ReLU
Flatten to 16×1					
Densely connected	N/A	16 nodes	N/A	N/A	Sigmoid
Densely connected	N/A	392 nodes	N/A	N/A	ReLU
Reshape to $7 \times 7 \times 8$					
Transposed convolution 2D	3×3	64 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	32 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	1 map	1×1	same	Linear
Output shape $28 \times 28 \times 1$					

Table 6: **FCAE**: Fully convolutional autoencoder

Layer/Operation	**Kernel**	**Features**	**Stride**	**Padding**	**Activation**
Input shape $28 \times 28 \times 1$					
Convolution 2D	3×3	32 maps	2×2	same	ReLU
Convolution 2D	3×3	64 maps	2×2	same	ReLU
Convolution 2D	4×4	1 map	1×1	valid	Sigmoid
Flatten to 16×1					
Reshape to $4 \times 4 \times 1$					
Transposed convolution 2D	4×4	1 maps	1×1	valid	ReLU
Transposed convolution 2D	3×3	64 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	32 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	1 map	1×1	same	Linear
Output shape $28 \times 28 \times 1$					

Table 7: **4S**: 4-Split

Layer/Operation	**Kernel**	**Features**	**Stride**	**Padding**	**Activation**
Input shape $4 \times 14 \times 14 \times 1$					
Convolution 2D	3×3	32 maps	2×2	same	ReLU
Convolution 2D	3×3	64 maps	2×2	same	ReLU
Flatten to 4×1					
Densely connected	N/A	4 nodes	N/A	N/A	Sigmoid
Concatenate from 4×4 to 16×1					
Densely connected	N/A	392 nodes	N/A	N/A	ReLU
Reshape to $7 \times 7 \times 8$					
Transposed convolution 2D	3×3	64 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	32 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	1 map	2×2	same	Linear
Output shape $28 \times 28 \times 1$					

Table 8: **F4S**: Fully convolutional 4-Split

Layer/Operation	**Kernel**	**Features**	**Stride**	**Padding**	**Activation**
Input shape $4 \times 14 \times 14 \times 1$					
Convolution 2D	3×3	32 maps	2×2	same	ReLU
Convolution 2D	3×3	64 maps	2×2	same	ReLU
Convolution 2D	3×3	1 map	1×1	valid	Sigmoid
Flatten to 4×1					
Concatenate from 4×4 to 16×1					
Reshape to $4 \times 4 \times 1$					
Transposed convolution 2D	4×4	1 map	1×1	valid	ReLU
Transposed convolution 2D	3×3	64 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	32 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	1 map	1×1	same	Linear
Output shape $28 \times 28 \times 1$					

Table 9: **16S**: 16-Split

Layer/Operation	**Kernel**	**Features**	**Stride**	**Padding**	**Activation**
Input shape $16 \times 14 \times 14 \times 1$					
Convolution 2D	3×3	32 maps	2×2	same	ReLU
Convolution 2D	3×3	64 maps	2×2	same	ReLU
Flatten to 4×1					
Densely connected	N/A	4 nodes	N/A	N/A	Linear
Batch Normalization					
Activation	N/A	N/A	N/A	N/A	Sigmoid
Collect 16×4 *outputs after 16 encoder iterations.*					
Concatenate from 16×4 to 48×1 (*16-Split encoder output*)					
Densely connected	N/A	1568 nodes	N/A	N/A	ReLU
Reshape to $14 \times 14 \times 8$					
Transposed convolution 2D	3×3	64 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	32 maps	2×2	same	ReLU
Transposed convolution 2D	3×3	1 map	1×1	same	Linear
Output shape $56 \times 56 \times 1$					

