THE UNIVERSITY *of* EDINBURGH

Radioisotope Identification with Neuromorphic Methodology: Different Solutions and Evaluations

Shouyu Xie

Doctor of Philosophy

THE UNIVERSITY OF EDINBURGH

2024

To the one who lifted me up from my low,

Ruona

Abstract

Early detection of radioisotopes plays an increasingly important role in the modern world. It allows the possibility of quick countermeasures when faced with potentially hazardous radioactive materials like dirty bombs, and nuclear leakage. This could secure the lives of the innocent in populated areas including airports, stadiums or ports. A light-weight compact handheld device could be used in this situation for the patrol team. However, the operating hours for these devices are normally constrained by the batteries they carry. More efficient algorithms or solutions are needed for this resource-constraint application to extend the battery life so that security patrol is not frequently interrupted by the recharge.

Event-based processing is a novel technique that allows the computing unit to operate only when there is a key event while staying idle otherwise. Spiking neural network (SNN) is a promising candidate for event-based processing and also known as neuromorphic methodology due to the biomimicry plausibility, which could be easily implemented and still offer comparable accuracy to its counterpart — artificial neural network (ANN), which is notoriously power-hungry.

In this research work, it will be demonstrated that using SNN for radioisotope identification (RIID) is possible and capable of achieving the same or even better accuracy when compared with ANNs. Meanwhile, the power consumption of the proposed method on a field programmable gate array (FPGA) shows that power reduction is highly significant compared with the old software implementation on a smartphone.

The task has been delivered in two parts, we first attempted an unsupervised Spike-Timing-Dependent Plasticity (STDP) SNN implementation on SpiNNaker, an emulation platform for SNN. This demonstrates the capability of classifying radioisotopes using purely SNN compatible training methods and architecture.

We then managed to implement a more complex bin-ratio ensemble SNN (BESNN) on FPGA with better performance. To achieve this implementation, a new SNN conversion method was created to facilitate the digital hardware implementation. This conversion flow allows the highly sparse weight matrix representation without sacrificing overall accuracy. In the meantime, the power consumption of the mentioned design has been characterised, which could be used to estimate the battery life of a handheld system while functioning.

Even though this design has been validated on an FPGA, further squeeze for the power saving is possible if an application specific integrated circuits (ASIC) could be delivered. Furthermore, the analogue unit used in the design is a compromise given that the logarithm could not be done by a spiking neuron at the moment. This prevents an end-to-end application, which is preferred for higher integration and potentially more power conservation.

According to our knowledge, applying neuromorphic methodology to address RIID represents uncharted territory, especially in the context of power characterisation, an aspect that has not been explored previously. This research work fills the gap that is present in the research field and also offers a functional low-power prototype for the handheld RIID device producer.

This project pioneers the use of an event-based processing algorithm for radioisotope identification, marking a significant advancement in the field. Leveraging Spiking Neural Networks (SNNs) on specialised hardware, the project establishes a comprehensive application flow, showcasing the efficacy and potential of SNNs in this domain.

The implementation of an unsupervised STDP algorithm for radioisotope identification is also groundbreaking, introducing a local self-learning rule for complex tasks beyond handwritten digit recognition.

Additionally, the bin-ratio ensemble project achieves remarkable accuracy, setting new benchmarks in the field. It represents the first ensemble SNN application in radioisotope identification, further enhanced by an innovative ANN-SNN conversion method with iterative pruning to reduce computational overhead.

Furthermore, this research provides detailed insights into sparse SNN construction and characterises hardware implementation, shedding light on power and energy consumption considerations.

Lay Summary

Historically, human beings have experienced multiple radiological disasters like the Chernobyl nuclear power plant meltdown (World Nuclear Association, 2022), Fukushima Daiichi plant incident (World Nuclear Association, 2023) etc. It has been well established that immediate excessive exposure to radioisotopes can lead to lethal consequences. Early detection of these hazardous radioisotopes can potentially save people's lives by great extent. This is why the security in populated public places like an airport or a big stadium is normally equipped with handheld devices for radioisotope identification.

A company called Kromek Group PIC, based in Durham, UK, is specialised in the development of these mobile devices. However, the problem they are facing with the current design is that it drains the battery very quickly. This has motivated this project to develop a more power efficient alternative to the solution they provide now.

The main objective for this study is to develop a prototype for this specific task, radioisotope identification, which can prove to improve the power efficiency without sacrificing the performance. This big objective mainly includes the following steps:

- Exploring the software solution for high performance that also does not require too much computation
- Implementing that software into a hardware to achieve faster processing speed and therefore increase the throughput
- Characterising the new solution's power consumption and make sure it fits the target

Artificial intelligence is a big part of this study. This provides us with a great methodology that does require complicated mathematical analysis of the input data, which normally burns out a lot of energy. Nevertheless, they can still offer a high confidence of accuracy if enough data is provided for the training process. Within this big category, there is one computing paradigm called spiking neural network, which is an event-based processing approach. As the name suggests, it is event-driven, which will only compute when a significant event occurs. This can potentially save power but also retain a satisfying performance.

By operating our research, we concluded that the application of spiking neural networks for this task is feasible. It brings a resilient model that requires limited computation for the task, which only takes 0.334 ms for one single piece of analysis to complete. In the meantime, our solution is immune to noises that are generally found in the data for this task and excels in performance with great accuracy. In terms of power, our prototype demonstrates better than a magnitude of improvement when compared to the original solution.

The results discovered in this research proves that power saving using spiking neural networks can be done without compromising performance. The power estimation by the end of the research shows that the battery life for the handheld device can be probably extended for at least tenfolds. This greatly facilitates the security work with only a single charge of the device.

In summary, this research work introduces a new solution for radioisotope identification at low power with high accuracy. This can potentially strengthen the hazardous material control and reduce the risk of being exposed to radioactive isotopes and protect the public from terrorism.

Acknowledgements

First I would like to thank Dr Alister Hamilton, for being beyond supportive. He has been there for me at my high and low, whenever I needed a talk. His presence is like a real father figure to my academic life.

This work has also not been possible without the loan of a SpiNNaker board from the University of Manchester and their prompt technical support at the online user group.

A great tribute to the late Professor Yannis Goulermas, Doctoral supervisor of Siru Zhang, who sadly passed away in May 2022. His significant contributions paved the way for this piece of research work.

Additionally, I want to thank Dr. Cristian Sestito for the expertise offered on FPGA characterisation.

This project is supported under the grant from the Defense Threat Reduction Agency (DTRA).

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Shouyu Xie

Contents

Figures and Tables

Figures

Tables

Nomenclature

λ	Wavelength of the photon
ν	Electromagnetic wave frequency
τ_m	Membrane potential decay constant
c	Speed of light
E_K	Reversal potential of the potassium channel
E_L	Leakage reversal potential
E_b	Binding energy from the nucleus
E_{e^-}	Kinetic energy of the photoelectron
E_{Na}	Reversal potential of the sodium channel
g_K	Conductance of the potassium channel
g_L	Leakage conductance of the neuron membrane
g_{Na}	Conductance of the sodium channel
h	Planck constant
I_C	The current flow through the capacitor
I_K	The current flow through the potassium channel
I_L	Leakage current for the neuron model
I_{Na}	The current flow through the sodium channel
V_{re}	Recovery variables in Izhikevich model

Chapter 1

A history lesson

Human beings have been using nuclear materials for different things. There are successful applications like cancer treatment and nuclear power plants. But as a powerful tool, nuclear materials need to be carefully managed so that no radiation hazard would happen and pose a strong threat to the user or any innocent people involved. On the other hand, radioactive materials could also sometimes be involved in terrorism and war due to the potential massive destruction it could bring.

1.1 Chernobyl accident

On April 26, 1986, a tremendous disaster happened at a nuclear power plant in Chernobyl, Ukraine. This was caused by the poor operation during a test at low-power, which led to the explosion of the number four RBMK reactor (World Nuclear Association, 2022). The reactor building crashed, graphite blocks, also known as the moderating material caught fire. As a result, huge amount of radioactive elements were released into the air, including uranium, iodine, strontium and caesium. 28 people died within a few weeks from exposure to a high level of radiation in a short time, which is called acute radiation syndrome (ARS).

The whole town of Pripyat nearby the power plant was evacuated in 36 hours (International Atomic Energy Agency, 2016). This was subsequently followed by another 67,000 people's relocation within few weeks. The accident caused the contamination of an area of 150,000 square kilometres with 30 kilometres around the power plant considered "exclusion zone". This has resulted in the abnormal level of radiation being detected around Ukraine, Belarus and Russia even in some Scandinavian countries due to the scattered radioactive fallout.

Among all the emitted radioisotopes, the most dangerous ones are iodine, strontium and caesium, specifically $^{131}\mathrm{I}$, $^{90}\mathrm{Sr}$ and $^{137}\mathrm{Cs}$. Each has the half-lives of 8 days, 29 years and 30 years respectively. This would mean that Chernobyl is still lingered with $^{90}\mathrm{Sr}$ and $^{137}\mathrm{Cs}$ until this day. This accident has been labelled as level 7 by the International Nuclear Event Scale (INES) according to International Atomic Energy Agency. This is the highest level of safety significance.

1.2 Fukushima Daiichi accident

On 11 March 2011, a great earthquake with a magnitude of 9 hit Great East Japan (Tohoku). This was followed by a tsunami, which was generated by the main shock from the earthquake. (World Nuclear Association, 2023) The combination of the earthquake and tsunami caused the shutdown of the power supply to the cooling system that circulates around the Fukushima Daiichi reactors. This consequently resulted in the meltdown of 3 reactors in the Fukushima Daiichi Nuclear Power Plant operated by Tokyo Electric Power's Company (TEPCO) in 3 days because of the overheat on fuel rods. A considerable amount of radiation was released into the air, which marked this event at the same level as the Chernobyl accident by the scale of INES. Event though no direct radiological death was caused, one death from lung cancer 4 years later was confirmed related to the radiation this accident released (Ministry of Health, Labour and Welfare, 2019).

After the accident, evacuation orders were made by Fukushima prefecture with a radius of 2 Km around the power plant. This radius was later extended by the prime minister to 20 Km with a no-fly zone of 30 Km. Around 47,000 people left their homes in the following few days (Encyclopaedia Britannica, 2023).

Similar to the Chernobyl accident, the released radionuclides mainly are iodine and caesium. They are $^{131}\mathrm{I}$, $^{134}\mathrm{Cs}$ and $^{137}\mathrm{Cs}$ with half-lives of 8 days, 2 years and 30 years respectively. An approximate estimation in June 2011 reported the radiation release as being 770 PBq.

Huge amount of water was used to cool down the power plant after the accident, this left the water contaminated and has been sealed up in the tanks. Efforts to treat the wastewater were made to remove the radionuclides in the water. The initial solution including US proprietary adsorption and French conventional technologies was designed to remove caesium in 2011. These plants were able to drop the caesium radiation from 55 MBq/L to 5.5 kBq/L. In early 2013 the Advanced Liquid Processing System (ALPS) was implemented to further remove the remaining 62 radioisotopes. In 2014 a strontium removal system was added, it could further clean up the water processed by ALPS. In 2021 Japanese government confirmed the processed water would be released into the Pacific Ocean with the endorsement from International Atomic Energy Agency (IAEA).

Because of the technical shortcomings, tritium could not be removed in the process using ALPS. This has sparked heated opposition and protests to the decision to release processed water (Wong, 2023).

1.3 Dirty bombs

Dirty bombs are also known as radiological dispersal devices. They are a type of radiological weapon that infuses the radioactive material with the conventional explosives. The main aim of this type of weapon is to disperse the radiation around an area with radioisotopes. Even though there is not any successful attack, attempts have been made as a measure of terrorism.

In 1996, a dirty bomb that was made up of explosive and radioactive $^{137}\mathrm{Cs}$ was found in Moscow's Izmailovo Park (Anonymous, 2022). This was later reported to be set up by the rebels from Chechnya. Fortunately no one was hurt and the bomb was defused before detonation. 2 years later, a container of unidentified radioactive materials was discovered attached to an explosive mine near the railway station in Chenchnya (Krock & Deusser, 2003). The dirty bomb was safely defused. It was believed to be planted by the Chechen rebel group, whose leader was found out to be associated with the incident in 1996.

In 2002, an American national was arrested in Illinois for the suspicion of planning a dirty bomb attack in an American city. The investigation showed the connection between Al Qaeda and the gang he previously was in. A similar case was logged in the United Kingdom 2 years later (Carlisle, 2007) where a British citizen was charged of conspiracy to commit murder for planning terrorist attacks in the United States and the United Kingdom.

The most recent discussion arose amid the Ukraine war where Russia claimed Ukraine could use a dirty bomb during the war and it should be considered as nuclear terrorism. However, given the big impact a dirty bomb could do to either side's troops or the land, military analysts are saying it is unlikely either side will be using one.

In most cases, successful making of a dirty bomb would require radioactive materials. They could be found in a lot of cancer treatment facilities in the hospital or institutes for the purpose of research. Careful management of these radioactive materials is crucial to prevent nuclear accidents like this.

1.4 Illicit radioactive material trafficking

Despite the fact that there is highly tightened management of radioactive materials, hundreds of reports of these items missing are still being logged each year. There are 352 incidents of nuclear and other radioactive material outside regulatory control logged between 2020 and 2021 alone, with an average incidents of 176 incidents each year (James Martin Center for Nonproliferation Studies for Nuclear Threat Initiativeve, 2023). In these incidents, 43.5% of the

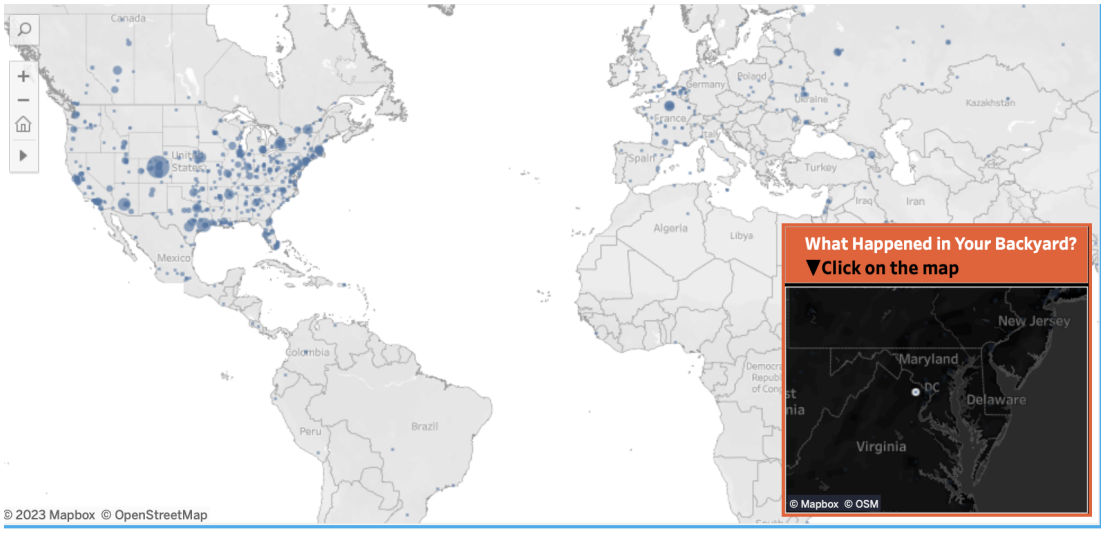


Figure 1.1: Incidents around the world (CNS for NTI, 2023)

incidents involved the radioisotopes that are highly suitable for dirty bomb making. There is a drop of incidents in the whole year of 2022 with 146 cases of illegal activities involving nuclear and other radioactive materials (International Atomic Energy Agency, 2023). But it aligns well with the average incidents counted.

Normally these trafficking incidents would be associated with dirty bomb assembling since it could be the only source of radioactive materials for making this powerful weapon.

1.5 Early detection

For cases like Chernobyl and Fukushima, it is crucial to make sure the radiation level of a certain area is safe for civilians to live. If not, what radioisotopes there are in these areas so that a rough estimation can be made to see how long it will take for the area to be habitable again. As for cases like dirty bombs and illicit trafficking, early detection of radioactive materials like $^{137}\mathrm{Cs}$ and $^{131}\mathrm{I}$ could prevent mass destruction from a powerful weapon and secure innocent people's life from terrorism.

Identifying these hazardous radioisotopes is a challenging task and it is also what this work explores. The thesis will be organised as follow:

- Chapter 2 will briefly explain radiation, how to capture the radiation.
- Chapter 3 reviews the radioisotopes identification (RIID) algorithms.
- Chapter 4 simply describes the objectives and motivation of the project.
- Chapter 5 discusses event-based processing and how it could be used for RIID.
- Chapter 6 reports an unsupervised training of spiking neural networks for RIID.
- Chapter 7 illustrates the supervised approach for spiking neural networks for RIID.

- Chapter 8 shows the FPGA implementation for the RIID algorithm.
- Chapter 9 analyses the results obtained from the work.
- Chapter 10 concludes the thesis by summarising the results and providing the scope for further work.

Chapter 2

Radiation: physics, radioisotopes and detection

To understand how RIID works, it is fundamental to know what radioisotopes are and the physics behind them. This is because they are invisible to human eyes, what sensors could be used to capture the motion of them is the basis of the whole research.

2.1 Radioisotopes

2.1.1 Natural isotopes

Radioisotopes of the same elements share the same number of protons but with different neutrons. They exist in nature in different forms. Mainly three types of them are observed:

- Primordial radioisotopes such as uranium and thorium. They were created at the astronomical events like supernova explosions. Most of them decayed quickly, only the ones with very long half-lives could be observed.
- Radiogenic radionuclides. They derive from the decay of primordial isotopes.
- Cosmogenic isotopes. These isotopes are generated by cosmic rays at the atmosphere.

These are the naturally occurring radioactive materials (NORM), excessive exposure to these isotopes also poses strong threat to human body. The atomic structures of the example

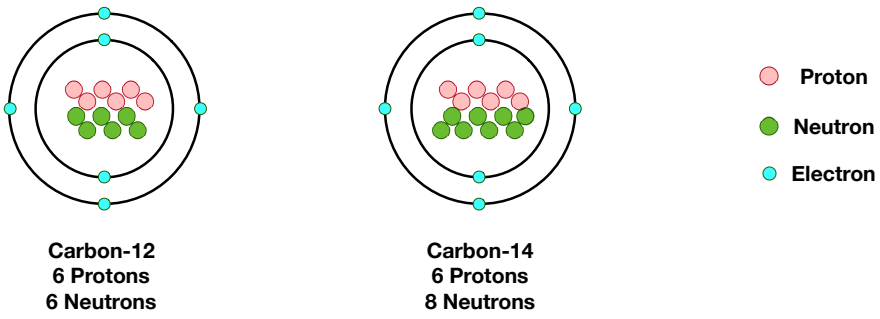


Figure 2.1: Example radioisotopes of ^{14}C and ^{12}C

isotope ^{14}C and ^{12}C are shown in Figure 2.1. It could be seen that the difference between these two is that ^{14}C has 2 extra neutrons than ^{12}C. The index number 14 and 12 represent the atomic mass, which is the sum of number of protons and neutrons.

2.1.2 Anthropogenic isotopes

There are also artificial radionuclides, also known as anthropogenic isotopes. They are normally created by nuclear reactor or nuclear fission. Nuclear fission is a process where the

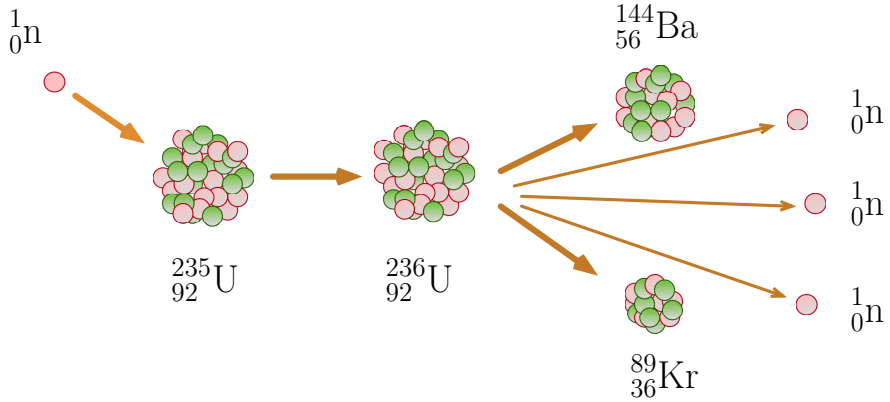


Figure 2.2: Nuclear fission of ^{235}U

nucleus of an atom break down into smaller nuclei under a certain condition (normally neutron collision). During this process a considerable amount of energy is released. An example of nuclear fission is shown in Figure 2.2 where $^{235}_{92}U$ briefly morphed into $^{236}_{92}U$ when bombarded with 1 neutron. But because $^{236}_{92}U$ is unstable, it splits into two smaller atoms $^{144}_{56}Ba$ and $^{89}_{36}Kr$. In the meantime, 3 more neutrons are released. This will lead to the chain reaction to create more lighter atoms.

$$^{1}_{0}n + ^{235}_{92}U \rightarrow ^{144}_{56}Ba + ^{89}_{36}Kr + 3^{1}_{0}n \tag{2.1}$$

This nuclear fission process could be summarised in Equation 2.1. The amount of atomic mass and number of protons remain the same on both sides of the formula. Radiation happens during the decay stage, i.e. when the unstable isotope splits and releases lighter isotopes.

2.2 Radiation

As mentioned before, radiation happens during the decay stage where a heavier isotope splits into smaller isotopes. Depending on different type of decay and isotopes involved, there are different radiations:

- Alpha ray, mainly made up of 2 neutrons and 2 protons (just like a helium atom without electrons)

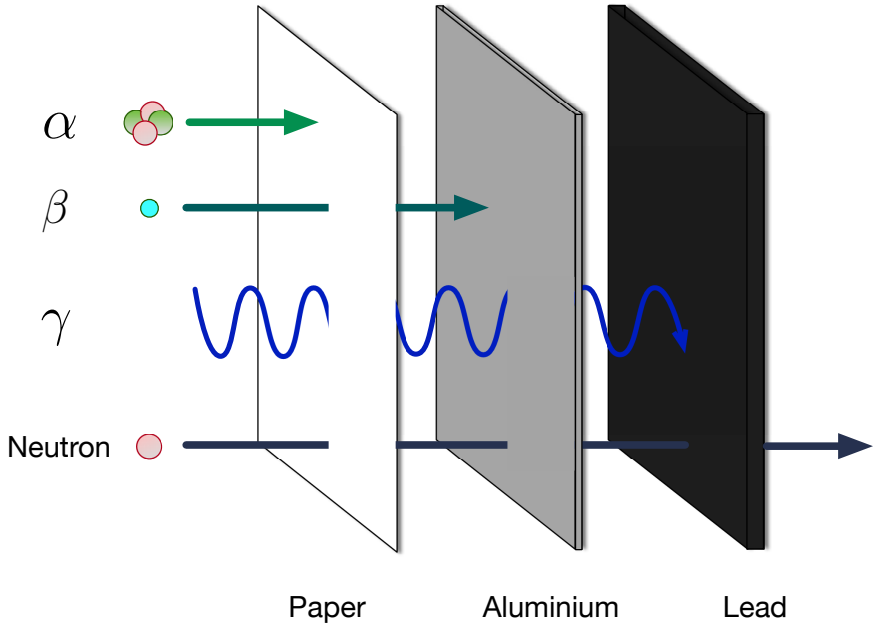


Figure 2.3: Shielding for different types of radiation

- Beta ray, which consists of electrons
- Neutron radiation, consists of neutrons, rarely observed
- Gamma ray, a type of electromagnetic radiation. X-ray and gamma rays are both electromagnetic waves. They are a type of radiation that has the shortest wavelength, therefore they have a lot of energy.