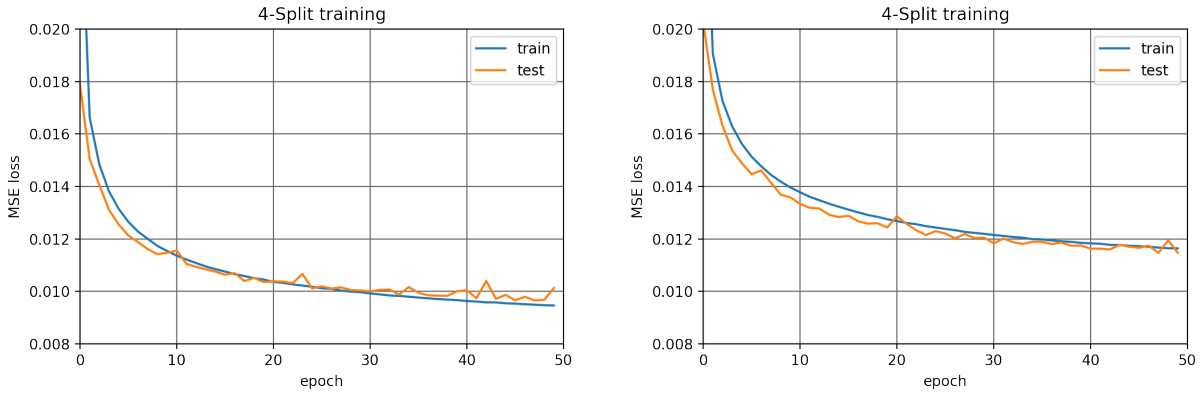
Table 10: **16S-C**: 16-Split classifier head

Layer/Operation	**Kernel**	**Features**	**Stride**	**Padding**	**Activation**
Input shape 48×1 from *16-Split encoder output*					
Densely connected	N/A	256 nodes	N/A	N/A	ReLU
Densely connected	N/A	128 nodes	N/A	N/A	ReLU
Densely connected	N/A	64 nodes	N/A	N/A	ReLU
Densely connected	N/A	10 nodes	N/A	N/A	Softmax
Output shape 10×1					

Table 11: **16S-R**: 16-Split regressor head

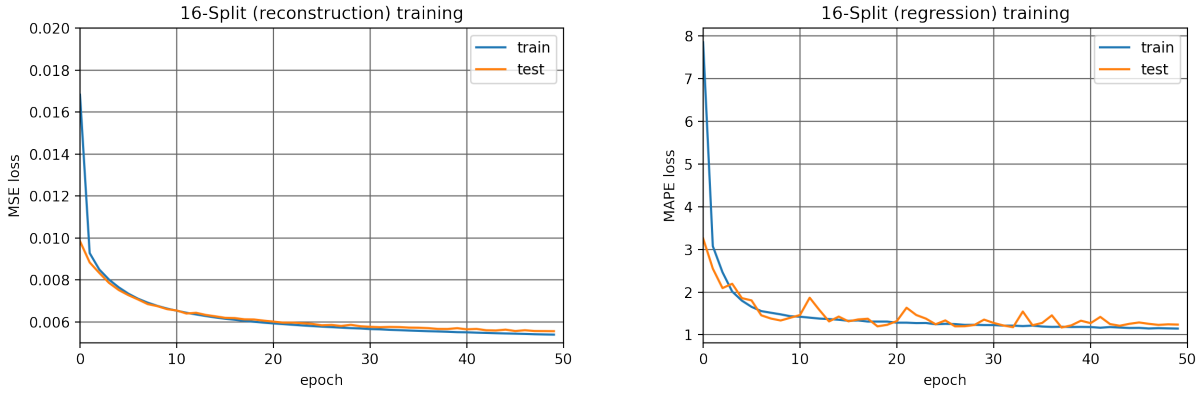
Layer/Operation	**Kernel**	**Features**	**Stride**	**Padding**	**Activation**
Input shape 48×1 from *16-Split encoder output*					
Densely connected	N/A	256 nodes	N/A	N/A	ReLU
Densely connected	N/A	128 nodes	N/A	N/A	ReLU
Densely connected	N/A	64 nodes	N/A	N/A	ReLU
Densely connected	N/A	2 nodes	N/A	N/A	Linear
Output shape 2×1					