They require different shielding materials to prevent the leakage of the radiation. This is because each type consists of particles with different mass and energy. For example, an alpha ray could be blocked by a sheet of paper due to the mass of alpha particles. They are the heaviest radiation particles, therefore it makes them easier to be stopped.

Figure 2.3 shows that each radiation has different penetration capabilities. α particles could simply be stopped with a piece of paper. β particles could penetrate paper but not through aluminium. Gamma ray, on the contrary, could not be stopped until lead. Neutron particles are the strongest among them all, and could penetrate through even lead. They could only be blocked by thick concrete or water (Occupational Safety and Health Administration, 2023).

2.3 Detection

It has been well-established that radiation (mainly gamma radiation) could pose severe harm to the human body. Early detection of radioisotopes plays a key role in securing people's life. Depending on different scenarios, different detectors could be used.

2.3.1 Geiger counter

Geiger counter or Geiger-Muller counter is a type of simple instrument that is capable of detecting radiation. It is commonly seen in daily life and some theatrical programmes due to its simplicity and cheap cost. It makes clicking noises when detecting radiation. The more noises it makes, the higher doses of radiation there are.

The Geiger counter itself is simply made up of a sealed up tube, which is filled with noble gas and an anode in the centre as shown in Figure 2.4.

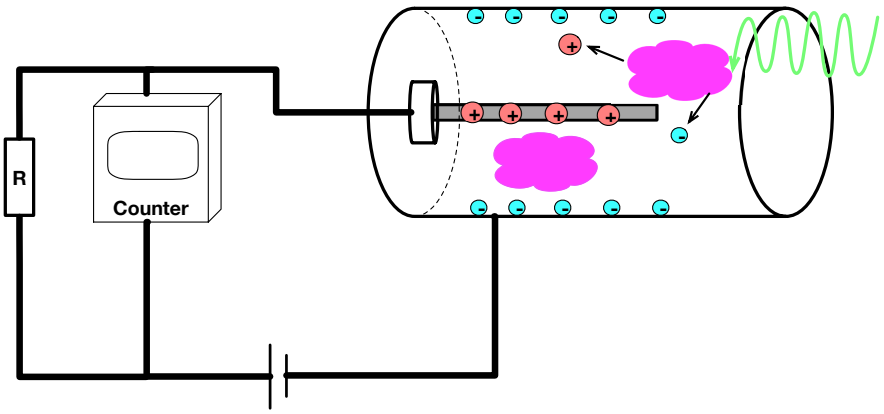


Figure 2.4: The structure of a Geiger counter

Just like mentioned above, the tube is filled with noble gas and tube itself is negatively charged. The rod or anode in the centre is positively charged. The external circuitry has a counter and a huge resistance connected in parallel. Since the tube is filled with noble gas, there will be no reaction when there is no radiation. When the tube receives ionising radiation, electrons on the noble gas atom will be ionised to leap off its orbit. This process will create an ionised pair with a free negatively charged electron and a positively charged atom.

Since the wall on the tube is negatively charged and the rod is positively charged, the newly created electron and atom will be drawn towards the rod and wall respectively. To get the charge neutralised, they need to travel back to the power supply. But the resistance is really huge, most of them will need to go through the counter. The counter will then be able to read out the number or give clicking noises based on the current (United States Nuclear Regulatory Conmission, 2020).

This is the simplest tool to detect ionising radiation, but it could only give a rough estimation of the radiation intensity. That is to say, this tool will not be able to give an analysis of the energy resolution. It could not discriminate against the type of radiation. Therefore, radioisotope identification is not possible with this tool. Besides, this tool needs a period of time to "reboot" between consecutive measurements of high pulses of energy. Therefore, it could not detect a high rate of radiation because of the downtime it needs. That is because the tube needs a very high voltage applied and the ionisation could cause a large avalanche so that the wall needs to recharge before there are enough electrons again.

2.3.2 Ionisation chamber

Similar to a Geiger counter, the ionisation chamber also works as a gas-filled detector but it has a much lower voltage (less than 1000 V) applied to the electrode so that the charge collected will only reflect the primary ionisation.

There is a non-linear correlation between the energy collected and the voltage applied between the electrodes in a gas-filled detector (Tsoulfanidis & Landsberger, 2021). This could be shown in Figure 2.5.

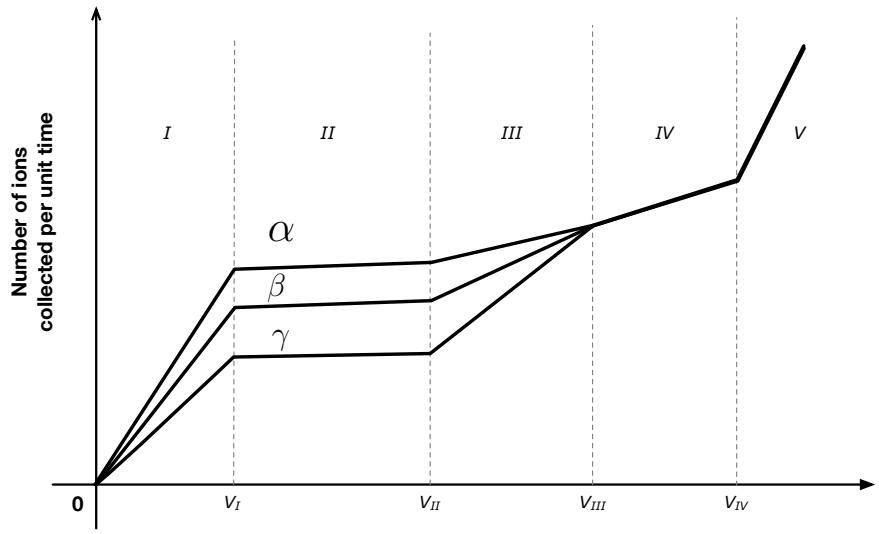


Figure 2.5: The relationship between the charge collected and the voltage applied

In the first region, it could be seen that the charge collected will increase when the voltage is increased. This is called the recombination region where the generated ion-pairs: ions and electrons have a possibility to recombine to become a stable atom again. This is because the voltage applied could not create a strong enough electric field to pull away the electrons. Therefore, the collected ions only represent part of the ionisation.

The ion chamber works in region II where the collected charge remains unchanged. While in region II, the voltage applied is strong enough to pull away ion pairs before they recombine. Since there is no ionisation multiplication (secondary ionisation), it is possible to calculate the particle energy using this tool.

While in region III, the collected charge will start increasing again. That is because the voltage applied provided a very strong electric field so that the part of the electrons created during the first ionisation start to trigger a secondary ionisation. This phenomenon is called ionisation multiplication. The ratio of the total ionisation and primary ionisation is independent of the amount of the primary ionisation for a certain applied voltage. Therefore, in region III, it almost looks like a linear correlation between the total charge collected and voltage applied.

The region IV is where a Geiger-Muller counter works. In this region, the electric field is so strong that a single pair of ion particles is capable of generating a whole avalanche of ionisation whose amount is independent of the primary ionisation. This allows the Geiger counter to be very sensitive to the radiation in the surroundings.

Since in region II, the collected charge is only dependent on the primary ionisation, the ion chamber is able to give an accurate measurement of the radiation. Therefore, ion chambers are normally placed in medical treatment such as cancer treatment for patient dosimetry. They could also be used in the monitoring station for airborne radiation detection.

2.3.3 Semiconductor detector

Semiconductors could also be used for radiation detection. Such detectors are also known as solid-state detectors. The way it works is fairly easy with the knowledge of semiconductor physics.

Consider two differently doped types of semiconductor materials P type and N type are formed in a junction like in Figure 2.6. The P-type semiconductor is doped with impurities that offer extra holes as positively charged carriers while N-type semiconductor has impurities that create extra electrons as negatively charged carriers. While connected in a junction, the extra carriers will diffuse from the higher concentration to the lower concentration. For example, holes will diffuse from left to right while electrons will diffuse from right to left. As a result, free electrons from the donor impurity atoms in N-type material migrate across the newly formed junction to fill the holes in the P-type material, resulting in the production of negative ions. Because the electrons have moved across the PN junction from the N-type silicon to the P-type silicon, they leave behind the positively charged donor ions. The holes from the acceptor impurity move across the junction in the opposite direction into the region where there are large numbers of free electrons. This leads to the fact that the charge density of the P-type along the junction is filled with negatively charged acceptor ions, and the charge density of the N-type along the junction becomes positive.

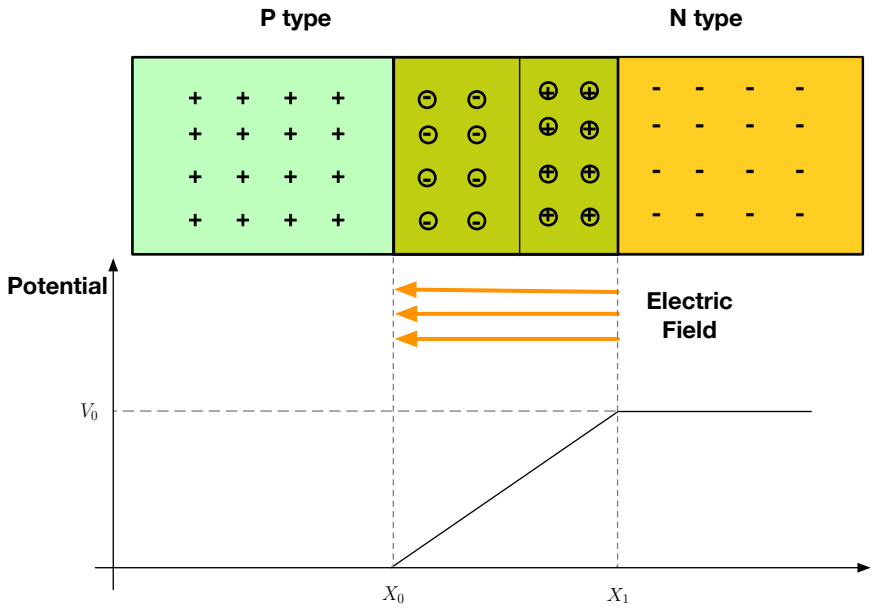


Figure 2.6: A functional P-N junction

As a result, at the junction in between two materials, an electric field is formed. This will keep the diffusion from expanding as the donor atoms repel the holes and the acceptor atoms repel the electrons.

Take electrons as an example, they diffuse from N-type material to P-type because of the density gradient. When they reach the other side, the combination will take place to fill the holes.

This consequently leaves the region in the P-type material with negatively charged ions thus forming an electric field from N-type to P-type. It stops more electrons from travelling across and eventually reaches an equilibrium state where the region remains unchanged and the free carriers in this region are totally depleted.

By using this structure a semiconductor-based detector could be built. Figure 2.7 shows an exemplary semiconductor-based radiation detector. It has been concluded that in a P-N junction, there is an internal electric field formed by the carriers on both sides. When the ionising radiation reaches the depletion region, an ion pair is created with an electron and an ion. Under the electric field, these two particles will be swept to the respective boundary. This will lead to the charge change on both sides which could be recorded by external electronics.

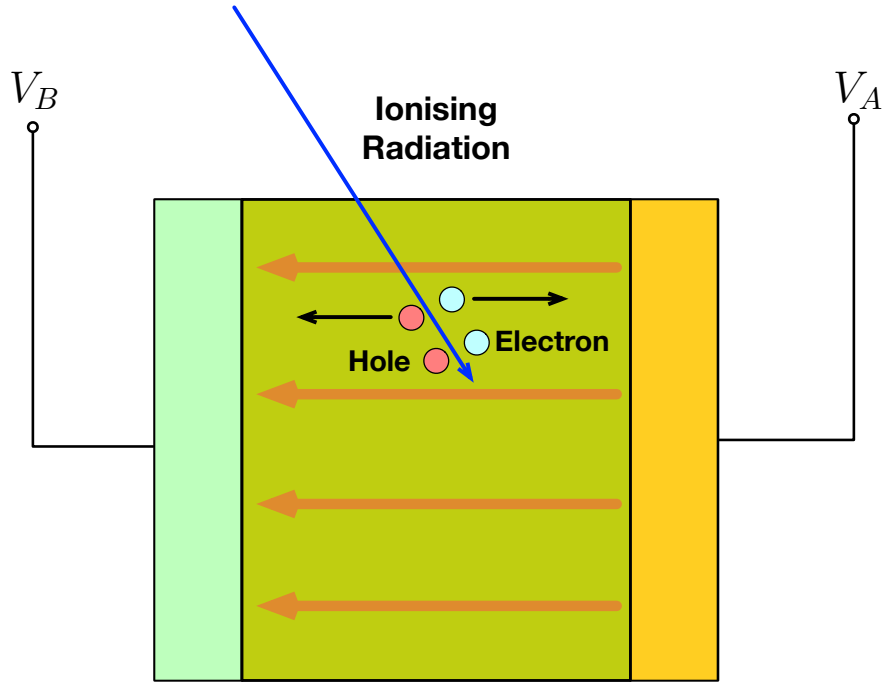


Figure 2.7: Solid state radiation detector

This type of detector acts much like an ion chamber with parallel plates except the conducting material is not gas but semiconductor. Compared to a gas-filled radiation detector, they excel in energy resolution. That is to say, they have better discrimination against different radiations with different levels of energy. The second big advantage is that they have a very fast response time. This is because the depletion zone is very thin, with proper external electronics, detection could normally be finished in order of $10^{-7}s$. Another superior quality is that they are very compact, therefore the whole sensor can be fabricated on a micro-scale.

2.3.4 Scintillation detector

The third category of radiation detectors is the scintillation detector. The core of this type of detector is a type of material called a scintillator. They are able to produce visible light or photons when there is incident radiation. The material itself could be inorganic, organic or gaseous. From another perspective, a scintillator could be in the form of solid, gas or liquid.

The very first observation of this phenomenon was in 1903 by William Crookes, who used ZnS screen with alpha particles (Leo, 2012). However, the manual light counting process requires the assistance of a microscope and was abandoned for its inefficiency back then. It was not properly applied until 40 years later with the help of a photomultiplier tube (PMT), an additional equipment to amplify the light. The typical scintillator materials include sodium iodide with impurities of thallium (normally denoted as NaI(Tl)), caesium iodide with thallium (ScI(Tl)), calcium iodide with sodium (CaI(Na)) etc.

How it works

The rough stages for a scintillation detector could be as follows:

1. Radiation absorption and initial light emission.
2. Photocathode emits electrons when exposed to light.
3. Electrons multiplied by the photomultiplier.
4. Electronics outputs the pulse height of the collected electrons at the scale.

The detection system with a scintillator is shown in Figure 2.8.

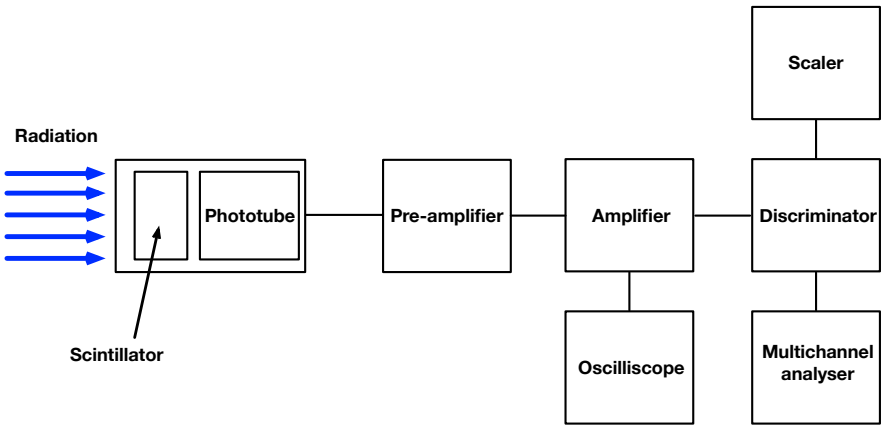


Figure 2.8: A system diagram of a scintillation detector

Scintillator

The photon emission process by the scintillator could be explained with the help of solid-state physics. Considering a typical inorganic crystal scintillator NaI(Tl), it has the allowed and forbidden energy bands depicted as in Figure 2.9.

There are allowed and forbidden energy bands in a crystal, and the electrons and holes can only exist in the allowed band. No electrons and holes are allowed in any states in the forbidden band. As for the allowed bands, the uppermost band is called the conduction band where the electrons can move freely and that is why the materials are conductive. Just below the conduction band in the energy band structure of a material is called the valence band where electrons are normally present at absolute zero temperature, which is normally full or nearly full.

In general cases, electrons and holes need to stay at the position where the total energy is the lowest, and that explains the vacancies in the conduction band. When there are impurities introduced into the crystal, there will be localised impurity centres formed at a certain energy. They will have a smaller forbidden band than the crystal and have excited states that are lower than the conduction band but really close to it. Meanwhile, they also have the ground state that is higher than the valence band but really close to it.

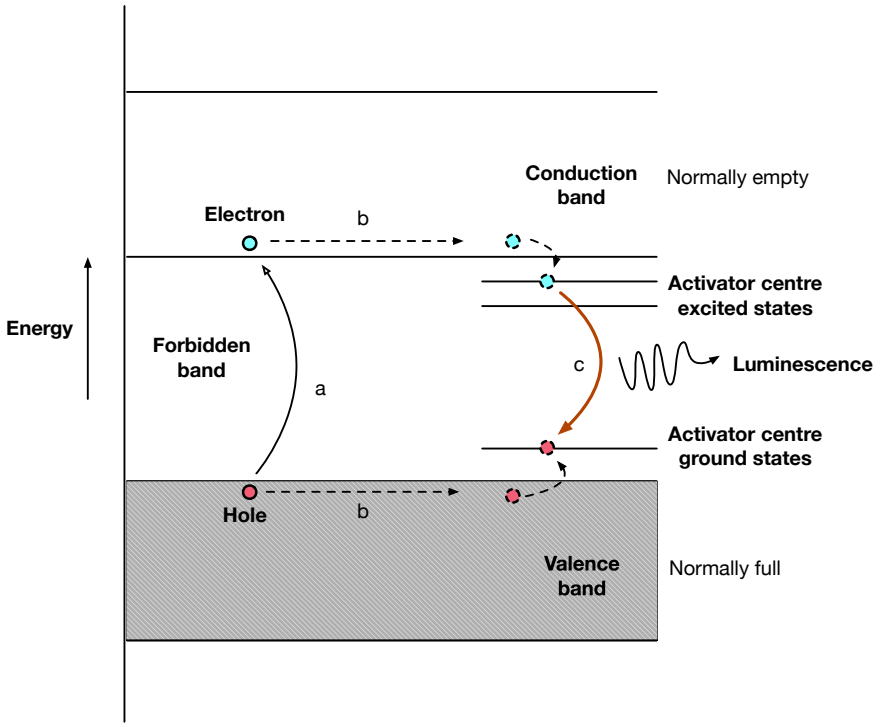


Figure 2.9: Allowed and forbidden bands of a crystal

When the crystal receives incident radiation, the electrons in the valence band will be excited to leap into the conduction band. Therefore, it creates an electron-hole pair, namely *exciton*. In this situation, the electron will need to drop back to the valence band so that the stable state will be restored. However, due to the quantum mechanics, such dropback is inefficient. But the electron could move freely in the conduction band, and the hole could have the same mobility in the valence band. When they travel near the impurity centre, the tendency to drop from the conduction band to the impurity centre excited states is much higher. A similar case also applies to the hole. When there are electrons in the excited states of the impurity centre, it is much more likely to de-excite from the excited states to the ground states rather than take the big forbidden band gap. When this de-excitation happens, the photons with the corresponding energy will be released and therefore the luminescence could be observed (Hilger Crystals, 2023).

To summarise, the scintillation could only occur when these happen:

a. Ionising radiation excites the electron to leap to the conduction band and form an electron-hole pair.
b. Electrons and holes travel near the impurity centre and ionise the impurities.
c. De-excitation from the excited states to the ground states at the impurity centre and emit a photon.

Photomultiplier tube (PMT)

However, as mentioned before, the light coming from the scintillator is so weak that is not particularly observable. The advent of the photomultiplier tube addressed this problem and facilitated the development of the scintillation detector.

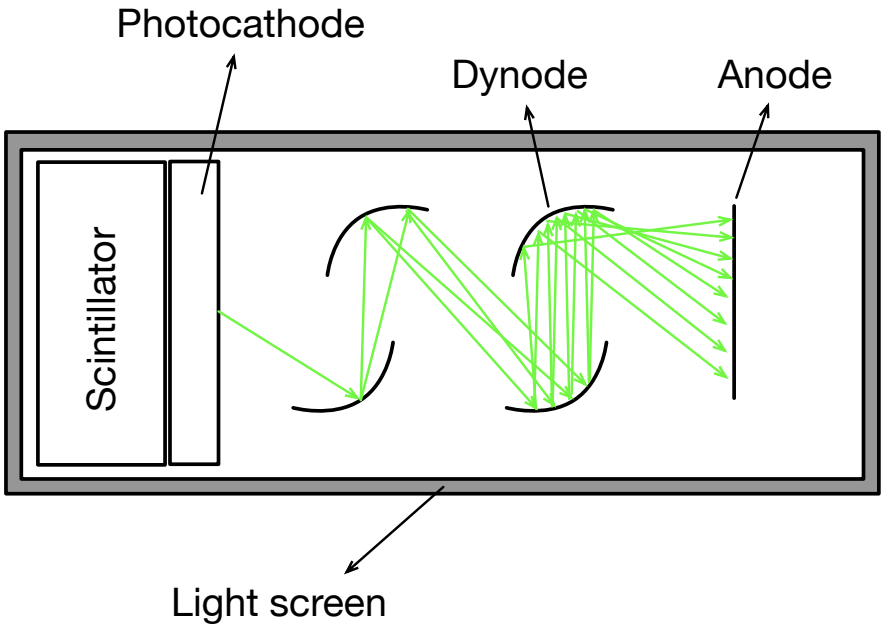


Figure 2.10: An exemplary phototube

An exemplary photomultiplier is shown in Figure 2.10. The main purpose of a photomultiplier is to amplify the signal since the initial light given by a scintillator is too weak to observe.

It could be seen from the figure that the photocathode is closely coupled with the scintillator, it is designed to convert the photon into photoelectrons. There are also dynodes arranged following a pattern to reflect the photoelectrons. These dynodes are coated with a type of material that is able to emit secondary electrons when impinged with electrons. By the end of the tube, there is the anode to collect all the photoelectrons, the signal will then be converted into an electric signal to be processed by the later analyser.

When there is an incident radiation towards the scintillator, it will produce weak visible light. This light travels to the photocathode to be translated into rays of electrons. Under the guidance of the electric field created by the high voltage, emitted electrons strike against the first dynode, which triggers more electrons. These electrons will then be guided towards the next dynodes and repeat the process until they reach the anode. To avoid unnecessary interference from the light outside the tube, the whole tube is covered with the light-reflective material as a shield.

The amplification theoretically gives an exponential increase towards the initial signal. In a commercial photo tube, there are normally 15 dynodes along the way and the amplification could reach as high as 10^7 (Photek Limited, 2021).