D Neural network training progression

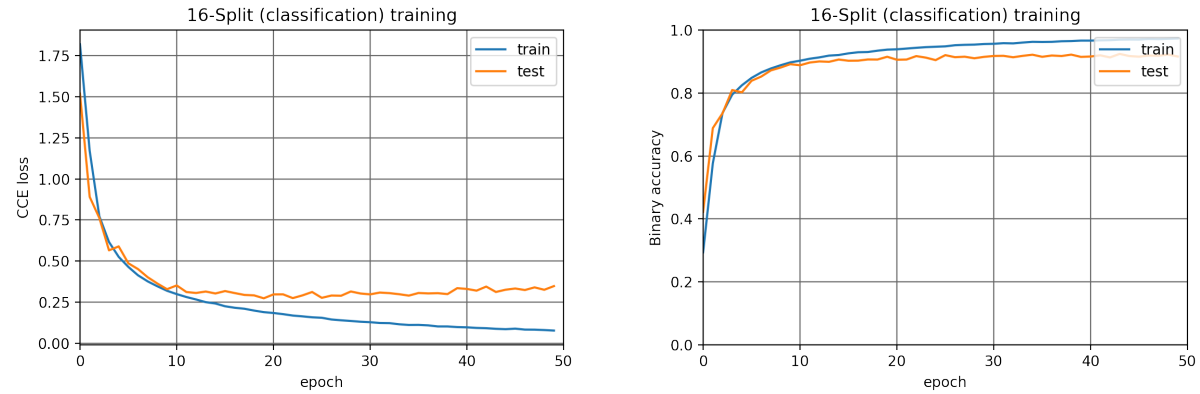


(a) (b)

Figure 57: 4-split reconstruction loss for (a) free weights and (b) shared weights, trained over 50 epochs.



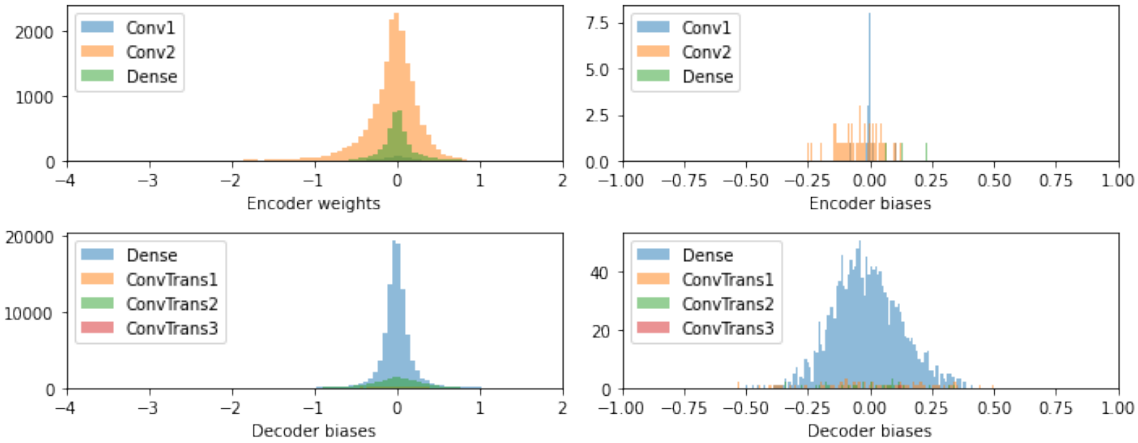
(a) (b)



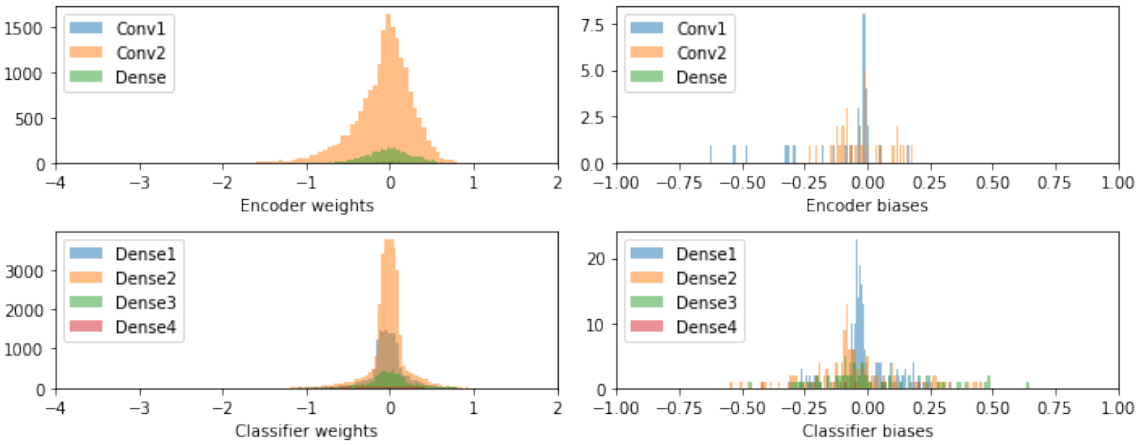
(c) (d)

Figure 58: 16-split (a) Reconstruction loss, (b) Regression loss, (c) Classification loss, and (d) classification accuracy, for a model trained over 50 epochs.

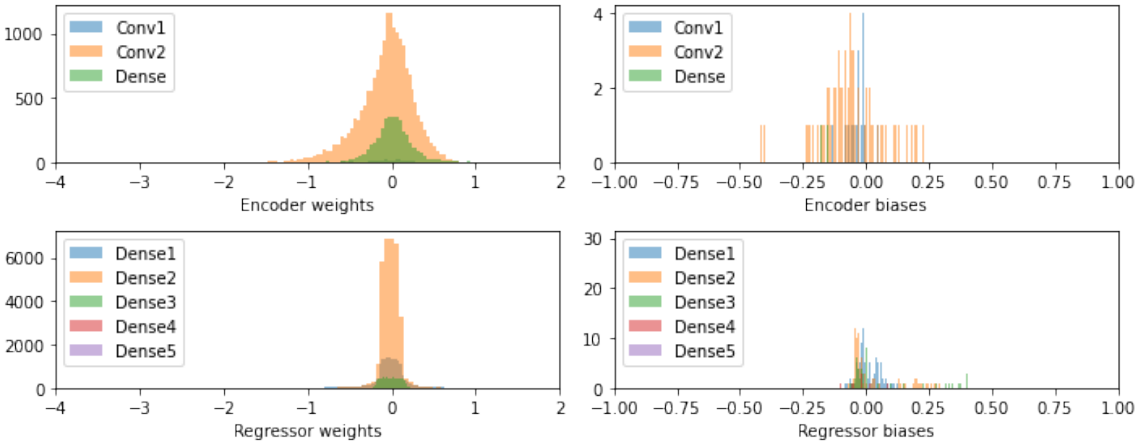
E 16-Split weight distributions



(a)



(b)



(c)

Figure 59: 16-Split weight and bias distributions per layer. Encoder parameters and (a) Decoder, (b) Classifier, and (c) Regressor parameters, each individually trained for 50 epochs.

References

[1] CMS Luminosity - Public Results. https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults#2018_proton_proton_collisions. Last accessed: 05 January 2021.

[2] Convolution operation (convolve). Last accessed: 05 January 2021. URL: https://peltarion.com/knowledge-center/documentation/glossary.

[3] HGCAL project Schedule. https://project-hl-lhc-industry.web.cern.ch/content/project-schedule. Last accessed: 11 December 2020.

[4] I2C Info - I2C Bus, Interface and Protocol. https://i2c.info/i2c-bus-specification. Last accessed: 19 November 2020.