Advantages and disadvantages

As a type of widely used radiation detector, it has multiple advantages:

1. High sensitivity and detection efficiency.
2. It has a fast response time so it is possible to detect a short-lived radioactive event.
3. A high range of energy measurement could be provided by the detector, it could also be used to detect different types of radiation.
4. Good energy resolution is given by the detector, which is crucial to discriminating energy peaks in a spectrum.
5. The sensor gives a fairly linear response in terms of the energy the charged particles carry.

Besides the good features mentioned above, this type of detector has good external electronics like multichannel analysers to provide a good spectroscopy analysis.

However, this type of detector also has some drawbacks:

1. The material used as the scintillator is susceptible to ambient factors such as temperature and humidity.
2. Organic scintillators tend to degrade during use.

Due to the good electric and photoelectric characteristics, scintillators have been widely used for spectroscopy analysis. This has been well-established as a standard for commonly used gamma-ray spectrometers.

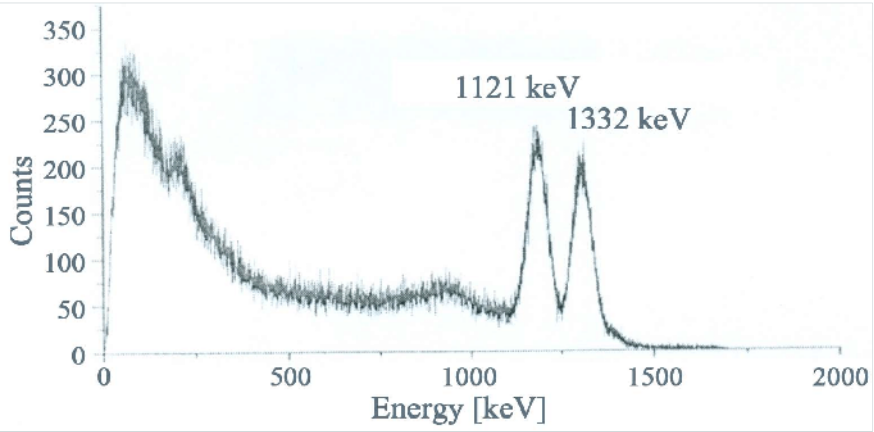


Figure 2.11: Example spectrum for ^{60}Co, figure from (Selim et al., 2013), available by the licence.

2.4 Gamma-ray spectroscopy

It was mentioned in the previous section that different types of radiation could be prevented by certain levels of protection. However, the gamma-ray is highly penetrative and easier to observe compared to the neutron radiation. Detection of the radionuclides using gamma-ray spectroscopy is effective and easy to implement.

Normally, one radioisotope will only give off the gamma-ray with characteristic energy. This could be well analysed in a spectrum where the horizontal axis is different energy bins and the vertical axis represents the intensity or "pulse height". For example, a typical $^{60}\mathrm{Co}$ will have a spectrum that has two big peaks at 1.1 KeV and 1.3 KeV as shown in Figure 2.11.

Thanks to this feature, gamma-ray spectroscopy has become the basis of radioisotope identification. Basically, all the radioisotope identification algorithms have been built upon the analysis of the gamma-ray spectra. The review algorithms shall be reviewed in the next chapter.

2.5 Conclusion

In this chapter, we have reviewed the basic physics of radioisotopes and radiation. This includes the atomic structure of radioisotopes and how they can be generated. After explaining the mechanism of radiation, the types of radiation were illustrated.

Following this, different radiation detection methods were discussed. This includes the simplest device like Geiger counter, which can only give a rough estimation of the radiation intensity but no source analysis. There are also ionisation chambers, which work similarly to a Geiger counter. Depending on the different voltages applied, it can operate in different regions for different applications. There are also semiconductor detectors or solid state radiation detectors which employ the properties of semiconductor materials for radiation detection. It has very good sensitivity and energy resolution, but the peripherals needed for it to operate are sophisticated and it is very costly to build these detectors. The most prevalent detectors are scintillation detectors, where illumination is given during ionisation radiation. They are cheap to produce, the related circuitry has also been systematically developed.

With the detectors ready, the analysis of the radioisotopes can be conducted by operating Gamma-ray spectroscopy, where the Gamma-ray energy spectra are studied to give a conclusion about the presence of certain radioisotopes.

Chapter 3

Radioisotope identification (RIID) algorithms

Radioisotope identification is a process to give the constitution of the present radionuclides based on the observation of the spectrum (usually gamma-ray spectrum). These algorithms evolved in terms of autonomy, efficiency and accuracy. Here in this chapter we shall briefly look at the development of these algorithms and discuss the state-of-the-art and future trends in this field.

3.1 The early RIID algorithms

The very early RIID algorithms appear to be more dependent on human expertise. This approach is a more empirical methodology. Based on scientific computation and observation, when a radioisotope decays, there is more than just one common phenomenon happening during detection. Therefore, when looking at a gamma-ray spectrum, some specific characters could be observed to be coupled with a certain isotope.

In general cases, there are three interaction mechanisms in the detection medium that could bring significant impacts on the spectrum. They are photoelectric absorption, Compton scattering and pair production. These mechanisms shall be explained and pointed out in an exemplary gamma spectrum.

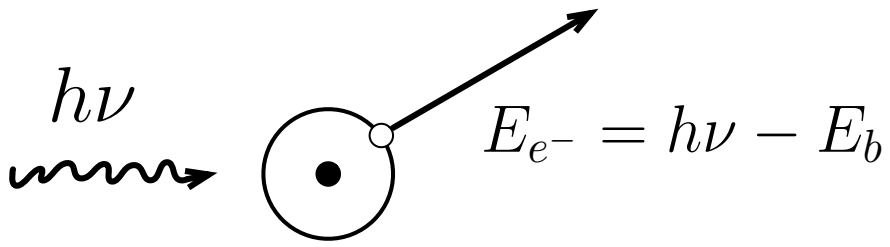


Figure 3.1: Photoelectric absorption

3.1.1 Photoelectric absorption

When the incident gamma-ray photon disappears and all the energy is transferred to the photoelectron in the form of kinetic energy, the photoelectric absorption would happen. This process could be shown in Figure 3.1. It could be seen that the the energy photo carries excites the electron on the shell of the atom so that it could get away from the binding the nucleus is stringing. The kinetic energy the photoelectron carries equals the difference between the gamma-ray energy and the binding energy.

It could be seen in the figure that the incident gamma-ray carries the energy $h\nu$, where h is the Planck constant, ν stands for the electromagnetic wave frequency. E_{e^-} represents the kinetic energy of the free electron and E_b is the binding energy this electron was constrained to.

The escaping electron will leave a vacant spot in the original position but very quickly will be filled by other electrons. This process will release X-ray because of the amount of the energy it liberated is the binding energy. This characteristic X-ray could be observed in 88% of the cases for iodine (Broyles, Thomas, & Haynes, 1953). But this X-ray will not travel long before it is reabsorbed through photoelectric interaction by other electrons that are less tightly bound.

Therefore, the photoelectric absorption process is the liberation of a photoelectron that carries most of the gamma-ray energy and some characteristic X-ray excited photoelectrons. The sum of the kinetic energy of these two types of photoelectrons should equal to the original gamma-ray energy if there is no more other escaping. And the corresponding spectrum should look like the one presented in Figure 3.2 where only a delta function is observed. Along the two axes of the figure, the horizontal axis illustrates different levels of electron kinetic energy represented by **E**, while the vertical axis demonstrates the whole spectrum from a statistical point of view as in number of absorption events per energy channel. This explains the annotation of **dN**/**dE**.

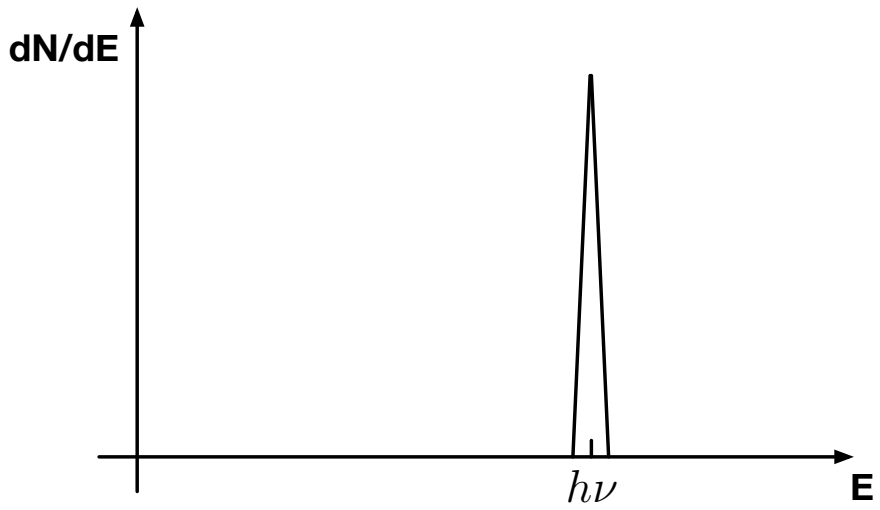


Figure 3.2: The gamma spectrum when there is only photoelectric absorption

3.1.2 Compton scattering

Another more complicated case is when the incident gamma-ray is reflected by the electron but shares part of the energy to the photoelectron during this process. This type of interaction is called Compton scattering. Figure 3.3 gives a simple graphical illustration.

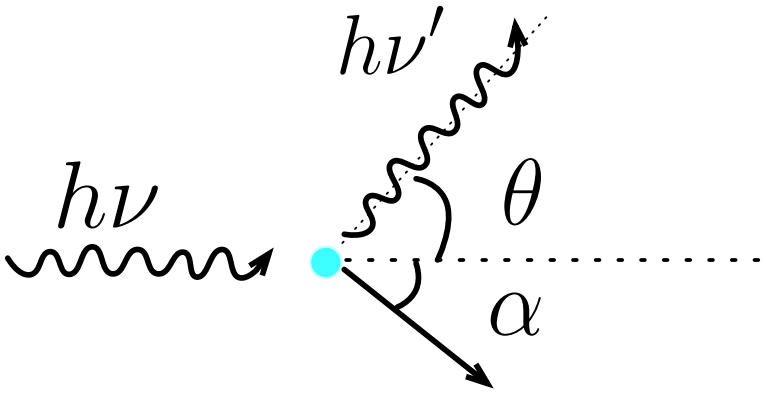


Figure 3.3: Compton scattering

The new scattered energy $h\nu'$ is dependent on the angle θ it gets scattered from the incident direction. Details on the derivation of the scattered energy will be omitted here. According to the theorem of conservation of momentum and energy, the Compton equation could be given as:

$$\lambda' - \lambda = \frac{h}{m_0 c}(1 - cos\ \theta) \tag{3.1}$$

Where λ gives the corresponding wavelength of the incident gamma-ray photon, λ' gives the scattered photon wavelength. The speed of light is denoted by c, and the mass of the electron is m_0.

Meanwhile, by definition of particle wavelength:

$$c = \lambda \nu \tag{3.2}$$

We could obtain the scattered photon energy as:

$$h\nu' = \frac{h\nu}{1 + \frac{h\nu}{m_0 c^2}(1 - cos\ \theta)} \tag{3.3}$$

this would leave the recoiled electron with energy:

$$E_{e^-} = h\nu - h\nu' = h\nu \cdot \frac{\frac{h\nu}{m_0 c^2}(1 - cos\ \theta)}{1 + \frac{h\nu}{m_0 c^2}(1 - cos\theta)} \tag{3.4}$$

Equation 3.3 shows that the scattered photon energy is dependent on the scatter angle θ, so is the excited photoelctron. There are two extreme cases:

- When $\theta \cong 0$, $1 - cos\ \theta \cong 0$, the whole equation 3.4 gives the conclusion that $E_{e^-} \cong 0$.
- when $\theta = \pi$, the photon was bounced back to the opposite direction of the incident, and the recoiled electron will travel in the direction of the incidence. This is the maximum energy that could be transferred to the electron.

In the cases of the maximum energy transferred, we could have the scattered photon energy in:

$$h\nu' \mid_{\theta=\pi} = \frac{h\nu}{1 + \frac{2h\nu}{m_0c^2}} \tag{3.5}$$

the recoil electron energy in:

$$E_{e^-} \mid_{\theta=\pi} = h\nu \cdot \frac{\frac{2h\nu}{m_0c^2}}{1 + \frac{2h\nu}{m_0c^2}} \tag{3.6}$$

There will be a gap between the max recoil energy and the incident gamma energy and this could be calculated by:

$$E_C = h\nu - E_{e^-} = h\nu' \mid_{\theta=\pi} = \frac{h\nu}{1 + \frac{2h\nu}{m_0c^2}} \tag{3.7}$$

Reflected by the gamma-ray spectrum, it should give a Compton continuum along the axis of energy with the max value of E_{e^-} and minimum value of essentially 0. This is due to the fact that the value for θ ranges from 0 to π and this could be seen as in Figure 3.4. The

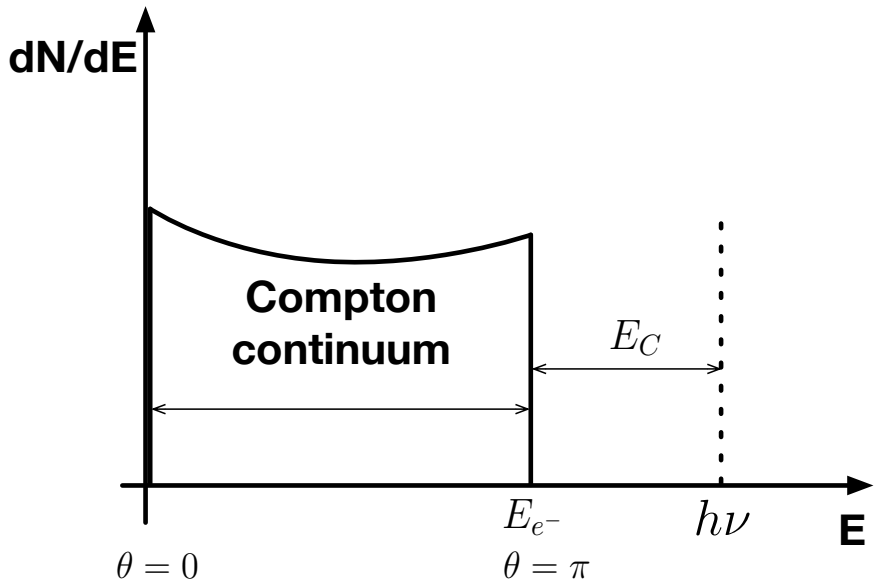


Figure 3.4: Compton continuum in a gamma-ray spectrum

upper and lower limit of this detectable energy is a continuous distribution and is called the Compton continuum with the upper limit called Compton edge when $\theta = \pi$. In the case where $h\nu \gg m_0c^2$, this energy gap is basically a constant of $\frac{m_0c^2}{2}$, which is around 256 keV.

3.1.3 Pair production

Another significant interaction is when the incident gamma-ray created a pair of electron-positron at the cost of the disappearance of the initial gamma-ray. According to the conservation of energy, the incident gamma-ray energy equals the sum of the base creation energy ($2m_0c^2$) and kinetic energy of the electron-positron pair. This could be expressed as:

$$h\nu = (E_{e^-} + E_{e^+}) + 2m_0c^2 \tag{3.8}$$

These two photoelectrons will not travel far before they lose their kinetic energy and get reabsorbed by the detector. Therefore, this could be shown in the gamma-ray spectrum as a delta function of the sum of the kinetic energy.

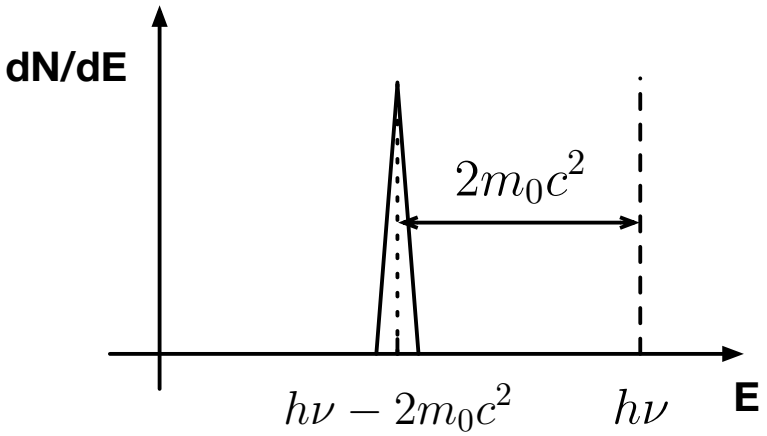


Figure 3.5: Gamma-ray spectrum for pair production

This resembles the spectrum given by photoelectric absorption but at a different energy, which is $h\nu - 2m_0c^2$.

However, a positron is highly unstable. When its kinetic energy is very low, a process called annihilation will happen where the positron will combine with the electron and both the positron and the combined electron disappear. This annihilation will give off the energy of these two particles energy in gamma-ray ($2m_0c^2$).

3.1.4 Expert interpretation

In an average-sized detector, an expert can compare the distribution of the gamma-ray spectrum combined with the position of these characteristic energies to conclude the presence of radioisotopes. Hypothetically, in an average medium-sized detector, when an incident gamma-ray hit the surface, there are effects coming from all the three interactions mentioned. Besides, there are even secondary interactions made possible from the primary Compton scattering or annihilation. This could be illustrated in Figure 3.6.

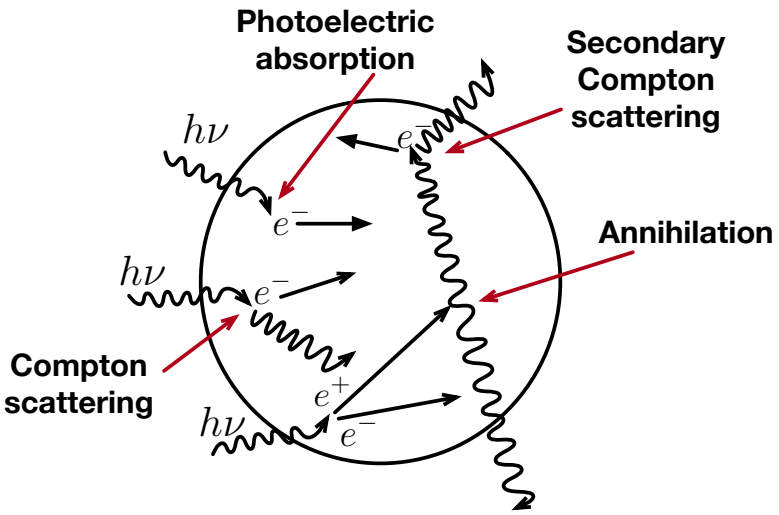


Figure 3.6: Possible particle interactions depiction

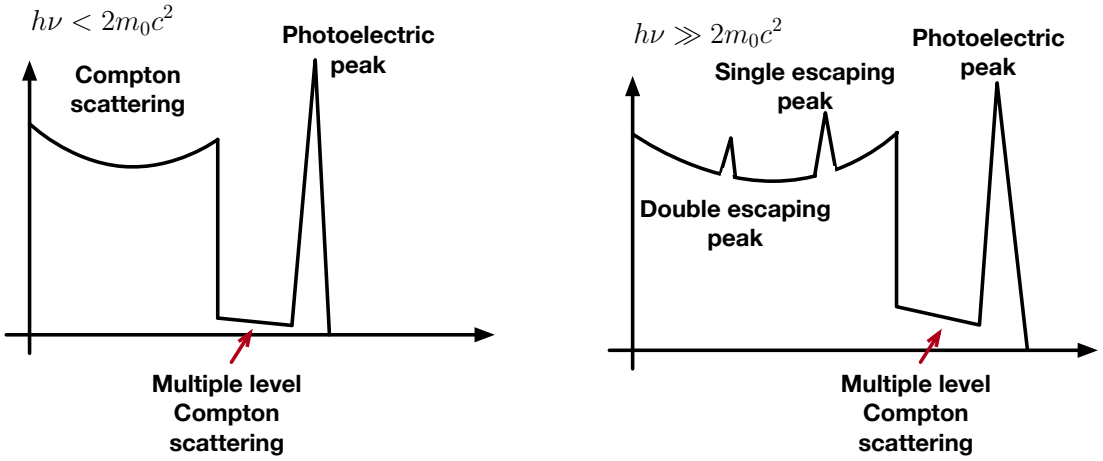
From the figure, it could be observed that all three types of interactions could happen. When the photoelectric absorption happens, the incident gamma-ray is entirely detected. Compton scattering will also happen in this case where a Compton continuum will be formed in the response function. The condition for pair production to happen is when the incident energy is at least one time higher than the rest mass energy.

Other than the photoelectric absorption, other types of interaction may produce "escaping photons" where secondary radiation may exist and trigger more interactions. This has resulted response function being continuous along energies between the Compton edge and incident energy.

There could also be the case when pair production is possible, the annihilation creates two escaping radiation that may or may not be absorbed depending on the position of this production. When two escaping radiation are not trapped, there shall be a "double escaping peak" spotted at $h\nu - 1022keV$ at the energy axis. It is also possible when there is just one escaping radiation and the other one is totally absorbed. This will produce a "single escaping peak" spotted at $h\nu - 511keV$.

To conclude, two cases of the response functions can be shown in Figure 3.7.

Take the example from a real gamma spectrum of $^{137}\mathrm{Cs}$, which is displayed in Figure 3.8. It could be seen that the character energy photopeak is located at 662 KeV, which is the character gamma energy from $^{137}\mathrm{Cs}$, because this energy is much lower than the required mass energy there is no pair production. But there is a Compton continuum with a clear plateau displayed with the Compton edge located around 250 keV lower than the photopeak. In the meantime, character X-ray and backscatter could be observed. This is due to the interaction that happened in the shield (normally lead).



(a) No pair production **(b)** Pair production is possible

Figure 3.7: Response function when incident energy lower and greater than twice the rest mass energy

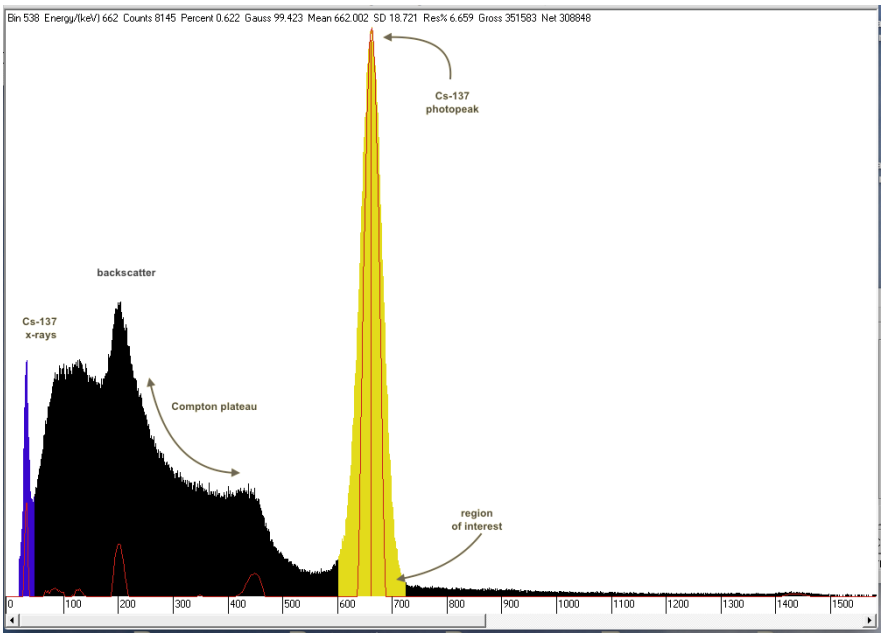
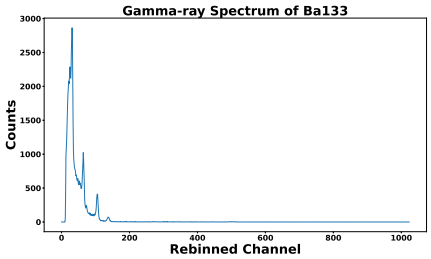


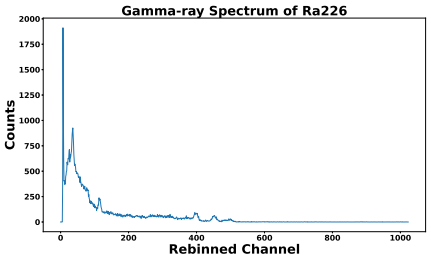
Figure 3.8: A real gamma spectrum of ^{137}Cs (available from Gammaspectacular)

A well-trained expert is able to interpret these peaks along the spectrum and give a fair conclusion of what isotopes might have been detected. Certain character energy bins could always be a strong indicator. For example 662 keV for ^{137}Cs, 1.12 MeV and 1.33 MeV for ^{60}Co.