[5] LHC Season 2: Facts & Figures. https://home.cern/sites/home.web.cern.ch/files/2018-07/factsandfigures-en_0.pdf. Last accessed: 09 December 2020.

[6] Magnets & Detectors — Taking a closer look at LHC. https://www.lhc-closer.es/taking_a_closer_look_at_lhc/0.magnets___detectors_i. Last accessed: 19 January 2021.

[7] MNIST Database. https://en.wikipedia.org/wiki/MNIST_database. Last accessed: 05 January 2021.

[8] Standard Model. https://en.wikipedia.org/wiki/Standard_Model. Last accessed: 09 December 2020.

[9] Understanding Variational Autoencoders (VAEs. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. Last accessed: 13 January 2021.

[10] ZeroMQ. https://zeromq.org/. Last accessed: 19 November 2020.

[11] Trophyboard Schematic. https://edms.cern.ch/file/2232628/1/Trophy_Board_EDMS2232628V1_Schematics.pdf, 10 2019. Last accessed: 16 November 2020.

[12] J.-M. Andre et al. The CMS Data Acquisition - Architectures for Phase-2 Upgrade. *J. Phys.: Conf. Ser. 898 032019*, 559, 2017. URL: https://iopscience.iop.org/article/10.1088/1742-6596/898/3/032019.

[13] M. Anfreville et al. Laser monitoring system for the CMS lead tungstate crystal calorimeter. *Nucl. Instrum. Meth. A*, 594:292–320, 2008. doi:10.1016/j.nima.2008.01.104.

[14] G. Apollinari, A. I. Béjar, O. Brüning, P. Fessia, M. Lamont, L. Rossi, and L. Tavian. *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1.* CERN Yellow Reports: Monographs. CERN, Geneva, 2017. URL: https://cds.cern.ch/record/2284929, doi:10.23731/CYRM-2017-004.

[15] P. Aspell, P. Bloch, J. Hirschauer, M. Noy, and P. M. Rubinov. HGCAL silicon module DAQ and TRG architecture specification. Technical report, CERN, FNAL, Imperial College, 04 2020. Rev. 4. URL: https://indico.fnal.gov/event/15258/material/1/0.pdf.

[16] R. Aßmann, M. Lamont, and S. Myers. A brief history of the lep collider. *Nuclear Physics B - Proceedings Supplements*, 109(2):17 – 31, 2002. Proceedings of the 7th Topical Seminar. URL: http://www.sciencedirect.com/science/article/pii/S0920563202900058, doi:https://doi.org/10.1016/S0920-5632(02)90005-8.

[17] The ATLAS and CMS Collaboration. Expected pile-up values at the HL-LHC. Technical report, CERN, 09 2013. URL: https://cds.cern.ch/record/1604492?ln=en.

[18] P. Azzi, D. Barney, T. Bergauer, P. Dauncey, A. David, M. Mannelli, J. Mans, A. Marchioro, A. Martelli, S. Moccia, M. Narain, C. Seez, F. Sefkow, C. de la Taille, and T. Virdee. The Phase-2 Upgrade of the CMS endcap calorimeter. Technical report, CERN, 4 2018. URL: https://cds.cern.ch/record/2293646?ln=en.

[19] D. Barney. CMS High-Granularity Calorimeter (HGCAL). https://indico.cern.ch/event/718124/attachments/1624533/2612800/HighGranularityCalorimeter_CERN_200418_small.pdf, 4 2018.

[20] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey, 2018. URL: https://arxiv.org/abs/1502.05767, arXiv:1502.05767.

[21] M. Benedikt, P. Collier, V. Mertens, J. Poole, and K. Schindl. *LHC Design Report*. CERN Yellow Reports: Monographs. CERN, Geneva, 2004. URL: http://cds.cern.ch/record/823808, doi:10.5170/CERN-2004-003-V-3.

[22] L. Benussi, Bianco S., et al. CMS. Technical report, CERN, 2012. URL: https://www.lnf.infn.it/rapatt/2015/CMS.pdf.

[23] E. Brodolin. Silicon sensors for the CMS HGCAL upgrade: Challenges, sensor design & electrical characterization. Technical report, CERN, 11 2019. URL: https://arxiv.org/abs/2003.02461.

[24] O. S. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, and P. Proudlock. *LHC Design Report*. CERN Yellow Reports: Monographs. CERN, Geneva, 2004. URL: http://cds.cern.ch/record/782076, doi:10.5170/CERN-2004-003-V-1.

[25] O. S. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, and P. Proudlock. *LHC Design Report*. CERN Yellow Reports: Monographs. CERN, Geneva, 2004. URL: http://cds.cern.ch/record/815187, doi:10.5170/CERN-2004-003-V-2.

[26] F. Chollet. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017. URL: https://www.manning.com/books/deep-learning-with-python.

[27] CMS Collaboration. About CMS. https://cms.cern/detector. Last accessed: 09 December 2020.

[28] CMS Collaboration. Single-Particle Response in the CMS Calorimeters. Technical Report CMS-PAS-JME-10-008, CERN, Geneva, 2010. URL: http://cds.cern.ch/record/1279141.

[29] CMS Collaboration. *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software*. Technical Design Report CMS. CERN, Geneva, 2006. URL: https://cds.cern.ch/record/922757.

[30] CMS Collaboration. The CMS experiment at the CERN LHC. *Journal of Instrumentation*, 3(08):S08004–S08004, aug 2008. URL: https://doi.org/10.1088%2F1748-0221%2F3%2F08%2Fs08004, doi:10.1088/1748-0221/3/08/s08004.

[31] D. Contardo, M. Klute, J. Mans, L. Silvestris, and J. Butler. Technical Proposal for the Phase-II Upgrade of the CMS Detector. Technical report, CERN, 06 2015. URL: https://cds.cern.ch/record/2020886.

[32] P. Drannan. Understanding MOSFET Mismatch for Analog Design. *IEEE Journal of Solid-State Circuits*, 38, No.3, 3 2003. URL: https://ieeexplore.ieee.org/document/1183852.

[33] A. Duperrin. High Luminosity yellow report: what does HL-LHC physics look like in ATLAS and CMS? Sep 2019. URL: http://cds.cern.ch/record/2689937.

[34] F. Englert and R. Brout. Broken symmetry and the mass of gauge vector mesons. *Phys. Rev. Lett.*, 13:321–323, Aug 1964. URL: https://link.aps.org/doi/10.1103/PhysRevLett.13.321, doi:10.1103/PhysRevLett.13.321.

[35] C. W. Fabjan and F. Gianotti. Calorimetry for Particle Physics. *Rev. Mod. Phys.*, 75:1243–1286, 10 2003. URL: https://link.aps.org/doi/10.1103/RevModPhys.75.1243, doi:10.1103/RevModPhys.75.1243.

[36] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[37] P. W. Higgs. Broken symmetries, massless particles and gauge fields. *Physics Letters*, 12(2):132 – 133, 1964. URL: http://www.sciencedirect.com/science/article/pii/0031916364911369, doi:https://doi.org/10.1016/0031-9163(64)91136-9.

[38] K. Jahr, R. Schlich, K. Dragos, and K. Smarsly. *Decentralized autonomous fault detection in wireless structural health monitoring systems using structural response data*. 07 2015. URL: https://www.researchgate.net/publication/289479445_Decentralized_autonomous_fault_detection_in_wireless_structural_health_monitoring_systems_using_structural_response_data.

[39] J. Jordan. Introduction to autoencoders. https://www.jeremyjordan.me/autoencoders/. Last accessed: 08 January 2021.

[40] T. Kolberg, K. Kruger, and J. Mans. Overall Layout information for the scintillator HGCAL. Technical report, CERN, DESY, FSU, UMN, FNAL, 09 2020. EDMS No. 2328490. URL: https://cms.desy.de/sites/sites_desygroups/sites_extern/site_cms/content/e78437/e83421/e85075/e85081/e121141/HGCAL-Varna.pdf.