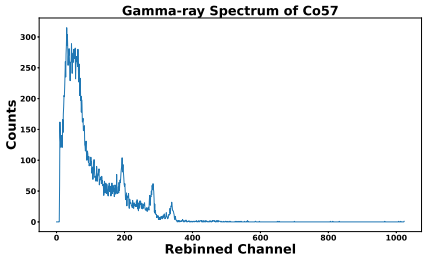
Here we present some spectra from our dataset where different patterns can be observed for different radioisotopes in Figure 3.9. One thing that needs to be declared is that the horizontal axis here is a different channel and it has been rebinned into 1024 channels.



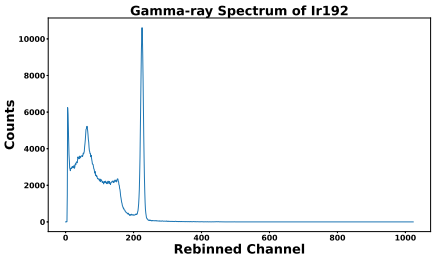
(a) Gamma-ray spectrum of Ba133



(b) Gamma-ray spectrum of Ra226



(c) Gamma-ray spectrum of Co57



(d) Gamma-ray spectrum of Ir192

Figure 3.9: Exemplary spectra from our collected dataset

3.2 More automated algorithm

What has been mentioned in the last section is still fairly manual therefore fast response for the evaluation towards a given spectrum is not possible. Besides, manual work has much less reliability compared to an automated algorithm running on a computer.

3.2.1 Rule-based identification

Region of interest (ROI) method

By employing the method, region of interest, the algorithm will primarily check the most prominent peak in the gamma spectrum and statistically fit this peak with Gaussian distribution to identify the centre of the gravity. This has been closely studied and implemented since a very early stage. In 1969, a software programme employing photopeak method was implemented in the language of Fortran IV called SAMPO was proposed by Routti (1969). The SAMPO program offers the flexibility of both automatic off-line and interactive on-line analyses on the computer CDC-6600. It incorporates algorithms for calibrating line shapes, energy, and efficiency, along with routines for peak search and fitting. Various options are provided to adapt the code for precise nuclear spectroscopic tasks and routine data reduction alike. As the continuation of this piece of research work, microSAMPO was introduced (P. Aarnio,

Routti, & Sandberg, 1988) in 1988 as the revision for SAMPO on IBM personal computers. It could perform peak search and centroid determination using generalised second differences. Radionuclide identification was also included by comparing the identified gamma peaks to the library of gamma ray.

In the same year, an interactive system, RICKI, was developed to perform the spectrum analysis (Proctor, 1988) for cases when there is one or more peaks or multiplets. This application seems to be used for the analysis and evaluation of Three Mile Island (TMI), where an incident happened in 1979. RICKI was created for the examination of spectra from severe fuel damage assessments. Two functionalities have been added to enhance the analysis of TMI core bore data: the ability to edit averaged activities and the generation of an output file for creating a spreadsheet.

Template matching

Template matching could be considered as an extension to the ROI methodology. The template approach aims to generate a unique "fingerprint" of a certain item and compare this signature with the template generated using same method from a reference that is known to be authentic (Yan & Glaser, 2015). This template carries the critical information that characterises the original item.

This method was used to not just as warhead verification but also in low resolution gamma-ray spectrometry to characterise radioisotopes (Göttsche, Schirm, & Glaser, 2016). This is a special case of application where the low-resolution of gamma spectra generation is preferred with the assistance of the information barrier, a mechanism to restrict the flow of sensitive data.

To conclude, template matching is a method that compares a segment of or the whole spectrum with the template generated to identify and characterise the isotopes. This is radically different with ROI where only the photopeaks are logged and recorded.

Variable subset selection

Variable subset selection methods operate similarly to the template matching methods. The difference is that variable subset selection method tend to use a varying mixture of isotopes templates from the library to generate a hypothesis, which is then used to compare against the actual spectra.

In an official application called GADRAS (Mitchell & Mattingly, 2009), which is short for Gamma Detector Response and Analysis Software, variable subset selection methods are likely to have been used to build a radiation sources and detector response model. This could be used to predict the response for a customised radiation source with a customised detector. The programme employs mainly the measured gamma spectrum and neutron count rate as the source to identify the radiation sources. The response function is later calculated with the point models of the detector material, dimensions, collimation, and scattering environment.

3.2.2 Statistical methods

Poisson clutter split (PCS)

Poisson clutter split (PCS) is a commercial algorithm that is designed by Physical Science Inc. (PSI). This novel approach is used for detection and radioisotope identification in various radiological backgrounds (Shokhirev et al., 2012). According to the paper, in the detection process with the radiological spectrum at different locations, there are two main noises in the background:

- Background clutter, which describes the changes in the energy-dependent count rate due to the change in isotopic composition because of the locations, weather conditions etc.
- The random radioactive decay process which could be described by Poisson statistics.

The main idea for this approach is to reduce these noises by merging an accurate model of the Poisson processes and a representation of the background clutter into one single framework. The newly created background estimation will then give the verdict of the existence of a certain hazardous signature in the spectrum collected. RIID is also performed in this process by computing the correlation against an authentic set of isotopic spectra library.

This method employs the Generalised Likelihood Ratio Test (GLRT) to compute the possibility for two hypotheses:

- H_0 The collected spectrum shows only the background
- H_1 The collected spectrum presents the combination of the background and a source isotope

The ratio for these two hypotheses' likelihood (H_1/H_0) indicates if there is the existence of a source isotope. If this ratio exceeds a pre-defined threshold, it is likely that an radioisotope is detected.

Machine learning methods overview

Machine learning is a big category with mainly multiple different branches: Here we will nominate just 2 big categories:

- Artificial neural networks (ANNs)
- Other machine learning methods

There have been a significant amount of examples to show that machine learning algorithms could be applied to this specific task. Machine learning algorithms mainly employs a model that is trained on a library of examples, during which the model will learn to associate the features with a specific label. After the training process, the model will be fed the unseen spectra to give a possibility or inferred values for each class. Generally, the class that has the highest values or possibilities will be inferred as the prediction from the model.

As for other types of machine learning methods, there are examples of supporting vector machine (SVM) (Alamaniotis, Lee, & Jevremovic, 2015), principal component analysis (PCA) (Boardman, Reinhard, & Flynn, 2012), k-nearest neighbour (KNN) (Du, Zhao, Zhu, & Tian, 2021), naïve Bayes' classifier (Sullivan & Stinnett, 2015), non-negative matrix factorisation (NMF) (Bilton et al., 2019).

On the other hand, the ANN methods are widely used due to the big breakthrough for the training and accessibility. There are well developed framework to build self-customised ANNs in a really simple manner. Application cases with ANNs for RIID include (Kamuda, Stinnett, & Sullivan, 2017) (Keller, Kangas, Troyer, Hashem, & Kouzes, 1995) (Yoshida, Shizuma, Endo, & Oka, 2002).

Here in the table 3.1 and 3.2, some past application examples are listed.

Citation	Year	Dataset Type	Frame Size	Detector Type	# of Isotopes	Algorithm	Isotope Mixture
Keller and Kouzes	1994	Hybrid of synthesised and measured	N/A	NaI	8	One layer ANN	Yes
Keller et al.	1995	Hybrid of synthesised and measured	N/A	NaI	8	One layer ANN	Yes
Yoshida et al.	2002	Measured only	1 hour	Ge	28	Two layer ANN with peak searched result as input	Yes
Chen and Wei	2009	N/A	N/A	NaI	8	K-L transform combined with linear associative network	Yes
Boardman et al.	2012	Measured only	2-60 s	NaI	12+1 (background)	Principal component analysis	No
Medhat	2012	Measured only	3-18 hours	Ge	7	Three layer ANN giving probability of the existence of an isotope	Yes
Alamaniotis, Heifetz, Raptis, and Tsoukalas	2013	Hybrid of synthesised and measured	1 s	NaI	6	Fuzzy logic RIID	Yes
Alamaniotis et al.	2015	Measured only	3 s	NaI	6	Support vector regression combined with fuzzy logic	Yes
Ninh, Phongphaeth, Nares, and Hao	2016	Measured only	2-30 s	NaI	5+1(natural uranium)	Matrix linear regression	Yes

Table 3.1: Some machine learning applications on RIID

Kamuda et al.	2017	Hybrid of synthesised and measured	Gross $10^3 - 10^5$ counts	Ortec 905-3 NaI	32+1(background)	Three layer ANN giving relative gross count attribution	Yes
Bilton et al.	2019	Measured only	1 s	NaI	2	Non-negative matrix factorisation	No
Hague, Kamuda, Ford, Moore, and Turk	2019	Sythesised only	$6.5 \times 10^2 - 6.5 \times 10^4$ counts	Ortec 905-3 NaI	29	Deep CNN	Yes
Liang et al.	2019	Hybrid of synthesised and measured	10^8 counts	NaI	4	Space-filling curve+CNN	Yes
Daniel, Ceraudo, Limousin, Maier, and Meuris	2020	Hybrid of synthesised and measured	3×10^5 counts	CdTe	6	Ensemble CNN	Yes
Gomez-Fernandez et al.	2021	Synthesised only	N/A	NaI	16	Deep CNN	No
Jeon, Kim, Lee, Moon, and Cho	2021	Synthesised only	1 hour	Polyvinyl toluene (PVT)	8	Multitask learning deep neural network	Yes

Table 3.2: Continued:Some machine learning applications on RIID

Artificial neural networks

It could be seen that there is an increasing amount of citations to support the machine learning applications on RIID. Within the list, the first application (Keller & Kouzes, 1994) could even date back to 1994 with just a simple one-layer ANN. In this work, a combination of a spectrometre and a linear perceptron was proposed where the spectrometre produces 512 channels of values which are directly fed into the output layer with 8 classes. The detectable isotopes are ^{22}Na, ^{54}Mn, ^{57}Co, ^{60}Co, ^{137}Cs, $^{152-154}$Eu, ^{226}Ra and ^{232}Th. There are only limited samples for validation, the efficacy for this simple network is therefore not substantially established.

Later in 2002, a two layer ANN used for RIID was proposed. This is the first time where a proper neural network is used. A simple three-layer ANN (input layer included) was trained in a supervised fashion where the input was taken as the binarised energy bins. Each input neuron stands for a specific energy bin, where the presence of a peak will be highlighted as 1 otherwise the input is 0. It used a peak search method to pick out the high intensity within a smoothed spectrum. By this approach, a spectrum is translated into 47 binary values to be fed into the ANN to give inferences. Even though this piece of work shows the possibility of analysing a mixture of 28 different isotopes, only countable samples were presented. On the other hand, this algorithm's pre-processing limits the end-to-end automation of gamma-ray spectroscopy.

It was first by Kamuda et al. who used ANN systematically to train on the dataset as an approach to identify and quantify the mixture of the radioisotopes. Unlike the approaches used before with peak search as the input, the raw spectra value was used to feed the ANN for training, while the output is expected to be the percentage of each isotopes in the mixture. There are 33 classes in total including the background. In this work, the simple three layer ANN successfully made fairly accurate quantification of the mixture of isotopes in the spectra. It is also surprising to see that this algorithm demonstrated certain extent of resistance towards detectors' variance, which is an actual common issue with the real detector.

Two years later, the group of Kamuda et al. showed the approach of deep convolutional neural networks (CNNs) to tackle RIID in the comparison of adaptive and non-adaptive algorithms (Hague et al., 2019). The CNN in the paper presented a strong distinguishing ability when compared to the non-adaptive algorithm. The non-adaptive template matching algorithms were struggling with ^{192}Lr even with really long dwelling time given. But the CNN was still performing a much better classification by comparison. In the meantime, CNN did a better generalisation despite the difference with the training set.

Since CNNs are normally more applied to 2D images because they employ the local spatial features, attempts to convert spectra into an image have been made so that CNNs could be better at generalisation during training (Liang et al., 2019). In this work, the author used Hilbert curve as a way to generate 2D images from 1D spectra as illustrated in Figure 3.10. The author used this curve to wrap a vector of spectrum with size of 1024 into a matrix size

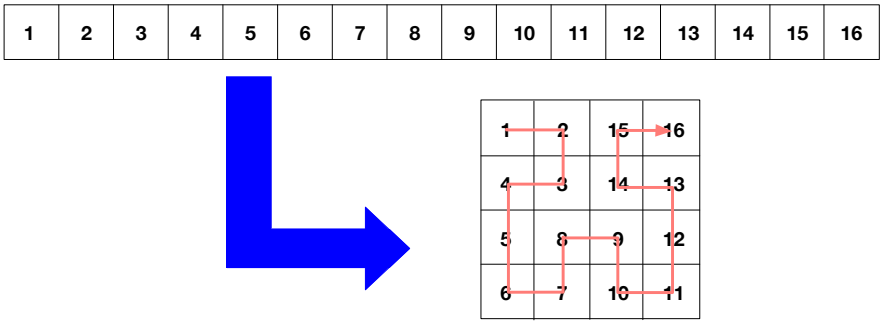
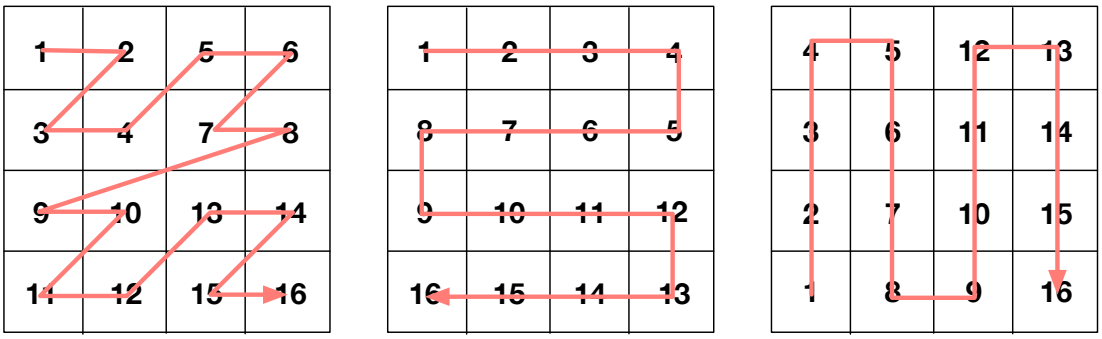


Figure 3.10: Hilbert curve to convert a 1D spectrum to a 2D image

of 32×32 and used it as the input image to the CNN. Hilbert curve have been compared against other types of curve including: z-order, vertical scanning and horizontal scanning. The Hilbert curve demonstrated its superior performance over the other curves during CNN



(a) Z-order curve **(b)** Horizontal scanning curve **(c)** Vertical scanning curve

Figure 3.11: The other curves

training and validation. The trained CNN showed that it is resilient spectrum drift and delivering a stable and higher accuracy over other machine learning algorithms including ANN, k-nearest neighbour (KNN), support vector machine and decision tree.

To better detect the mixture of radionuclides, the idea of using an ensemble was suggested by Daniel et al.. In his work in 2020, he used an ensemble of 6 CNNs with each specialised in one specific source. Each CNN will only report the probability of the source it is assigned to. A probability larger than 0.5 will be considered as detection of the isotopes. The model was

tested against the synthetic data derived from the real measurement. A strong sensitivity was shown in the test where the model could correctly report the presence of the source in 99% cases with equal or less than 0.1 of the sample photons, which is 30,000. Overall an accuracy of 90% could be achieved with simply just 1,000 photons.

Other more advanced type of neural networks were presented for the task as well, which just shows how versatile ANNs could be. One good example is (Jeon et al., 2021) where a variation of multitasking learning (MTL) and weighted multi-head self-attention was applied. These are the features applied in the state-of-the-art model transformer. Two tasks were implemented using the same encoder:

- Full energy peak analysis
- Relative radioactivity prediction

The proposed deep learning module demonstrated strong capability of unfolding full energy peaks and accurate prediction of the relative radioactivity.

Other machine learning methods

Other than neural networks, attempts with other machine learning methods were demonstrated. For example, principal component analysis (PCA), as a commonly used tool for dimensionality reduction, it could also be used for RIID (Boardman et al., 2012). This technique constructs a big covariance matrix based on the training set, which describes the correlation between features in the sample. After finishing Eigen decomposition and deciding the principal components, the new unknown spectrum will be projected into an N dimensional space as a point. In this space, samples are clustered by their label. The Mahalanobis distance will then be calculated to determine which class this new point should belong to. However, this technique requires a considerable amount of computation for the eigenvector decomposition. In the meantime, the approach needs re-bin and intensity normalisation pre-processing for all the samples.

Similarly, non-negative matrix factorisation (NMF) shows another potential way to deal with the same task. NMF is another dimensionality reduction approach like PCA but with a difference data handling. The main idea is to approximate the original sample matrix $X \in \mathbb{R}^{P\times N}$ (P features with N samples) using the dot multiplication of two new sparsely represented matrix W and H so that $X \approx WH$, where W has the dimension of $P \times R$ and H has $R \times N$ and are both constraint to be non-negative. Meanwhile, R should be smaller than N. In general cases, W represents the basis sparse pattern that could be used to constitute a specific sample therefore has P features. While each column in H is the combination to a sample to linearly use the basis pattern in W so that a sample could be reconstructed. In the work proposed

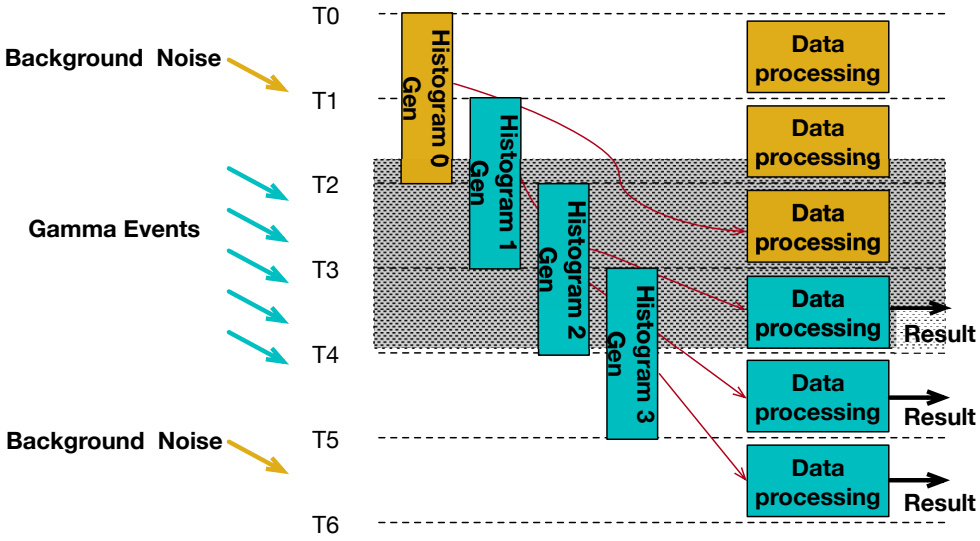
by Bilton et al., this approach was applied to be compared with a standard region of interest algorithm. The mentioned NMF method showed superior performance over the counterpart and demonstrated improvments over PCA. However, this method is more computationally intense than PCA but retains better Interpretability.

Another great example is the fuzzy logic application. Fuzzy logic is a mathematical tool that deals with uncertainty or imprecision in decision making. Classical logic or "crisp" logic has 1/0 to stand for true/false. While fuzzy logic allows the representation of degree of true with a value between 1 and 0. In (Alamaniotis et al., 2013), Alamaniotis et al. used peak search to find the prominent photopeaks and filter out the background fluctuation as pre-processing. All the candidate peaks are later processed against the isotope signature library using fuzzy inference. This approach has proven to be effective and reported comparable detection accuracy by maximum-likelihood fitting. However, this fuzzy logic RIID demonstrated much lower false alarm rate and significantly less execution time. As an extension, the same author proposed the combination of support vector machine and fuzzy logic for RIID (Alamaniotis et al., 2015). Support vector regression was applied following the raw spectrum to enhance the detectability of the following fuzzy logic. This will yield a better quality spectrum for the later stage. The following step is similar to the previous work with peak search and fuzzy inference. This new methodology was compared against multiple linear regression spectrum fitting algorithm. It shows effectiveness of such method by significantly reducing the false detections. Besides, a 13.3% higher sensitivity and 46.8% higher precision were observed.

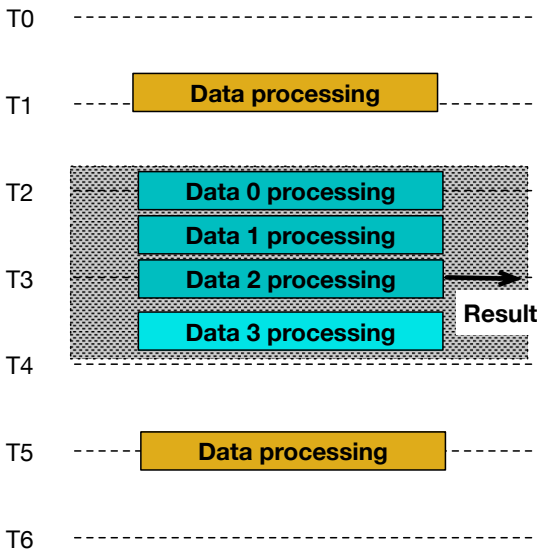
3.3 Event-based processing?

We have reviewed in the last two sections about the RIID algorithms either through expert interpretation or more automated solutions. They are all based on the fact that the inference stage has to happen after a whole frame has to be captured by the multi-channel analyser. In other words, all these algorithms are based on a spectrum captured at a specific time. This spectrum could be seen as an "image" of the accumulation of all the previous radioactivity like a frame in a film. In this sense, all the review algorithms could be considered as frame-based algorithms.

However, just like reviewed in Chapter 2 and earlier this chapter, the nature of a radioactive decay is totally spontaneous and stochastic. Having the calculation on all the time while capturing frames at each time step may not be the most power efficient and fast solution. In terms of the power efficiency, it could be concluded obviously that the algorithm will need to be constantly running at each time step. This would mean that the computation will cost as much energy as the time goes. As for the speed, this is because all the algorithm needs to reach a certain level photo counts to perform well. Yet different isotopes present different levels of activity. This has made the time step determination difficult so that all the frames have



(a) Frame-based processing



(b) Event-based processing

Figure 3.12: Frame-based processing vs event-based processing

to be taken at a specified time period so that there is enough counts or confidence for the following algorithm to process. Imagine a group of 5 different isotopes with various half lives, the time they need to collect good quality of spectrum for analysis are {4s, 5s, 2s, 8s, 12s}. For simplicity, we would need to set up a minimum of 12s as an interval for frames capture so that all the supported isotopes could be detected. Therefore, flexibility and inference speed are compromised for the good quality of data collection.