[41] Y. Lecun. The MNIST Database of handwritten digits. http://yann.lecun.com/exdb/mnist/. Last accessed: 08 January 2021.

[42] A. Lobanov. ProbeDC and CalibDAC values from HGCROC2 test session 1. https://readthedocs.web.cern.ch/display/HGCELE/2020/07/14/ProbeDC+and+CalibDAC+values+from+HGCROC2+test+session+1. Last accessed: 20 November 2020.

[43] A. Menshawy. *Deep Learning By Example*. Packt Publishing Limited, 2018. URL: https://www.oreilly.com/library/view/deep-learning-by/9781788399906/.

[44] S. Mittal. A Survey Of Architectural Techniques for Managing Process Variation. *ACM Computing Surveys 48(4)*, 2 2016. URL: https://www.researchgate.net/publication/285926063_A_Survey_Of_Architectural_Techniques_for_Managing_Process_Variation.

[45] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/.

[46] C. Olah. Neural Networks, Manifolds, and Topology. http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/. Last accessed: 08 January 2021.

[47] P. Paulitsch. The Silicon Sensors for the High Granularity Calorimeter of CMS. Technical report, CERN, 06 2020. URL: https://arxiv.org/abs/2002.11449, doi:arXiv:2002.11449v3.

[48] T. Quast. *Qualification, performance validation and fast generative modelling of beam test calorimeter prototypes for the CMS calorimeter endcap upgrade*. Dissertation, RWTH Aachen University, Aachen, 2020. URL: https://publications.rwth-aachen.de/record/792901, doi:10.18154/RWTH-2020-06473.

[49] M. Rieger. *Search for Higgs Boson production in association with Top Quarks and decaying into Bottom Quarks using deep learning techniques with the CMS experiment*. PhD thesis, RWTH Aachen University, 6 2019. URL: https://publications.rwth-aachen.de/record/763526.

[50] A. W. Rose. *The Level-1 Trigger of the CMS experiment at the LHC and Super-LHC*. PhD thesis, Imperial College London, 5 2009. URL: http://www.hep.ph.ic.ac.uk/~awr01/thesis/thesis.v1.1.dbl.pdf.

[51] J.-B. Sauvan. Updates on FE data reduction studies, 01 2020. URL: https://indico.cern.ch/event/879499/contributions/3714002/attachments/1978368/3293454/20.01.30_FE.studies.summary.pdf.

[52] W. H. Smith, G. Wrochna, et al. The TriDAS Project: The Trigger Systems. Technical report, CERN, 12 2000. URL: https://cds.cern.ch/record/706847?ln=en.

[53] J. Sonneveld. Commissioning and first results from the CMS phase 1 upgrade pixel detector, 2018. URL: https://arxiv.org/pdf/1807.08987, arXiv:1807.08987.

[54] T. Speer, R. Frühwirth, A. Strandlie, T. Todorov, and M. Winkler. Track reconstruction in the cms tracker. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 559:143–147, 04 2006. doi:10.1016/j.nima.2005.11.207.

[55] A. Steen. The CMS HGCAL detector for the HL-LHC upgrade. 2017. URL: http://cds.cern.ch/record/2293145.

[56] A. Straessner. Overview of the Calorimeter Readout Upgrades. Apr 2018. URL: https://cds.cern.ch/record/2314636.

[57] M. Tanabashi et al. Passage of particles through matter. *Phys. Rev. D 98*, 030001, 2018. Updated 2019. URL: http://pdg.lbl.gov/2019/reviews/rpp2018-rev-passage-particles-matter.pdf.

[58] Texas Instruments. *INA250 36-V, Low- or High-Side, Bidirectional, Zero-Drift Current-Shunt Monitor with Precision Integrated Shunt Resistor*, 04 2015. Revised Dec. 2015. URL: https://www.ti.com/product/INA250.