By comparison, an event-based processing may be ideal for the power and computational efficiency. As depicted in Figure 3.12a, big differences could be observed from the side by side comparison between these two processing. In Figure 3.12a, a pipe-lined frame-based processing is displayed where the data processing will only start when histogram generation finishes. On the left of the figure, the events are denoted as background noise and gamma events. Background noises are not of interest for the RIID while gamma events are the events that characterise the sources. That is to say, temporally, the shaded area stands for the key information time period. But the same computational power is consumed at each time step. Event-based processing, as the name suggests, is a data processing method where computation will only happen when triggered by key events. In Figure 3.12b, event-based processing deal with the data only when there is an event. This has efficiently stocked all the computation during the gamma events period and finish inference early. The processing after result may not even be necessary in this case.

Considering that spectrum generation is essentially the photon receiving and counting, each photon reception could represent an event. This has naturally decided that RIID may be quite ideal to be dealt with using event-based processing.

More details about event-based processing will be discussed in Chapter 5.

3.4 Conclusion

Since Gamma-ray spectroscopy is the main methodology for radioisotope identification, this chapter focused on the interpretation of a Gamma-ray spectrum. Features like peaks and continuity bands were explained and how they may relate to the radioisotopes detected.

Based on these unique features, specialists can operate isotope identification manually based on their expertise. But there are also more automated algorithms to this problem. They are roughly categorised into rule-based and statistical methods, where rule-based algorithms basically use predefined rules to match the patterns to known radioisotopes, while statistical methods leverage mathematical models and algorithms to analyse spectra data and make probabilistic determinations. Rule-based methods like region of interest (ROI), template matching and variable subset selection are introduced. On statistical methods regime, algorithms including Poisson clutter split (PCS) and emerging machine learning algorithms were mentioned. Artificial neural networks (ANN) especially, are the main algorithms used.

But event-based methods can be a promising candidate for this specific problem. This is due to the fact that Gamma-ray spectrum generation is also event-based. One can process when there is an update on the radioisotope event (radioactive decay) and reserve energy when there is no event.

Chapter 4

Motivation and background

We have reviewed the big historical radiological events in and addressed the importance of radioisotope identification in chapter 1. RIID algorithms are reviewed in the last chapter. It could be seen that there is countless effort in the research field for RIID. Kromek Group PLC is also one of them, who is dedicated to develop effective RIID devices for national security.

4.1 A realistic application case

Kromek is a company based in Durham UK that focuses on the development of both hardware and software of cadmium zinc telluride (CZT) solid-state Radiation Detectors and Scintillation Radiation Detector components and products (Kromek Group PLC, 2023). CBRN detection is a big part of their business, which includes chemical, biological, radiological and nuclear threats. Along this line, they have different sectors for various scenarios like security & defence, civil nuclear and biological detection.

Especially in the security & defence sector, applications like area monitoring, border control, emergency services and nuclear power plant management heavily rely on sensitive radioisotope detection devices. They offer products like:

- a drone-based monitoring system, AARM, which stands for airborne radiation monitoring system, which provides detailed quick radiological survey in the air from a safe distance
- a guided radiation interdiction detector (GRID) backpack, where a whole set of radiation detection, identification and localisation system packed in one backpack
- D3S ID, a wearable detection solution with two separate components, a detection sensor and an analysing device
- D5 RIID, a compact wearable all in one radiation detection solution

(a) D3S ID RIID

(b) D5 RIID

Figure 4.1: Two wearable radiation detection devices from **Kromek**

The two wearable devices in Figure 4.1 are powered by battery and they could be equipped on security officers who patrol around populated area like stadiums, airports or nuclear power plants to prevent potential hazard like illicit trafficking, dirty bomb threat or nuclear leakage. They could also be equipped for military use to detection possible dirty bomb use so that corresponding countermeasures could be used before the detonation, which causes irreversible damage to either soldiers or environment.

D3S RIID

While in operation, the security just has to holster the detector in a belt or life-jacket. As for the case of D3S ID RIID, the always-on detector will send over the spectrum over to the handheld smartphone for data processing and inference. It could deliver a strong performance for a quick isotope identification within 30 seconds for library standard of ANSI 42.34. If it is in search mode, it could achieve even faster identification with just 3 seconds. In total, the supported isotope list covers 42 isotopes, which is 22 isotopes more than ANSI 42.34.

From the supported list of radionuclides from Kromek, it could be seen that two powerful detectors are built in the gadget with the ability to detect both gamma and neutron ray for RIID. The reported gamma detector is a typical CsI(Tl) scintillator with the size of 16 cm^3 or 1 in^3. This detector has a detectable range from 30 keV to 3 MeV and a high throughput for the gamma channel at 10,000 cps (counter per second). While the included neutron detector is made from Non-^{3}He, which offers 9 cps in 1 neutron per cm^2. In total, the neutron detector has a slightly lower throughput of 5,000 cps for the neutron channels.

Americium-241	Indium-111	Radium-226
Antimony-124	Iodine-123	Scandium-46
Barium-133	Iodine-131	Selenium-75
Bromine-82	Iridium-192, in various shielding	Sodium-22
Caesium-134	Lutetium-177	Strontium-90 [1]
Caesium-137, in various shielding	Lutetium-177m	Technetium-99m
Californium-252[2]	Manganese-54	Thallium-201
Chromium-51	Molybdenum-99	Thorium-232
Cobalt-57	Neptunium-237	Tin-113
Cobalt-60, in various shielding	Palladium-109	Uranium-235
Europium-152	Plutonium-239	Uranium-238
Fluorine-18[3]	Plutonium, reactor grade in various shielding	Depleted uranium, in various shielding
Gallium-67	Plutonium, weapon grade in various shielding	Highly enriched uranium, in various shielding
Gold-198	Potassium-40	Yttrium-88

Table 4.1: Supported isotope list of D3S RIID (additional 22 isotopes in red)

The communication between the separate detector and the smartphone that runs on Android could be achieved using either micro USB or Bluetooth®. Battery life for the detector itself is around 12 hours while the smartphone runs on a similar if not shorter battery life.

Here we list some key properties of isotopes from ANSI42.34 in Table 4.2, which are essential for identification and spectra analysis.

D5 RIID

As a more compact model with both the scintillator and analyser, D5 RIID has different specifications. This device is more rugged so it could survive a more harsh using environment. Besides, the whole system is powered by a dual battery support which could work independently or in unison. Therefore, this device is more suitable for the military application case where battery life and resilience should be prioritised.

Compared to the mentioned D3S RIID, D5 RIID supports a shorter list of isotopes but still delivers identification of 28 different radionuclides. The library of isotopes offers 8 additional sources than ANSI 42.34. They are ^{152}Eu, ^{18}F, ^{123}I, ^{177}Lu, ^{177m}Lu, ^{99}Mo, ^{237}Np, ^{22}Na.

1. Beta- emitting radionuclide
2. Neutron emitting radionuclide
3. Beta+ emitting radionuclide

Isotopes	Proton	Neutron	Decay mode	Gamma Ray energy (MeV)	Half life
^{241}Am	95	146	Alpha and Gamma	0.0595409	432.60 years
^{133}Ba	56	77	Electron capture	0.0530 0.0796 0.0800 0.1606 0.2232 0.2764 0.3029 0.3560 0.3838	10.51 years
^{57}Co	27	30	Electron capture	0.014 0.122 0.136	271.74 days
^{60}Co	27	33	Beta and Gamma	1.1732 1.3325	5.27 years
^{137}Cs	55	82	Beta and Gamma	0.6617	30.05 years
^{67}Ga	31	36	Electron capture	0.093 0.184 0.300 0.393	3.2617 days
^{131}I	53	78	Beta	0.364	8.03 days
^{192}Ir	77	115	Beta and Gamma	0.136 0.316 0.468 0.604 0.884 1.06 1.378	27 days
^{40}K	19	21	Beta, electron capture and Gamma	1.460	1.25 billion years
^{99m}Tc	43	56	Isometric transition	0.140	6.0067 hours
^{201}Tl	81	120	Electron capture	0.135 and 0.167	73 hours
^{226}Ra	88	138	Alpha	0.186 0.241 0.295 0.352 0.609	1602 years
^{232}Th	90	142	Alpha	0.059 0.081 0.238 0.911 1.290 2.615	14 billion years
^{235}U	92	143	Alpha	0.186	703.8 million years
^{238}U	92	146	Alpha	N/A	4.468 billion years
^{239}Pu	94	145	Alpha	0.080 0.205 0.224 0.302 0.332 0.461 0.650	24.11 thousand years

Table 4.2: Properties of various isotopes from ANSI42.34

This device houses another type of inorganic scintillator called Cerium-doped Lithium Lanthanum Borate Chloride (CLLBC). This is a gamma-neutron detector which could yield scintillation towards both gamma ray and neutron, which made this detector a versatile choice. CLLBC has few advantages:

- High light yield
- Good energy resolution
- Fast decay time

A typical CLLBC gives a light yield of 180,000 photons per million electron volt with neutron ray and provides a light yield of 60,000 ph/MeV in terms of gamma ray (Cieślak, Gamage, & Glover, 2019). This high light emission property outperforms most of the scintillators of its kind.

As for the energy resolution (ER), which is vital to distinguish apart the photopeaks, a typical CLLBC offers an ER better than 3.5% at 662 keV. This property is expressed as follow:

$$Energy\ Resolution = (\frac{FWHM}{Energy\ of\ Photopeak}) \times 100\% \tag{4.1}$$

In the formula, FWHM stands for Full-Width-at-Half-Maximum and this definition is illustrated in Figure 4.2. It could be seen that FWHM has been defined as the width of the photopeak at

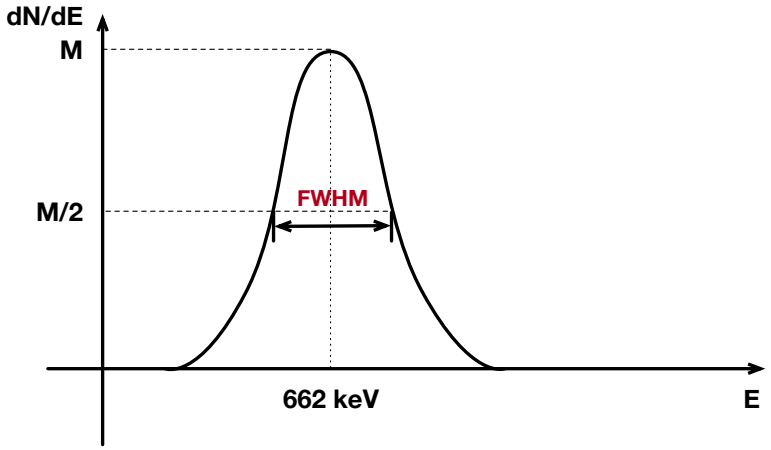


Figure 4.2: Definition of FWHM

its half maximum just as suggested by the name. A typical energy value of 662 keV is used because this is the prominent gamma energy from the isotope of $^{137}\mathrm{Cs}$. Because it is the only gamma line in the spectrum and therefore has been used as the standard for calibration. This shows that in terms of this property, a smaller index indicates a better performance. Graphically, the photopeaks in the spectra will look "sharper" and "slimmer". By comparison, CsI(Tl) detectors normally offer an ER of 6.6% at the same energy, which is almost twice as much as the ones of CLLBC.

Last but not least, the decay time of a CLLBC presents a parameter smaller than 270 ns. A commercial CLLBC gives 115 ns as its decay constant (Radiation Monitoring Devices Inc., 2023). The decay time refers to the time it takes for the scintillation light intensity to decrease to a certain fraction (usually half) of its maximum value after the scintillator has been excited

by radiation. This could be taken as the resolution in terms of time. Apparently, the smaller this parameter, the better the response time it will be for the whole detection system. A CsI(Tl) detector normally has a longer decay constant of 800 ns. CLLBC detectors apparently have a huge advantage over CsI(Tl) in this perspective.

However, CLLBC is not perfect. It mainly has two disadvantages and they are the hygroscopicity and its cost. Hygroscopic materials tend to absorb water from their environment, which deteriorates its optical and physical characteristics.

Overall, D5 RIID has an improved detection performance over D3S ID but with a smaller supported isotope library. Besides it has a more rugged design and longer battery life thanks to the dual power system design. What is more, D5 RIID has a much faster response time and stronger energy resolution due to a more advanced scintillator built in.

4.2 A problem we are facing

However, a reported case with these handheld RIIDs is that the battery life has not been sufficiently long for the application according to Kromek. D5 RIID addressed this issue by providing a dual power system with replaceable AA batteries. When the device runs low on the battery, it could be restored by a quick in-field battery replacement. During the replacement, the device will not be interrupted thanks to the internal lithium battery. In total, with the external batteries, D5 RIID could last as long as 24 hours. D3S ID, on the other hand, cannot function as long as D5 RIID because of the huge power drain by the smartphone. According to Kromek, the smartphone system could not last longer than 8-10 hours before recharge.

This is because the solution offered in D3S ID and D5 RIID is a software-based PCS implementation. As reviewed in the last chapter, PCS performs a generalised likelihood ratio hypothesis test between maximum likelihood estimators (MLE) for Poisson sources incorporating a model of background radiation based on the given detector and the 'dominant modes of variation'. The hypothesis test involves comparing two hypotheses: one positing that the spectrum results solely from the background (H0, the null hypothesis), and the other suggesting that a specific quantity of a given isotope or isotope is accountable (H1).

However, the computational cost for PCS is not insignificant. It has been pointed out in (Cosofret et al., 2013) and (Cosofret, Shokhirev, Mulhall, Payne, & Harris, 2014) that a test spectrum can be processed against a library of 16 isotopes in 20 ms on an Intel Core i7 2.80 GHz with 4 GB RAM. Given this information, it could be calculated that one spectrum process takes at least 5×10^7 clock cycles to finish. It is also reported that in implementation on a dual-core ARM Cortex A-9 using 'Xilinx evaluation hardware' spectra could be processed in 500 to 800 ms (Cosofret et al., 2014). Without additional technical information regarding the hardware and algorithmic operations, it becomes challenging to provide accurate estimates or

definitive statements about power consumption. However, if we consider the processing of 1-second spectra, this translates to a processor operating with a duty cycle ranging from 50% to 80%, likely consuming power in the range of 0.1 W to 1 W. Given current battery technologies, such a system could operate for a few weeks at most. While this surpasses the performance of current systems requiring daily charging, it falls approximately two orders of magnitude short of the nearly passive efficiency seen in threat detection systems like household smoke detectors, which typically boast battery lives spanning multiple years.

Given the fact that the handheld device mentioned above is running on a resource-constraint application case, a new solution or implementation for longer battery lives is evidently needed. This potential improvement will assure a more sustainable use and less interruption brought by frequent battery change or recharge.

4.2.1 Possible proposal of a solution

Just as reviewed in the last chapter, the idea of event-based processing might work synergetically well with the task. And this idea has already been experimented on a couple of publications (X. Huang, Jones, Zhang, Furber, et al., 2020), (X. Huang, Jones, Zhang, Xie, et al., 2020). In 2020, the idea of using event-based processing, specifically spiking neural networks for RIID was proposed at the conference of Event-Based Control, Circuit and Signal Processing (EBCCSP). The author pointed out the power consumption problem caused by the current frame-based processing and qualitatively analysed the feasibility and advantages of using SNN for this application. A rough design flow was established in the paper. It started with the data collection and data pre-processing to the SNN implementation and rough hardware idea using high abstract symbols.

In the same year, the author proposed a more detailed work at IEEE International Conference on Electronics, Circuits and Systems (X. Huang, Jones, Zhang, Xie, et al., 2020). In this piece of work, the author implemented an FPGA design of a simple 3-layer SNN to do RIID. This proposed SNN constitutes of 100 input neurons, 40 hidden neurons and 8 output neurons. The 8 output neurons correspond to 8 different classes of radioisotopes: $\{^{241}\mathrm{Am}, ^{133}\mathrm{Ba}, ^{57}\mathrm{Co}, ^{60}\mathrm{Co}, ^{137}\mathrm{Cs}, ^{152}\mathrm{Eu}, ^{226}\mathrm{Ra}, ^{232}\mathrm{Th}\}$. This implementation followed the design flow proposed previously and delivered the FPGA design with a really low power consumption of 72 mW. However, this approach has a quite limited library of isotopes supported and the dataset is relatively easy. The testing results showed that the accuracy for this network is slightly low with just 87.61%. Even though the conversion from ANN to SNN only cost around 1% of accuracy drop, this still proves that this simple architecture is not substantially capable of handling more complex dataset. What is more, the system requires 3 s for the results, given that the source event rate set at 500 Hz, this system calls for 1500 events for the inference. This is not really efficient to some extent.

The newest work that follows this approach was published in 2021. A more complex convolutional neural network architecture was converted into SNN and implemented on FPGA. This network was made up of a convolutional layer, a pooling layer and and a fully connection layer. The convolutional layer consists of 4 different kernels with the kernel size of 5 and stride of 1. Because the spectra only provide 1-dimensional vectors, the kernel is also just 1-dimensional. This was followed by a simple average pooling layer with the size of 16 and stride of 16, which reduced the dimensionality to 256 neurons. This was later followed by the full connection to the output layer of 8 neurons. The dataset still include 8 isotopes classes as mentioned above. A better performance was observed from this convolutional spiking neural network with the accuracy of 90.62%. However, it showed very high false rate on classes of two specific radioisotopes: $^{226}\mathrm{Ra}$, $^{232}\mathrm{Th}$. The system was often found accidentally inferring $^{226}\mathrm{Ra}$ as $^{133}\mathrm{Ba}$ due to the similarity of the spectrum between these two isotopes. The reason why it performs badly on $^{232}\mathrm{Th}$ is that the signal for this class showed strong stochasticity and did not give off a strong pattern.

Nevertheless, the feasibility of using SNN to deal with RIID has been proven in these two publications. Therefore, combination of event-based processing and RIID should be further explored so that more efficient solutions with higher accuracy could be implemented. A more detailed overview of event-based processing and SNN shall be discussed in the next chapter including the algorithms, tools and applications.

4.3 Datasets

To deliver new solutions to the problem, multiple different dataset were sourced thanks to Kromek. Here we shall briefly explain and label them for later reference. These datasets were used to train and evaluate our proposed algorithms and implementation.

4.3.1 Simple re-binned dataset

Collection

The data was collected from real radioisotopes with two different setups from different distances to the source. These two setups are differentiated by the presence of a Polymethyl Methacrylate (PMMA) phantom, which is supposed to mimic the torso movement of the observer. Spectra for each radioisotope were recorded every second for 120 seconds. This measurement applies to all distances setup: 10 cm, 25 cm, 50 cm, 1 m and 1.5 m.

Pre-processing

Raw data for each sample contains 4096 bins, simple re-binnning was applied to reduce the dimensionality to 100.

Dimensions

There are 9,600 samples in total with total classes of 8 different radioisotopes. Each sample contains 100 features. These radioisotopes include: $^{241}\mathrm{Am}$, $^{133}\mathrm{Ba}$, $^{57}\mathrm{Co}$, $^{60}\mathrm{Co}$, $^{137}\mathrm{Cs}$, $^{152}\mathrm{Eu}$, $^{226}\mathrm{Ra}$, $^{232}\mathrm{Th}$.

4.3.2 CLLBC synthetic dataset

Collection

The data was synthesised based on real isotope spectra collected from CLLBC scintillation detectors. The collection was made at varying distances: 10 cm, 20 cm, 25 cm, 30 cm, 45 cm, 50 cm, 60 cm, 1 m and 1.5 m. Same as the simple re-binned dataset, they were recorded every second for 2 minutes.

Pre-processing

Simple re-binning was applied first to reduce the dimension to 1024 channels. 105 basic spectra patterns were then processed with randomly generated gain shift and background noise.

Dimensions

In total, there are 19,005 samples covering 8 different radioisotope classes. 1024 channels exist for each synthetic sample. The covered radioisotopes include: $^{241}\mathrm{Am}$, $^{133}\mathrm{Ba}$, $^{57}\mathrm{Co}$, $^{60}\mathrm{Co}$, $^{137}\mathrm{Cs}$, $^{152}\mathrm{Eu}$, $^{226}\mathrm{Ra}$, $^{232}\mathrm{Th}$.

4.3.3 Enhanced CLLBC synthetic dataset

Collection

Just like the last dataset, this dataset was also artificially generated based on real spectra collected from CLLBC scintillation detectors. A minimum of 10 spectra were collected for each source prescribed from American National Standards Institute (ANSI) N42.34-2015. Following the same collection distances and timing setup as CLLBC synthetic dataset, this dataset was created with an enhanced library

Pre-processing

This dataset used the same pre-processing methodology as CLLBC dataset for re-binning, gain shift adding and background noise generation.

Dimensions

In total, there are 9,000 samples covering 18 different radioisotopes. 1024 features are expected from each sample. The supported radioisotope list include:

^{241}Am, ^{133}Ba, ^{57}Co, ^{60}Co, ^{137}Cs, ^{152}Eu, ^{226}Ra, ^{232}Th, ^{67}Ga, ^{131}I, ^{192}Ir, ^{99m}Tc, ^{201}Tl, ^{237}Np, depleted uranium (DU), highly enriched uranium (HEU), weapons-grade plutonium (WGPu), naturally-occurring radioactive material (NORM).

4.4 Conclusion

This chapter discussed the motivation and background for this PhD programme. A real application case has been presented where a handheld device made by a UK company called Kromek for radioisotope identification was introduced. Details about these products including supported radioisotopes and resolutions were given.

In the meantime, the problem with these products was brought forward too. They need battery changes frequently, which can interrupt the patrolling and efficiency of security control. The existing pure software-based solution has proven to be not the most power efficient.

Different aspects can be potentially improved to address this power shortage issue. Here we proposed a possible direction for the solution, which includes both software and hardware redesign. This might push the computation towards an event-based paradigm, a promising methodology in terms of power saving.

To set the objectives clearly, we nominated a few different datasets to train and evaluate our solutions. These datasets are all real or generated from real spectra. The simpler one would only measure the data ideally while for a more realistic application case gain shift and background noises were added into the dataset during synthesis.

Chapter 5

Event-based processing and spiking neural networks (SNN)

Event-based processing system distinguishes itself from the ordinary system by its specific way of encoding/decoding input signal and processing methodology. Spiking neural network could be considered as a type of event-based computing system given its intrinsic way of representing signals. This chapter shall briefly discuss the definition of event-based processing, how the signal could be processed and the most popular methodologies in this realm. Spiking neural networks, a representative event-based processing method, will also be explained in detail.

5.1 Background for the event-driven computing

It still remains a big challenge for human technology to catch up even close with the capability of animal's brain. Regardless of the most recent advancement in digital computer hardware, software, algorithms and system concepts, brains are still outperforming our computers or computing systems on a wide variety of tasks, especially when it comes to the resource-constraint applications. Caenorhabditis elegans, for example, is a type of microscopic roundworm that has only 302 neurons in the whole system. With a really limited number of neurons, they could still demonstrate complex functions and behaviours like locomotion, memories and learning. Another example is the honeybees, where they only need less than a milliwatt to achieve navigation and social intelligence, which is many orders of magnitude superior in both task competence and power efficiency to the artificial neuronal simulations and autonomous robots.

Although synchronous circuits have made its way in the market and modern semiconductor technology, they are definitely not the perfect answer in terms of resource utilisation. Constant computation and signal sampling even when there is no evident change in the signal is apparently not efficient, but that is how synchronous logic works. Efficient utilisation of

communication, computing capabilities and energy has become increasingly vital nowadays because of how the new computing systems are built. They are more networked, and spatially distributed. The need for a "frugal" system which offers more economic utilisation of resources has become inevitable (Miskowicz, 2018).

That is when event-based processing start to grab attention in the research and application. Such unique computation paradigm priorities resources and only responds to the operation that is imperative. Under this context, an *event* corresponds to a substantial change of the related state. A good example is the interrupt mechanism in a computer system where critical modules like timer, external input or serial communication raise interrupts to the central processing unit (CPU) which could be considered as an event. It prevents the CPU continuously checking these modules status but omits no critical information.

We have seen research progress in both control (Åarzén, 1999) (Heemels et al., 1999) and signal processing (Allier, Sicard, Fesquet, & Renaudin, 2003) (Aeschlimann, Allier, Fesquet, & Renaudin, 2004) in this specific research field. Inspiration for the research has been taken off from these great research works to catalyse the development of a systematic new event-triggered processing for the continuous-time systems.

To understand the event-based processing system, we should understand how the data has been encoded and how the typical event-based algorithms work. They shall be discussed in the later sections respectively.

5.2 Data acquisition and encoding

Analogue electronics are mainly used when dealing with the real world signals like temperature, humidity, audio etc. In a typical application case, they are captured by analogue sensors and later converted by analogue-digital converter (ADC) into discrete digital signals. This is because the digital systems are easier to design with the assistance of the electronics design automation (EDA) tools, meanwhile they offer greater immunity to noise and excellent signal precision.

Event-driven data acquisition and the following processing stage strongly demand the accurate time recording of the occurrence and processing. As the silicon technology progresses, voltage has been shrunk a lot to keep the power efficiency, which means amplitude-based processing will get even harder. The low voltage has limited the range a number could be represented. But they have secured a very high accuracy on the timing with the minimum time precision as low as hundreds of picoseconds ($10^{-12}\ s$). It is therefore expected that the performance of event-based systems could improve as technology scales.

But how is event data acquired and interpreted? Tsividis gave a tutorial on the mechanisms in his work in 2010. The flow was described in few different perspectives. There are various ways of sampling the continuous analogue signals, but the author focused on the level-crossing sampling and its corresponding data processing and hardware implementation.

5.2.1 Level-crossing sampling

Figure 5.1 gives a simple demonstration of the level-crossing sampling mechanism (Mark & Todd, 1981) (Sayiner, Sorensen, & Viswanathan, 1996) (Foster & Wang, 1991). A continuous

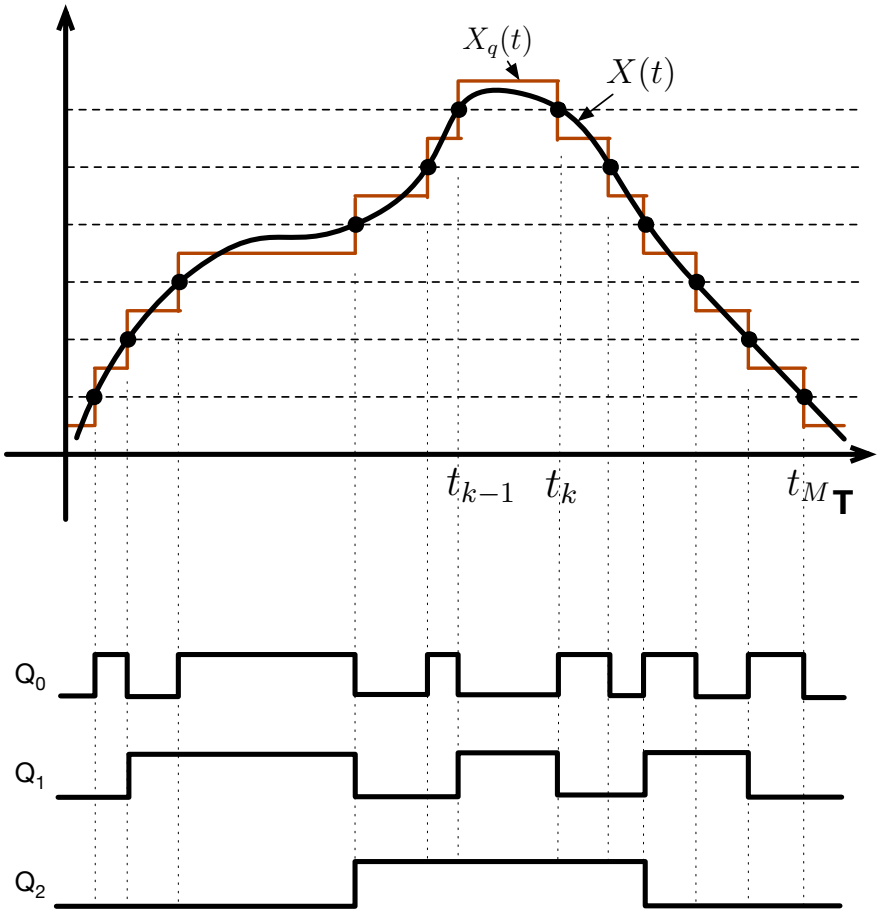


Figure 5.1: Level-crossing sampling and value encoding

signal $X(t)$ denoted in black is sampled every time when it crosses a certain pre-determined threshold. These thresholds illustrated as in horizontal dashed lines were evenly spaced amplitudes in this specific case (could be other types of distribution). The changing points to critically represent this signal has therefore been logged into a whole set of time-amplitude pairs

$$\{(t_k, X(t_k)) \mid k \in [0, M]\}$$

The sampled sample could be approximated with the quantised signal $X_q(t)$, which is painted in red. Following this scheme, it could be noticed that it does not require a clock to reach fairly good approximation. The sampling rate is entirely dependent on how fast this original analogue signal changes. This means when the signal increases or decreases faster, the sampling rate would be faster, and vice versa.

Accordingly, the sampling circuitry would only toggle when an event is triggered and cause power consumption. This would bring huge energy benefit in cases when the original signal only varies in a slow pace, since the crucial neighbouring events would be separated with a long time interval. This would mean that practically, using this scheme would save a significant power towards signals like audio when there is a long period of silence.

The quantised approximation is generated every time the original signal crosses a level. This approximate value is set between the two levels, normally half of sum of these two levels. Even though this approximation does not contain exactly the time-amplitude pair mentioned above, it contains essentially the same information as the dots.

Even though this sampling scheme is not susceptible to noise within the levels range. For example, if the speech audio signal is entangled with background or white noise during silence, a properly set threshold can leave the noisy wave under this value and therefore present a clean low volume. But in general cases, the input signal needs to go through a low-pass filter before sampling. This is to smooth out the unwanted local variations so that there will not be invalid noise around the threshold, which could cause unnecessary fluctuation or unwanted events.

5.2.2 Suggested circuitry

Continuous-time multi-bits representation

The level-crossing sampling shown in Figure 5.1 could be achieved with the hardware depicted in Figure 5.2 (Y. Li, Shepard, & Tsividis, 2005) (Akopyan, Manohar, & Apsel, 2006).

It could be seen that the circuitry is made of compactors, a simple serial connection of resistors and some encoding logic. This hardware could achieve the level-crossing and encoding function displayed in Figure 5.1. The input signal is represented as V_{in} and the outputs are multi-bits discrete encoding like $\{Q_2, Q_1, Q_0\}$. V_{Ref} defines the highest value of the range for a possibly allowed input. This maximum voltage is broken down linearly by the resistors with the same value. Therefore, linearly increasing voltage levels are created at each checkpoint as the reference voltage.

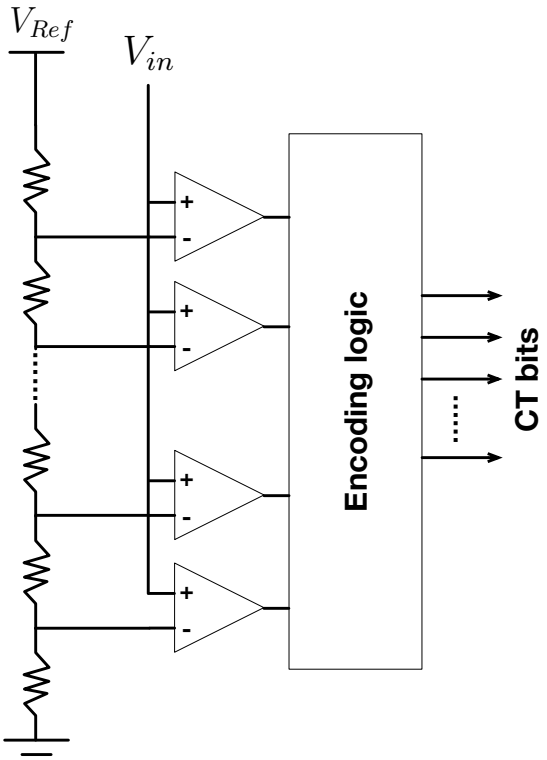


Figure 5.2: A multi-bit event-driven ADC

For example, to achieve the same functionality as in Figure 5.1 with 3-bits representation, we need 8 same resistors connected in serial. This defines the voltage level as in 8 different stages, i.e. from 000(0) to 111(7), or 7 different threshold from $\frac{1}{8}V_{Ref}$ to $\frac{7}{8}V_{Ref}$ according to Ohm's law. The comparators connected to these pre-determined thresholds will produce either logic high (1) or low (0) given the comparison results between V_{in} and these thresholds.

Take some actual numbers into this circuit, given that $V_{Ref} = 4V$, there will be thresholds listed as in $\{0.5,\ 1.0,\ 1.5,\ 2.0,\ 2.5,\ 3.0,\ 3.5\}$ connected to the negative terminal of the comparators. When the input voltage $V_{in} = 2.7V$, the corresponding output of these comparators V_{com} will be $\{1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0\}$. The eight different outputs from comparators and their 3-bit encoding Q_{CT} could be found as in: The implementation of the encoding logic simply has to follow the

V_{com}	Q_{CT}	V_{com}	Q_{CT}
000_0000	000	000_0001	001
000_0011	010	000_0111	011
000_1111	100	001_1111	101
011_1111	110	111_1111	111

Table 5.1: Encoding truth table of V_{com} and Q_{CT}

optimisation of the truth table and represent each bit with combinational logic.

Tri-bit representation

Since the number of bits N needed for representation of the true value should be the base-2 logarithm of the original number of levels, i.e. log_2^N, when a much higher magnitude of levels' representation is needed, the bits are also getting much more. To give a more efficient representation, a tri-bit representation shows a feasible solution. Because the input waveform shows strong continuity in the value, which means there is no sudden value drop like in digital signals, a tri-state up-and-down signal is all we need for this scheme.

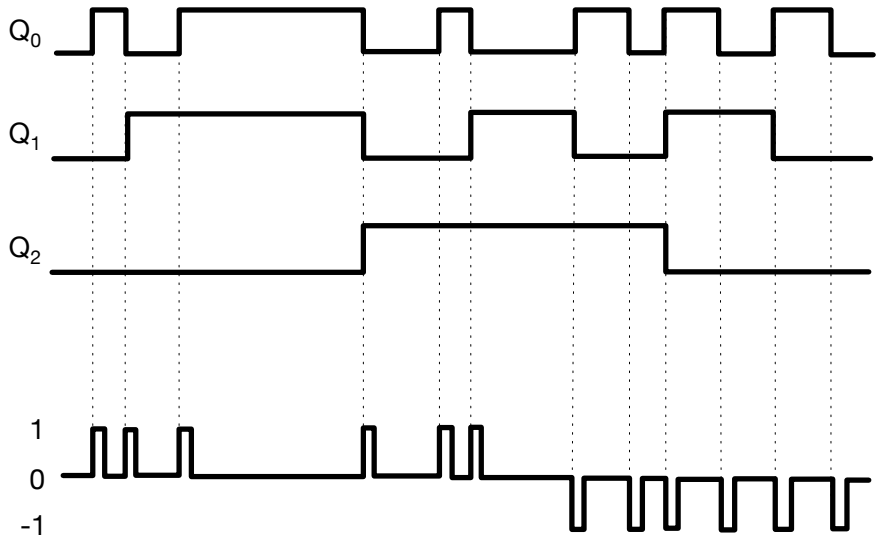


Figure 5.3: A tri-bit representation

Take the same input analogue input signal from above, it could be seen that the same multi-bit representation could be compressed to a single signal with three states $\{1, 0, -1\}$. It is obvious that when the input signal crosses the threshold upward, signal 1 is given and vice versa. Signal 0 suggests there is no change during this period. Each signal flip is supposed to have the same width and present the same value change in terms of the quantised approximation.

The hardware for this type of scheme can be found in Figure 5.4. A simple feedback loop is formed with the signal travelling from input voltage V_{in} through the comparators to the control logic which feeds back a digital signal to provide two threshold voltages V_{upp} and V_{low}. The input signal is compared against two values to check if any of the following conditions is met:

- $V_{in} > V_{upp}$
- $V_{in} < V_{low}$

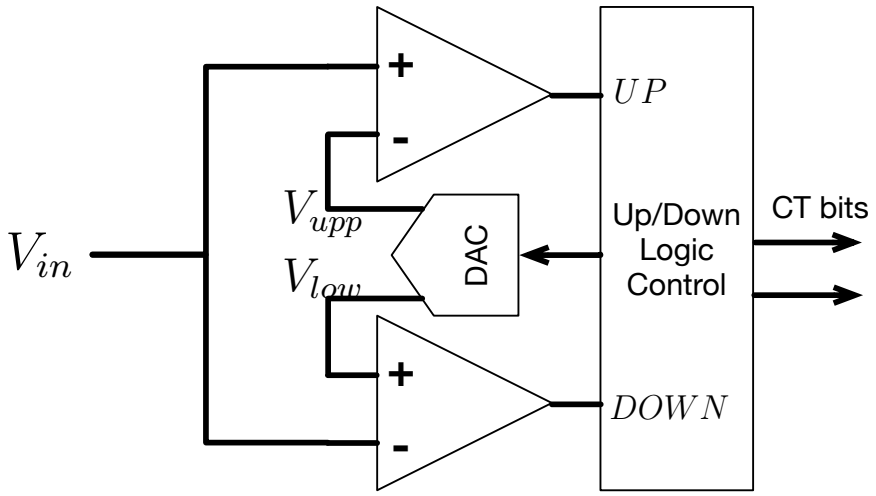


Figure 5.4: Asynchronous ADC

These two conditions signify the level crossing through the upper bound or lower limit respectively. The crossing event is then translated into signal UP or $DOWN$ to indicate the crossing, which is then encoded into continuous time bits. Since the crossing has already happened, the upper and lower bounds need to be updated to enclose the input, this is achieved by the logic control triggered by $UP/DOWN$ signal. This is then translated by the digital-to-analogue converter (DAC) to feedback to the input comparators.

5.2.3 Other sampling techniques

Besides what is mentioned above, there are also other sampling techniques.

Difference integral sampling

One of the representative sampling techniques is an integral-based sampling which detects changes of the signal (Ciscato & Martiani, 1967) (Miskowicz, 2007). This scheme uses the integration value as the sampling criterion to decide if the system should sample or not. To assume the last sampling time as t_k, the way to decide the next sampling time t_{k+1} is to check the following integration:

$$\int_{t_k}^{t_{k+1}} |x(t) - x(t_k)| dt = A \tag{5.1}$$

Where $x(t)$ represents the real value at time t, correspondingly, $x(t_k)$ is the actual constant value at time t_k; A is the integration value. Once value A is bigger than a certain threshold, a new sample will be made and its corresponding time step will be logged down and updated as the next integration criterion baseline. Just as depicted in Figure 5.5, the integral sampling calculates the absolute area under the curve to decide when for the next sampling point. When the amber area reaches the threshold, a new sampling point will be added to the system which is treated as an event just as indicated below the curve. Afterwards, the next integration

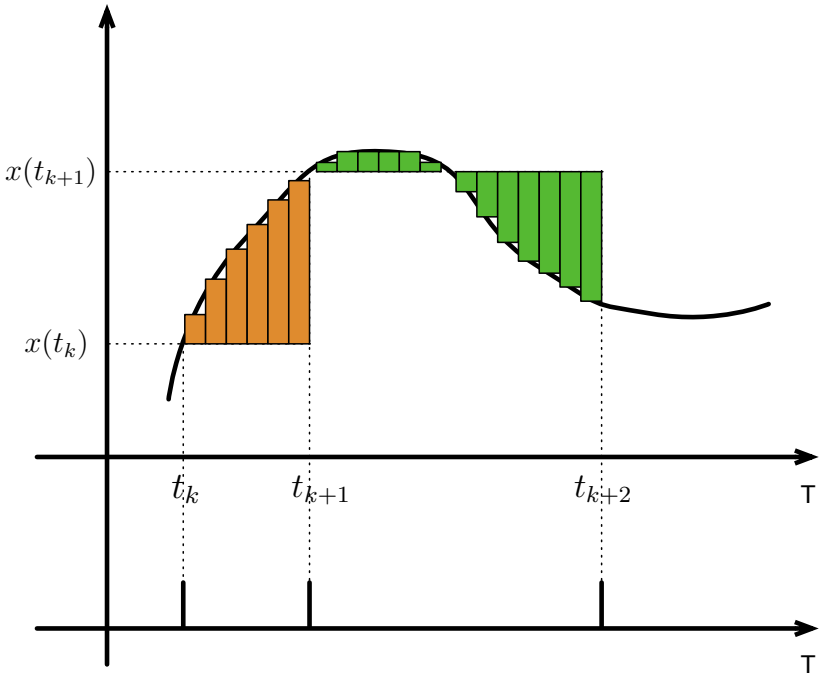


Figure 5.5: Integral-based event-driven sampling

will take the new baseline $x(t_{k+1})$ for the absolute area that is enclosed with this line and the curve. This demonstrates graphically well how the sampling works when the curve is declining or fluctuating around the baseline. Once the new green area reaches the threshold, the event will be logged and the baseline will be updated again at time step t_{k+2}.

As for low-variance signal, this scheme of sampling will result in a sparsely distributed discrete spike-cluster. Therefore, it was also suggested to add a vastly periodic sampling scheme in addition to this integral-based sampling scheme. This is to make sure there will be a minimum of sampling rate to prevent aliasing, which means the signal reconstructed from the sampled data strongly disagrees from the the original signal.

Error-driven sampling

In a control system, one can build an estimator that has variable sampling frequency. This is achieved by building a sampler whose sampling rate is a function of the error, i.e. the difference between the real signal and the prediction.

In an early publication, Dorf, Farren, and Phillips proposed a method where they implemented a close-loop control system to approximate the input signal (Dorf et al., 1962). This system is illustrated as in Figure 5.6.

Input is denoted as R and the error E is calculated by the difference of the input and the approximation or prediction C. The exact time for the next sample is calculated in the dotted box on the side of this loop where the input is error E and output is the next sampling time T.

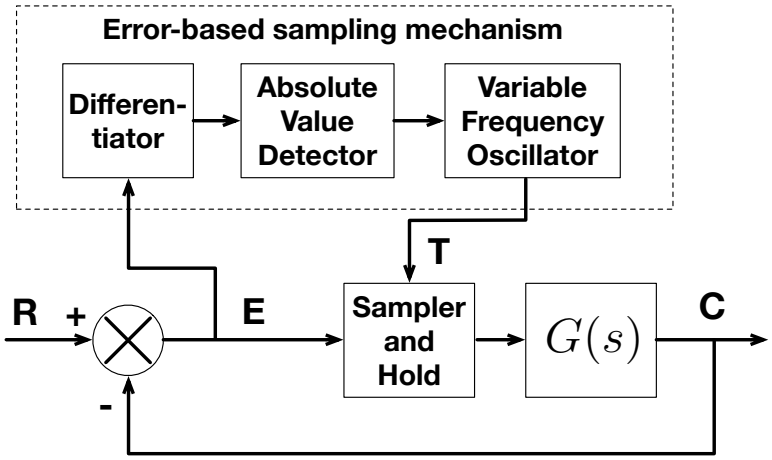


Figure 5.6: Error based sampled-data control system data flow

In a more recent publication, the author used a linear predictor instead of the flow mentioned above (Suh, 2007). The overall algorithm flow is described in Algorithm 1.

Algorithm 1 Adaptive sampling with a linear predictor

1: **Init:** $\hat{x}_0 = \ldots = \hat{x}_{-M+1} = 0$
2: **repeat**
3: **Op 1:** $\hat{x}_k = f(\hat{x}_{k-1}, \ldots, \hat{x}_{k-M-1})$
4: **until** $|x_k - \hat{x}_k| > \delta$
5: **Op 2:** Transmit $x_k, \ldots, x_{k-M}$
6: **Op 3:** $\hat{x}_k = x_k, \ldots, \hat{x}_{k-M} = x_{k-M}$
7: **Go to Op 1**

It could be seen that not all collected data from the sensor will be transmitted to the system. A selective transmission is achieved by checking the difference between the prediction based on the M data points in the memory and the newest actual value. If the difference is bigger than a threshold δ, a new group of data will be transmitted, this will also update the data points in the memory.

Overall, as demonstrated above, the event-driven data acquisition can effectively drop the sampling rate yet remain a relatively good approximation with the right signal reconstruction. This shows that all the necessary information from the input signal could be well retained with a more appropriate sampling time. However, this advantage may be wasted if there is no matching event-based signal processing tool or algorithm.

5.3 Address-event representation and communication

5.3.1 Representation

We have already discussed different ways of event-based encoding like level-crossing and other techniques. Imagine there are different channels from the input with each functioning the same for event-based data acquisition. How could this clusters of information be effectively transmitted so that the later signal processing could tell where the source of the event is coming from? An Address-Event Representation (AER) was introduced back in 1991 by Mead's lab (Lazzaro, Wawrzynek, Mahowald, Sivilotti, & Gillespie, 1993) to offer an effective way of encoding and decoding.

As suggested by the name, it carries the information of the source address as well as the event. In Figure 5.7, a simple AER encoding is demonstrated. It could be noticed that there

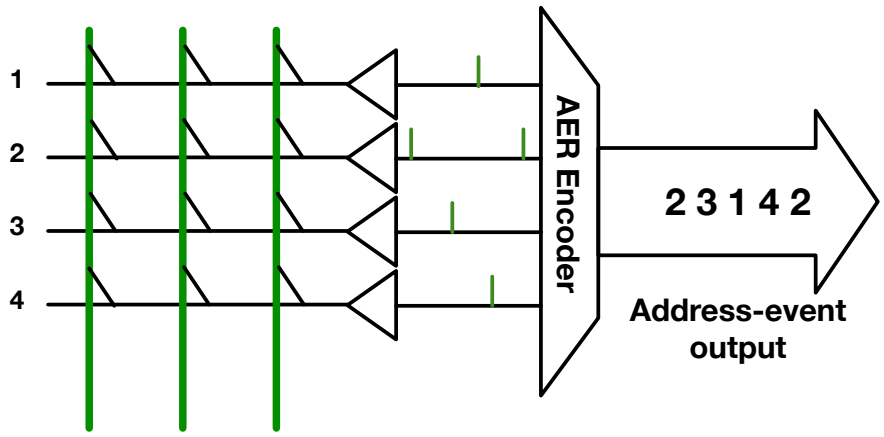


Figure 5.7: AER encoding

are four different lines with numbers labelled from 1 to 4, with each representing a source. The green line conveys an upper level logic that delivers these events. They could selectively drop event on a specific line. All the encoder has to do is to encode these discrete events at each address line into packets of events. Since events are defined to also carry temporal information, these AER should be encoded strictly following the time the event happens.