[59] D. Thienpont. HGCROC2/2A Datasheet. Technical report, CERN, 2 2019. URL: https://readthedocs.web.cern.ch/download/attachments/131957941/HGCROC2_datasheet_v1_draft.pdf?version=1&modificationDate=1580552871000&api=v2.

[60] V. Vlădăreanu et al. Detection of Anomalous Behavior in Modern Smartphones Using Software Sensor-Based Data. *MDPI*, 03 2020. URL: https://www.mdpi.com/1424-8220/20/10/2768.

[61] R. Wigmans. Calorimetry. *Scientifica Acta 2*, No. 1, 18 – 55, 2008. URL: http://lappweb.in2p3.fr/~chefdevi/Detector_reports/Calorimetry/Wigmans.pdf.

[62] Xilinx. *AXI Reference Guide*, 01 2012. Version 13.4. URL: https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf.

[63] R. Yohay. The CMS High Granularity Calorimeter for High Luminosity LHC. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 958:162151, 2020. Proceedings of the Vienna Conference on Instrumentation 2019. URL: http://www.sciencedirect.com/science/article/pii/S0168900219306023, doi:https://doi.org/10.1016/j.nima.2019.04.105.

[64] L. Zhang, D. Bailleux, A. Bornheim, and R.-Y. Zhu. Performance of the Monitoring Light Source for the CMS Lead Tungstate Crystal Calorimeter. *Nuclear Science, IEEE Transactions on*, 52:1123 – 1130, 09 2005. doi:10.1109/TNS.2005.852661.

Danksagung (Acknowledgement)

An dieser Stelle möchte ich mich bei allen bedanken, die zur Verwirklichung dieser Arbeit beigetragen haben.

Zunächst möchte ich mich bei Prof. Dr. Michael Wick für die Betreuung dieser Arbeit bedanken. Ihre freundliche Art und Unterstützung bereitete mir eine sehr angenehme, unkomplizierte und freie Arbeitsatmosphäre. Ich hoffe Sie haben genau so viel Freude beim Lesen dieser Thesis wie ich beim Schreiben hatte.

On the same note, I would like to thank Dr. André David Tinoco Mendes for his excellent supervision during my time at CERN. You were always there when I had questions, provided me with interesting new ideas and sparked new inspiration when needed. Your guidance had a major impact on the development of this thesis. I am very appreciative of having you as a supervisor.

Thanks to Dr. Thorben Quast and Dr. André David Tinoco Mendes for their valuable feedback and support in our weekly meetings. It was a very nice experience to discuss the results of the concept studies in Ch. 5 and to explore new ideas together.

The following people gave valuable comments to various parts of this thesis draft:
Dr. André David Tinoco Mendes reviewed the entire thesis and provided feedback to structure, content, and grammar. He improved the overall quality of the draft thesis significantly. Dr. Thorben Quast has reviewed chapters 1–3 and 5–6 and gave stimulating feedback on structure and scientific accuracy. Especially his comments to introduction, summary and the particulars on the theory of neural networks proofed to be invaluable. Dr. Arnaud Steen reviewed Ch. 4. His comments on technical details helped to clarify many aspects of the testing setup. Thanks to Dr. David Barney for reviewing chapters 1–4 of this thesis and providing constructive criticism on content and language. Your input helped to make the statements of this thesis more clear and accessible.

The chip testing efforts would not be possible without the collaboration of many people at CERN, DESY, FNAL, LLR, and the University of Minnesota. The weekly meetings on software development and DAQ and FE electronics provided valuable insights about the internal workings of many developed subsystems.

For helping in the preparation, the execution, and the analysis of chip tests I'd like to thank Dr. Shilpi Jain, Dr. Arnaud Steen, Dr. Artur Lobanov, and Dr. André David Tinoco Mendes.

At last, I would like to thank Yu Han Chao for providing 3D models and illustrations for Chs. 3 and 5. 感謝您一直以來的支持和愛，親愛的。