It could be seen that AER offers effective representation of events with the size of $log(N)$ bits when N sources are present. Time multiplexing is used instead of using lots of wires that are mostly not asserted if the events at the sources are sparse. This would result in concerns temporally since there are cases when two events happen simultaneously. However, modern electronics are capable of running at the frequency of at least KHz, which could theoretically support a fan-in as big as a thousand or millions if runs at MHz yet still allowing each input fires at around 10 Hz.

5.3.2 Arbitration mechanisms

However, this time multiplexing scheme still faces the need for an effective arbitration. It means there needs to be an order to sort out for the source to access the shared output. Cases when two events raise the request simultaneously have to be considered because these requests are raised asynchronously, which means they can arise basically at any time. The shared output in AER is the output bus, there will be information loss or distortion if two AER packets are loaded at the same time.

Typical arbitration mechanisms will be briefly discussed here, one should choose the optimal mechanism under a specific application case.

Carrier sense

This is the most simple and straightforward way of implementing the arbitration mechanism. The arbiter simply detects the usage of the bus before deciding if the arising event should be loaded or discarded (Abusland, Lande, & Hovin, 1996). Apparently, this will lead to data loss when the frequent input events are triggered. But it is able to deal with the scenario when the input is more sparse.

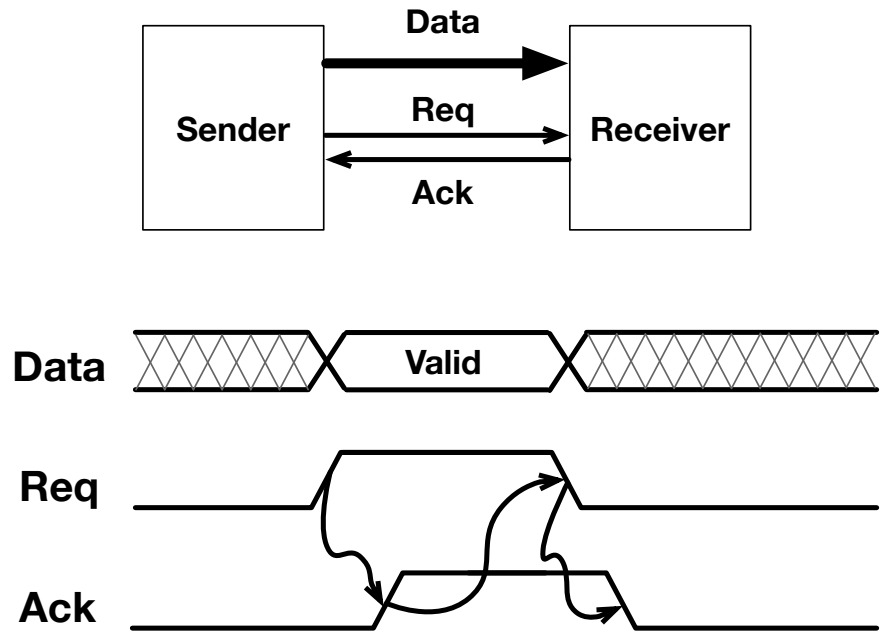


Figure 5.8: Asynchronous handshake

Tree arbitration

Generally in an asynchronous communication, a handshake protocol is followed. That is between two entities with one as sender and the other as the receiver, during which a request (Req) and an acknowledge (Ack) signal will be implemented between these two for the handshake and data transfer.

The flow can be seen in Figure 5.8, roughly 4 steps are involved in the communication:

1. Data is ready, sender raise the Req signal and hold
2. Receiver acknowledge the valid data by raising Ack
3. Handshake finishes, the Req signal is deasserted
4. Receiver deassert Ack signal after Req is deasserted

With this basic structure established, a tree of request handlers can be built in a hierarchy as illustrated in Figure 5.9. The access to the output AER channel is visualised as a red dot,

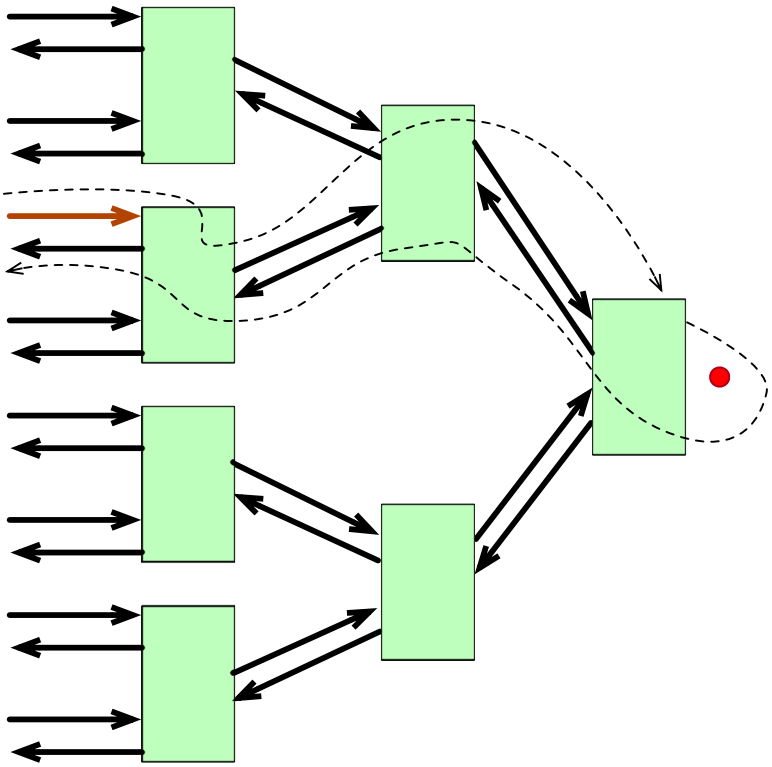


Figure 5.9: A basic tree arbitration scheme

which cannot disappear from this graph, but can freely move around the tree following the arrows. 8 individual sources are fed to the request handlers at the lowest hierarchy. Going up the tree, there are higher level handlers that only deal would grant access to its sub-handler. When an event is triggered, a request will be made towards the lowest handler, which is indicated as a brown arrow here. This request will be passed over to its higher-level handler

and then passed over to the highest handler on the right, which holds a request token. After this, the token will travel down the tree to the lowest level of the hierarchy to grant the access to the output AER channel. Before starting another request, this token needs to be restored back to the crown of the tree.

This has allowed the arbiter to make sure all the AER events will be process and all the requests answered. But the absolute timing of an AER event output may not accurately reflect the occurrence at the source.

Another more efficient variant called greedy arbiter works similarly as the basic tree arbitration scheme (Boahen, 2000). In the Figure 5.10, the same hierarchy is displayed. The essential

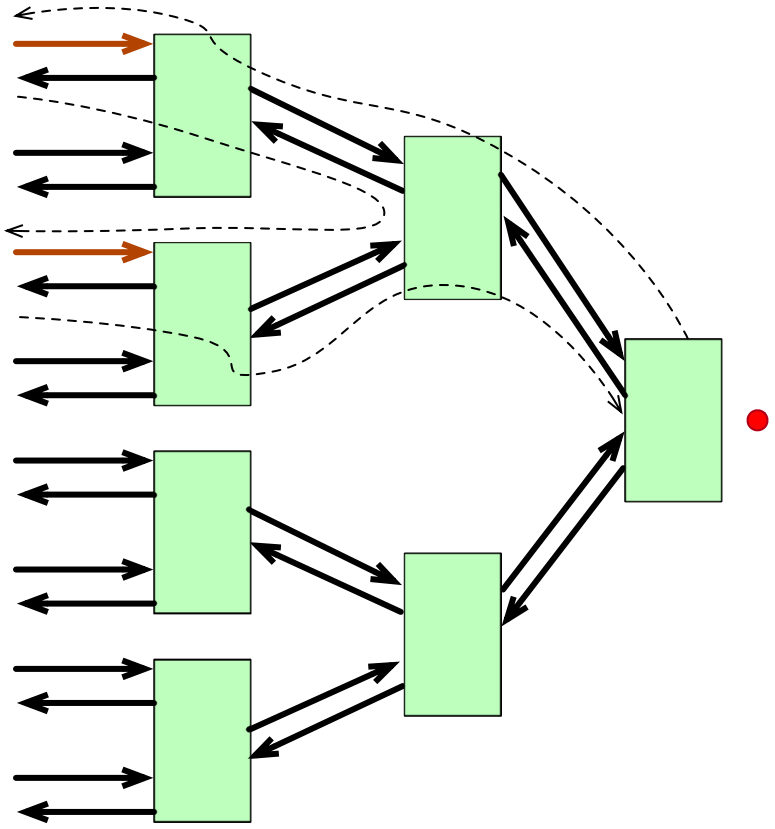


Figure 5.10: A greedy tree arbitration scheme

difference lies in how the token is directed after one transaction is finished. This change has allowed better handling with more concurrent events.

When two events submitted their requests, the token will travel down to one of the source to grant access. After this transaction is finished, the token travels back to the lowest hierarchical handler these two sources share instead of the top hierarchy. Then the access will be granted to the other request as the token travels down again. The token will only return back to the top handler after both requests are handled.

This modification enables the mechanism to effectively manage concurrent events. Enhanced performance is achieved in scenarios with increased concurrency, while in cases of infrequent events or low concurrency, the two mechanisms will exhibit comparable performance.

Ring arbitration

Another way of constructing arbitration is to build the handler in a circular ring (Imam & Manohar, 2011) as in Figure 5.11. The token only travels on request in one direction, clockwise in

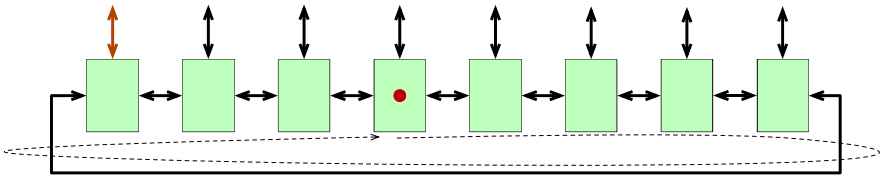


Figure 5.11: Ring arbitration scheme

this case, but remains fixed in a position otherwise. If one request is made, its corresponding handler will grant the access if the token is present. Otherwise, it will send request to its neighbours until the request reaches the handler that holds the token. For example, the left most request raised pushes the handler to send request to its left neighbour which ripples to where the token is. The token will then grant access to the left most request and then stay fixed until another request is made.

The advantages of this approach is less handlers use, it only needs N handlers compared to the tree arbitration. It will also have a good performance if the token travel distance is small.

5.4 Event-driven neuromorphic computing

It has been well established in this chapter about the event-based data acquisition, representation and communication. As for the event-based signal processing, there are a variety of applications and algorithms published recently. For example, in 2015, an event-driven integrated fault detection, isolation and control was designed for a linear system (Davoodi et al., 2015). Another example in the same year showed the implementation of a stabilising controller for the linear system, which employed a state observer, a given reference could be approximated using a quadratic Lyapunov-like function (Meslem & Prieur, 2015).

Apart from the event-based control system, another popular application in this field is the neuromorphic system. The countless effort has been made in this field in the great hope that human technology can ever come close to the power efficiency of the actual biological brains and neuronal systems. This has been demonstrated by various research works: sensory sensors design and development, bio-plausible neural networks (spiking neural networks), locomotion control by neuromorphic systems.

5.4.1 Neuromorphic sensors

Researcher have been utilising bio-mimicry to create artificial retinas, cochleas and nasal neurons aiming to replicate the function in a way that an actual sensory organ would perform. A lot of them have been integrated into applications with outstanding performances and power savings. Examples and reviews in these aspects shall be given in the following subsection.

Image sensors

Traditional imaging sensors are designed to export information in frames, with each pixel carrying its individual information. By comparison, the human retina has two desirable characteristics that can not be easily obtained by CMOS imaging sensor. First, it has a high dynamic range in terms of light intensity over nine decades. In other words, it can operate from very dim illumination to highly bright sunlight. The other feature is the spatiotemporal contrast contributed by the cells in the inner and outer plexiform.

To imitate the working mechanism of a human eye, event-based image sensors are designed. One good example is the dynamic vision sensors (DVS), which was invented by Delbruck et al.. The single pixel in this sensor is designed as in Figure 5.12.

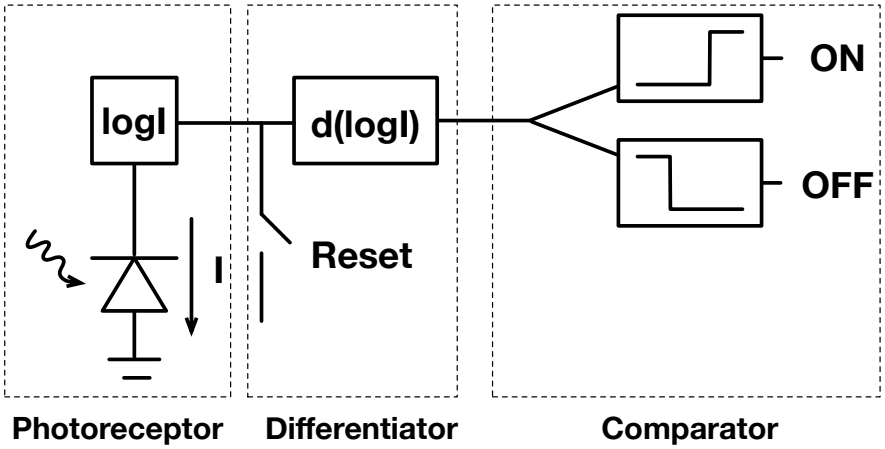


Figure 5.12: DVS pixel circuit design

Where the input light intensity, which could be linearly represented by the current I, is logarithmically compressed. This is followed by a differentiator, where the value changes against the time in the input is calculated. At last, the comparator does a similar AER encoding mentioned previously in tri-bit, where the ON and OFF signal indicates the level crossing up or down. They have been designed to asynchronously generate AER events so that a totally event-based processing system should be used to take full advantage of this sensor.

Applications around DVS have emerged at an increasing speed since the invention of this type of sensor. Since each DVS pixel is really sensitive to the brightness change, the most straightforward application is tracking. There are many tracking systems developed using DVS including object tracking (Gómez-Rodríguez, Miró-Amarante, Rivas, Jimenez, & Diaz-del Rio, 2011), satellite tracking (Cohen et al., 2019), camera pose estimation (Gallego, Forster, Mueggler, & Scaramuzza, 2015), car tracking (Litzenberger et al., 2006). These applications exploited the high dynamic range feature of the DVS and high response time given by the asynchronous architecture to present highly accurate tracking performance.

Apart from the tracking task, DVS has proven to be an effective component for classification (Schraml, Belbachir, & Brändle, 2010), 3D image construction (Kogler, Sulzbachner, Humenberger, & Eibensteiner, 2011), face detection (Barua, Miyatani, & Veeraraghavan, 2016), recognition (Ahn, Lee, Mullen, & Yen, 2011) and localisation (Weikersdorfer, Hoffmann, & Conradt, 2013).

Auditory sensors

Another great invention is the silicon cochleas, which is designed to imitate the biological organ in the human ear to capture audio by turning the vibration in the air into neural signals. Inside the cochleas, there is the crucial tissue called basilar membrane (BM), which turns the vibration into electrical signals that can be picked up by the brain. Topologically, BM is a thin tapered film that is situated inside the inner ear's scala media. This structure enables BM to sense a wide range of frequencies, where the stiffer base is more associated with high-pitched sound while the flexible apex resonates more with lower frequency.

A silicon cochlea is designed normally in a way to break down different bands of frequency of the input audio and give frequency analysis and signal processing. This can be normally implemented as in a cascaded set of second-order-section (SOS) bandpass filters (Lyon & Mead, 1988).

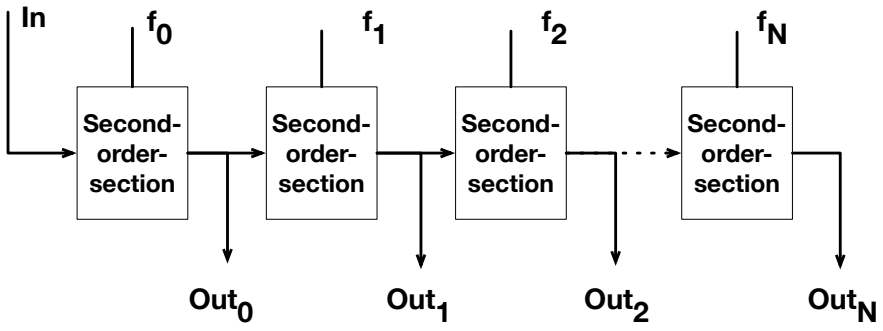


Figure 5.13: 1D cochlea structure

As depicted in Figure 5.13, input travels through a cascade of SOSs with identical designs except for the characteristic frequency. With the output from the last SOS driving the next, input audio can be dissected into different frequency bands with each resonating to a specific frequency and encoded into values.

Event-based cochleas like AEREAR2 work similarly except the output of the signal will be encoded into spikes using pulse-frequency modulators (Liu, van Schaik, Minch, & Delbruck, 2010). The pulse-frequency modulators essentially function like an integrate-and-fire (IF) model with pre-set thresholds. As for the neuron models, they will be explained in later sections.

With the silicon cochleas, event-based auditory systems are developed for various applications. The most prominent application case is localisation. It was both proposed in (van Schaik, Chan, & Jin, 2009) and (van Schaik, 2010) that the localisation can be achieved with just two microphones, a pair of silicon cochleas and a neuromorphic system. In (Finger & Liu, 2011), the author employed a binaural sound localisation system using a spike-based algorithm. This system uses the spikes from the 64-channel event-based cochlea AEREAR2 to evaluate the spike-based correlation, which indicates the interaural time differences (ITD) between the time of arrival of the sound to two different microphones.

Doppler effect explains that the shift in terms of the frequency of the sound is related to the relative velocity between the sound maker and the observer. This phenomenon could be used to navigate and locate, in (Figliolia, Murray, & Andreou, 2015), a system for micro-Doppler sonar is presented, a silicon cochlea with acoustic fovea and AER was utilised to deliver the function.

Speech recognition is another useful application field that could benefit from event-driven computing. In 2002, a neuromorphic algorithm towards speech perception was proposed where a fully connected feed-forward spiking neural network with synaptic plasticity mainly processes and visualises the speech patterns. Another example (Abdollahi & Liu, 2011) used AEREAR2 successfully to deliver a high classification rate on TIDIGITS dataset. The author used time-binned spikes as spike image maps as the input, which is later classified by supporting vector machine (SVM), and achieved speaker-independent digit recognition. In other words, this system could make the same inference despite the variations in the pronunciation of the same word. In 2017, Miró-Amarante et al. proposed an architecture of spiking neural networks that constitutes three different integrate-and-fire neurons to deliver vowel recognition tasks. This architecture is also implemented on an FPGA to recognise some Spanish words. In a more recent publication in 2019, the combination of a binaural event-driven silicon cochlea and a probabilistic model was presented. This piece of work was evaluated against TIDIGTS dataset, which showed that this pipeline is delivering a high accuracy and low computation cost at keyword spotting and speaker verification tasks.

5.4.2 Spiking neurons

In basically all the applications mentioned above, spiking neural networks were employed for signal processing. In this section, spiking neurons will be explained from different perspectives. That includes the definition, different types or models.

Neuron cells

As all commonly known, the neural system in vertebrate plays an important role for central signal processing and information transmission. Nuerons, as the basic element cell that constitutes the neural system in animals and human beings, are generally composed of axon, dendrites and soma, which is illustrated in Figure 5.14.

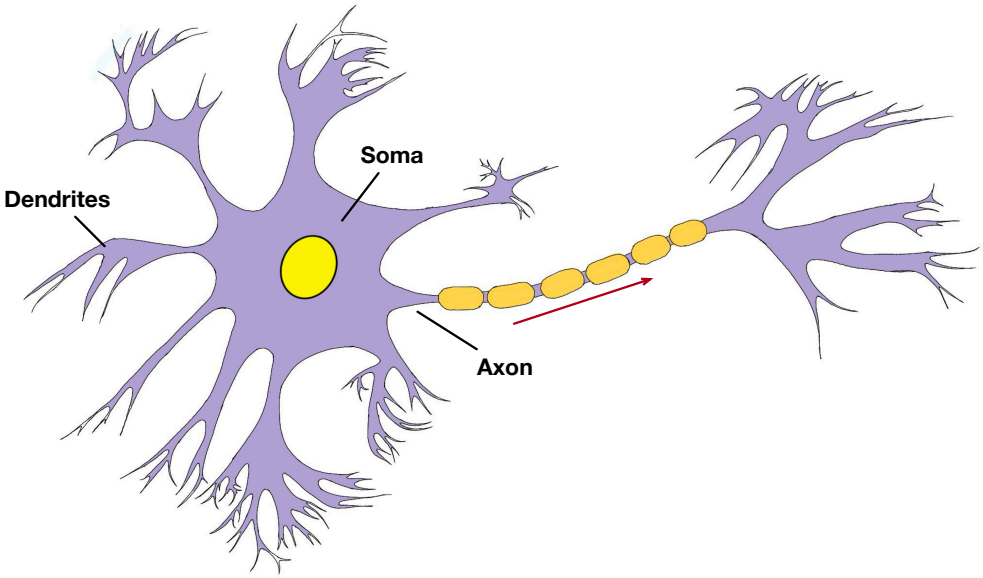


Figure 5.14: General illustration of a neuron cell

Each of the three essential structures has their own functionality. The dendrites are the tree branch looking structure positioned at one end of the neuron cell. They can be considered as the "input" for this neuron where information encoded as electrical pulses are fed. These signals will then by delivered to soma, where all the processing happens. Soma is the "central processing unit" of this neuron, the nonlinear behaviour happening in this part accumulates the signal from dendrites. Normally this accumulation stops when the accumulated value reaches a certain threshold. New signals, normally an electrical pulse, will then be generated by the soma and transmitted towards the other end of the neuron, i.e. axon. Equivalently, the axon in a neuron acts as an "output ports". The signals generated from the soma travels

through the axon, which is covered in a sheath to keep a good insulation from the outside interference, and eventually reaches the terminal of the neuron. Synapses, as depicted in

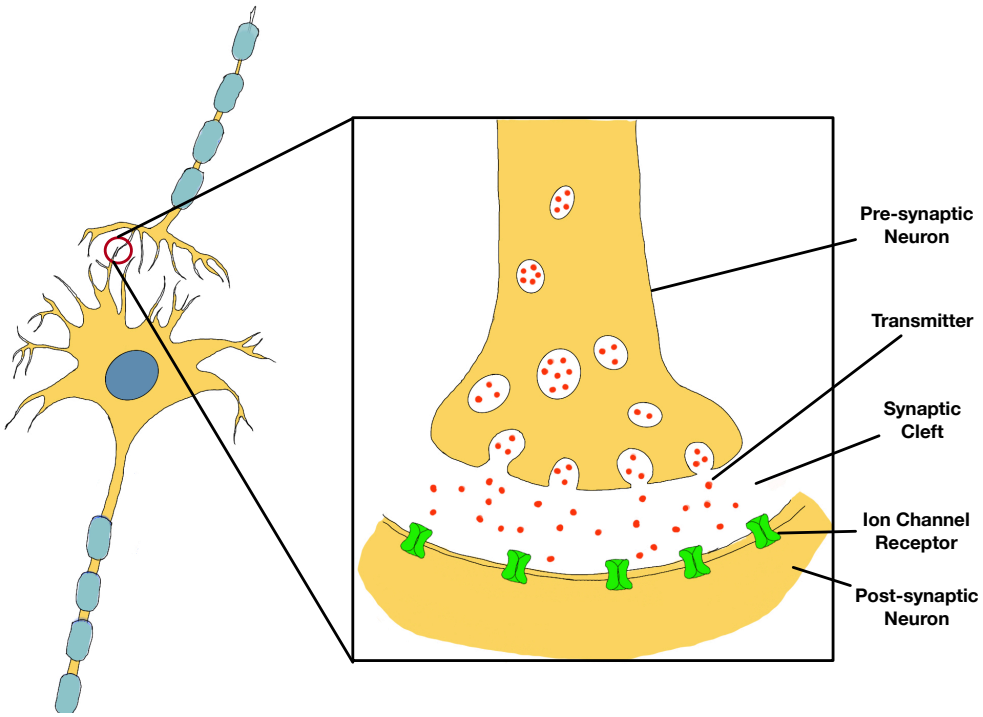


Figure 5.15: Synapses and transmitters

Figure 5.15, are the joint of two neurons or the neuron and the target cell where an electrical pulse is translated into a chemical signal, also known as neurotransmitters. Two neurons that are bound by a synapse could be called pre-synaptic neuron or post-synaptic neuron respectively. There is only a tiny gap called synaptic cleft between two neurons where it is filled with body fluid.

The signal transmitted to the terminal of the axon will trigger the release of the neurotransmitters. These transmitters then diffuse the synaptic cleft to reach the other neuron's dendrites. This is followed by the bounding process of the biochemicals and the receptors located on the surfaces of the neural membrane. Successful reception of these transmitters brings new electrical pulses that can be delivered across the next neuron.

Inspired by the structure and behaviour of the neurons, different levels of abstraction of this complex neuronal model exist. The modern deep learning application takes a simplistic construction of a neuron. It can be depicted in Figure 5.16.

It can be observed that the artificial neural model discards the electrical pulses in the mathematical construction and replaced them with simple numerical representation. The nonlinear behaviour is achieved by the activation function. Commonly used activation functions include sigmoid function, rectified linear unit (ReLU), hyperbolic tangent function (also known as tanh),

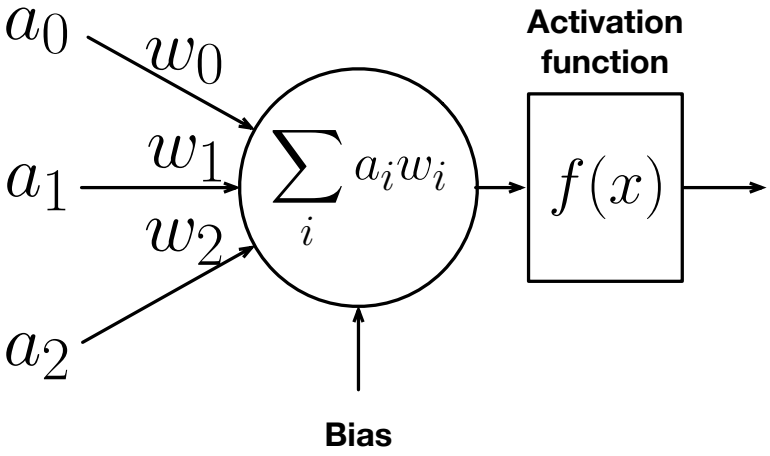


Figure 5.16: A simplified neuron model

exponential linear unit (ELU). Spiking neurons, on the other hand, retain the bioplausibility that

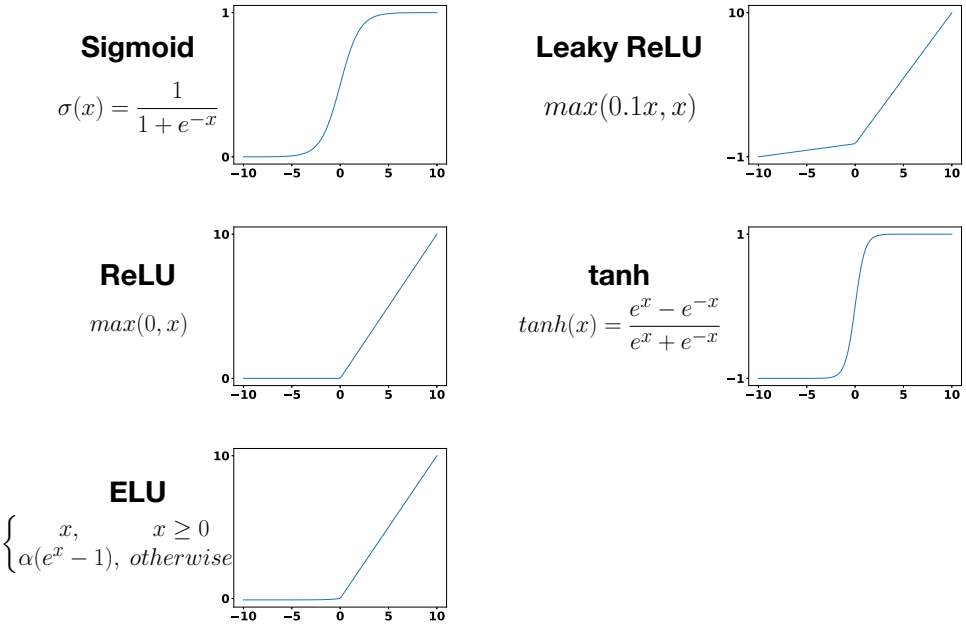


Figure 5.17: Commonly used activation function

gets lost in the oversimplification of neurons. Instead of encode the inputs as numbers, signals are generally encoded as spikes that approximate the actual shape of a pulse. However, depending on the applications, different spiking neuron models also exist for various requirements of the bio-plausibility. For example, if one wishes to simulate the actual behaviour in the brain for neuroscience research, a neuron model with higher level of bio-plausibility is desired

to deliver the realistic simulation. However, if the application is searching for an alternative neuron model for machine learning tasks, a higher level of simplification might be preferred due to the computational complexity and hardware overhead. Few commonly used spiking neuron models will be discussed in the following part.

Ion dynamics and reversal potential

Before explaining detailed neuron models, it is essential to learn how the ion dynamics and neuron potential works. Just like any other cells, neuron cells are enclosed with a membrane where interior is filled with liquid. The constitution of the liquid is mainly ions with different charges. Generally, the two big categories are sodium and potassium ions.

According to the thermodynamics, the possibility of a molecule taking a certain position with an energy E is proportional to the Boltzmann factor $e^{-(E/kT)}$, where T is the absolute temperature, k is the Boltzmann constant. On the other hand, the energy distribution for positively charged ions can be considered by the potential it is located at. i.e. $E = qU(x)$.

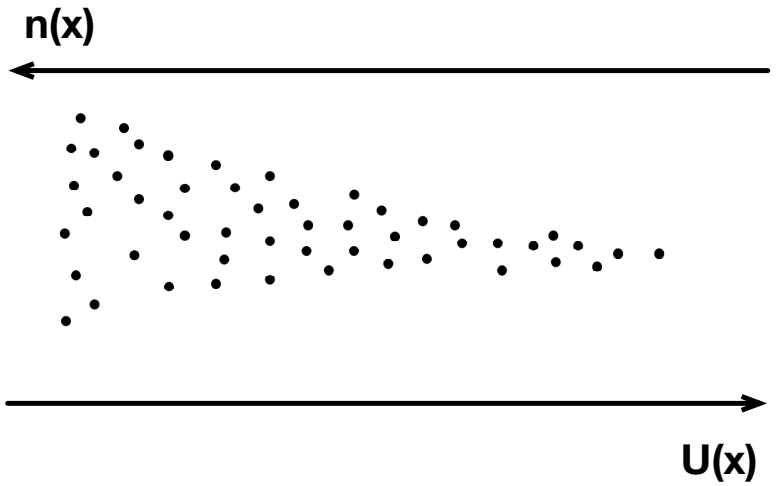


Figure 5.18: Density and potential energy

Imagine a distribution depicted in Figure 5.18, because of the positive charge of the ions, the energy distribution is proportional to the potential. Mathematically, this would make the possibility of finding an ion at a higher potential much less than the lower end. Therefore, the higher density of the ions lies in the opposite direction to the energy or potential.

When at an initial state, the ions will diffuse freely from higher higher-density area to the lower-density area. Due to the existing potential, ions are more encouraged towards the area with lower potential. When these two movements reach an equilibrium, the density of ions at a certain spot can be found as:

$$n(x) = exp[-\frac{qU(x)}{kT}] \tag{5.2}$$

The potential at a certain spot therefore can be formulated as:

$$U(x) = -\frac{kT}{q} ln[n(x)] \tag{5.3}$$

Given two different areas $x1$, $x2$ with different densities $n(x1)$ $n(x2)$, the potential difference can be given as:

$$\Delta U = U(x1) - U(x2) = \frac{kT}{q} ln \frac{n(x2)}{n(x1)} \tag{5.4}$$

While in an equilibrium at room temperature, the potential difference for sodium ions in a neuron is normally $+50\,mV$, similar metrics for potassium is $-77\,mV$. This means that sodium exists in a much higher concentration in the extracellular liquid than the inside, vice versa for the potassium ions.

In a neuron cell, there are ion channels and ion pumps in the membrane that connects the interior and exterior. Ion channels selectively allow a certain type of ions through the membrane, while ion pumps actively transport ions to a certain direction, usually against their electrochemical gradients. To maintain the neuron at the resting potential, normally at $-70\,mV$, ion pumps actively transport potassium in and sodium out of the membrane.

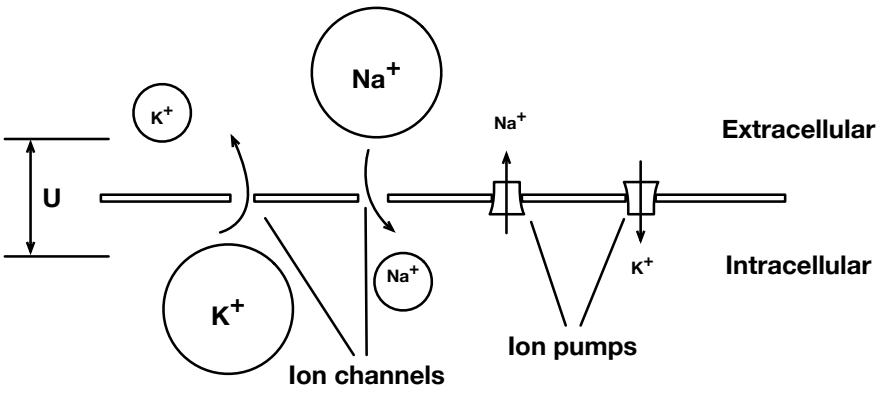


Figure 5.19: Neuron cell membrane model

Hodgkin-Huxley Model

From the dynamics developed above, the neuron's active potential is modelled as in Figure 5.20. Hodgkin and Huxley observed giant axon of the squid and found 3 different types of ion currents: sodium, potassium and leakage current. Status of ion pumps for both sodium and potassium depends on the action potential of the neuron.

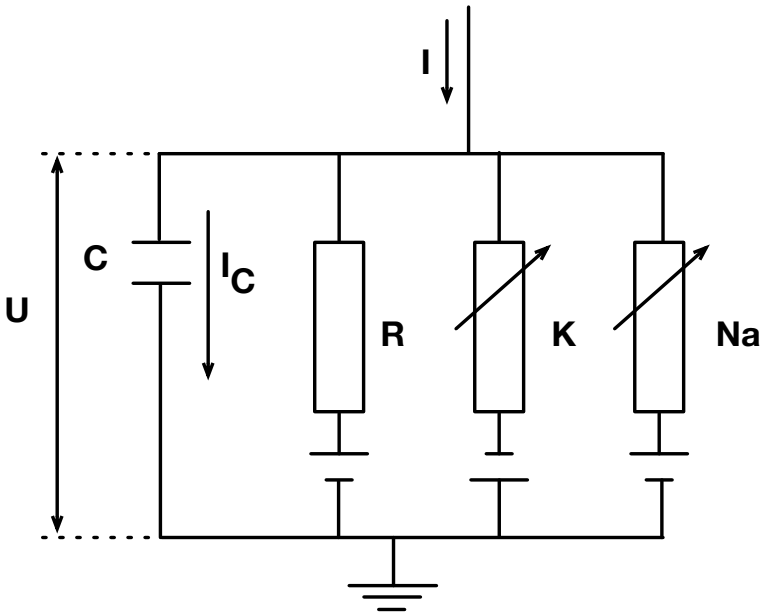


Figure 5.20: Equivalent circuit diagram of Hodgkin-Huxley model

When the neuron receives an injection of current I, it could flow into the capacitor and develop membrane potential U or flow through as leakage or reverse current derived from ion pumps. According to the rule of conservation of current, the current injection should equal to the sum of all current flow.

$$I = I_C + I_L + I_K + I_{Na} \tag{5.5}$$

The current flow through the capacitor can be derived as electric charge per unit time, which gives $I_C = C(dU/dt)$. Therefore, Equation 5.5 can be rewritten as follows:

$$C\frac{dU}{dt} = -(I_L + I_K + I_{Na}) + I \tag{5.6}$$

Leakage current is mainly the results of Cl^- leakage through ion channels and can be modelled with simple voltage-independent conductance g_L or resistance.

$$I_L = g_L\,(U - E_L) \tag{5.7}$$

The conductance for sodium and potassium channels are dependent on the voltage and they can be modelled as follows according to (Hodgkin & Huxley, 1952).

$$I_K = g_K\,n^4(U - E_K) \tag{5.8}$$

$$I_{Na} = g_{Na}\,m^3h(U - E_{Na}) \tag{5.9}$$

The conductance g_K and g_{Na} are the max conductance constant, while m, n and h are additional variables to model the voltage and time dependency. Details for these parameters are not included here for simplicity but can be found in (Hodgkin & Huxley, 1952).

Izhikevich Model

The model Hodgkin and Huxley developed is highly biologically accurate. But to simulate such models is computationally expensive. Therefore, parameters reduction was attempted to simplify the model and keep the essential neuron behaviours. Izhikevich model (Izhikevich, 2003) is one of the successful attempts that retains realistic neuron spiking behaviours and also cut the computational cost.

Instead of building a truthful neuron model with multiple dimensions, a reduced 2 dimensional neuron model was formulated. The author used mathematical and statistical approach to fit the neuron behaviour instead. All the behaviours can be modelled as in two variables described with differential equations:

$$\frac{dU}{dt} = 0.04U^2 + 5U + 140 - V_{re} + I \tag{5.10}$$

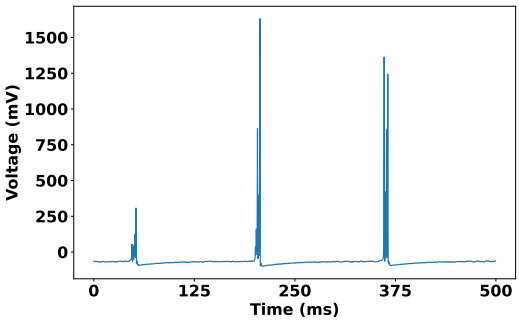
$$\frac{dV_{re}}{dt} = a(bU - V_{re}) \tag{5.11}$$

Where U is the membrane potential, while V_{re} represents the recovery variable mainly contributed by the activation of K^+ ionic current and inactivation of Na^+ ionic current. Just like the model described above, I is the injected current. This is complemented by the spiking condition:

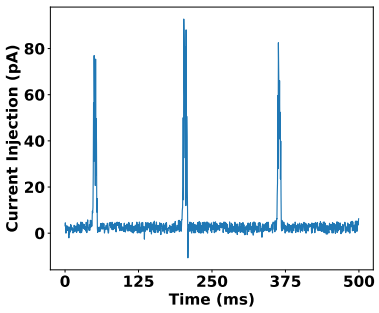
$$if\, U > 30mV, \begin{cases} U \leftarrow c \\ V_{re} \leftarrow V_{re} + d \end{cases} \tag{5.12}$$

This is explained by the fact that once the voltage crosses 30 mV, the membrane voltage shall be reset to c and the recovery variable is increased by d. Other coefficients a and b are the time scale of the recovery variable V_{re} and sensitivity of the recovery towards the subthreshold action potential v.

This model offers a simpler way to simulate neurons with complex spiking activities. For example, the spiking behaviours like regular spiking, intrinsically bursting and fast spiking observed from rat's motor cortex can be well simulated by tuning the four parameters, i.e. a, b, c, and d.



(a) One simple Izhikevich neuron's response to the current injection



(b) The corresponding current injection for this neuron

Figure 5.21: Izhikevich neuron simulation replicated from (Izhikevich, 2003)

In the original paper published by Izhikevich, a Matlab code was given for the neuron simulation. This is reproducible in Python[1] where 1002 neurons were simulated simultaneously to give an intrinsic spiking behaviour. This result is shown in Figure 5.21. A randomly picked neuron's membrane potential and its current injection intensity are illustrated in Figure 5.21a and Figure 5.21b respectively.

Leaky Integrate-and-Fire (LIF) model

As we have discussed above, the main functionality the neuron does is to accumulate the input and give electric pulses as part of a reset mechanism. In the meantime, there is leakage current present to establish and maintain the resting potential of the neuron. To sum up these two basic features, leaky integrate-and-fire neuron model was modelled.

As the name suggest, it has the leakage mechanism, but there is no complex voltage-dependent or time-dependent conductance that is described in Hodgkin-Huxley model. This electrical properties of the model can be understood with the assistance of the circuit diagram shown in Figure 5.22.

It can be seen that the same construction for leakage and the neuron membrane potential from Hodgkin-Huxley are retained. The Na^+ and K^+ channels' mechanism is replaced with a comparator, when the voltage reaches the threshold V_θ, a spike will then be generated at the current time step t_0, which is mathematically represented as a delta function. The same current injection I exists as an external input.

Accordingly, by the conservation of current, the equation to formulate such behaviour can be described as below:

$$I = I_L + I_C \tag{5.13}$$

1. This script can be found in the corresponding github repo.

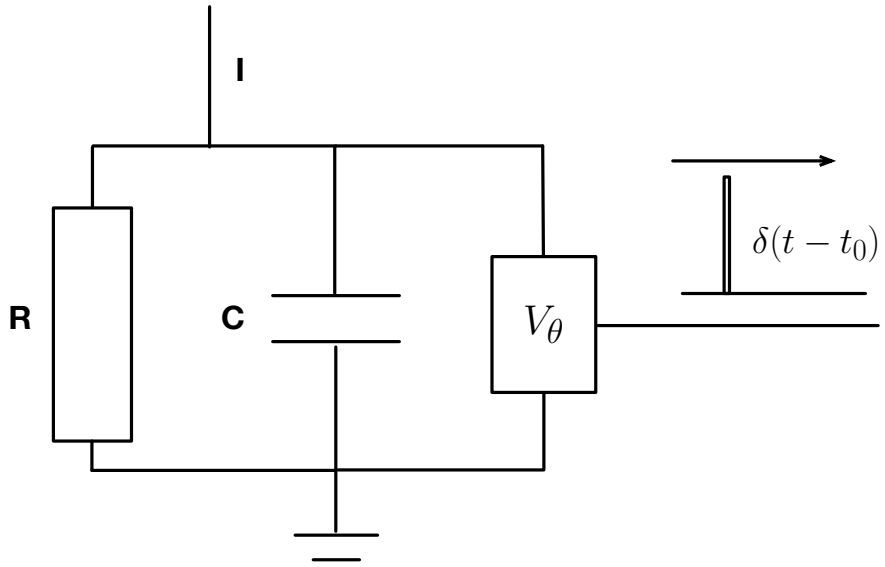


Figure 5.22: Leaky Integrate-and-Fire model equivalent circuit diagram

By expanding the equation 5.13, we can have the following:

$$I = g_L\, U + C\, \frac{dU}{dt} \tag{5.14}$$

Where g_L describes the conductance of the leakage resistance which equals $1/R$. Same as the notation used before, U here represents the membrane potential. Reorganising equation 5.14, we can have the membrane potential described in the ordinary differential equation:

$$C\, \frac{dU}{dt} = -g_L U + I \tag{5.15}$$

Or in another form of expression as:

$$\tau_m \frac{dU}{dt} = -U + IR \tag{5.16}$$

This is derived by multiplying both sides of equation 5.15 with the resister R. Where τ_m can be considered as the time constant of the neuron, which equals to RC. Similar to the Izhikevich model, the reset mechanism is a trigger-based mechanism. The membrane voltage is compared to a voltage threshold V_θ, when it crosses this threshold, a spike will be generated. In the meantime, the membrane potential will be reset back to a voltage level that is much lower than the threshold V_θ. Normally this value will be called reset voltage, which would sometimes be equal to the resting voltage or very close to this value.

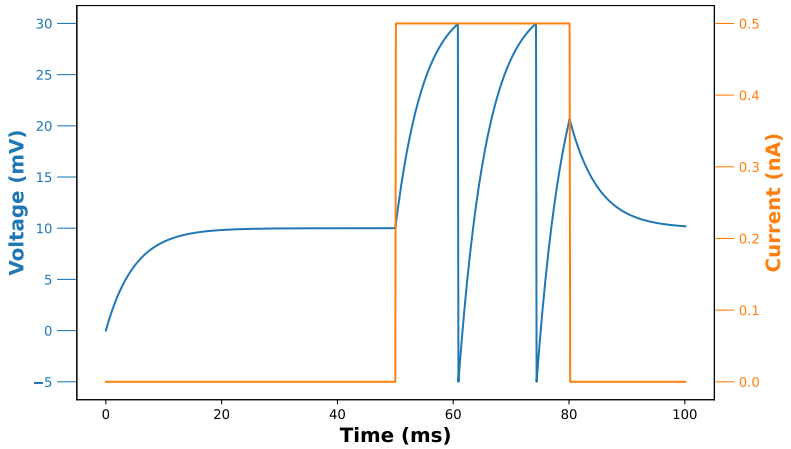


Figure 5.23: An exemplary LIF neuron membrane potential simulation

Take the simulation from Figure 5.23 for example, the neuron started with a membrane potential $0\ mV$, which is lower than the resting potential $30\ mV$. The leakage mechanism resulted in an exponential decay of voltage back to the resting voltage without the injection of the current. The potential starts to grow non-linearly under the injection of the voltage at $50\ ms$ and reaches the threshold $30\ mV$ and fires. When the current injection stops at $80\ ms$, the voltage increase halts following the current. Without the influence from the current again, the voltage drops back to the resting potential smoothly. Note that the resting voltage used in the simulation is not represented in Equation 5.16. It should be treated as a reversal potential similarly as in the Hodgkin-Huxley model.

A more simplified Integrate-and-Fire (IF) model

The LIF model mentioned above is already a substantial simplification of neuron models from Hodgkin-Huxley model. To push this step a bit further, a more simplified integrate-and-fire (IF) model was proposed. In this model, leakage was also discarded apart from integration and fire mechanism.

This functionality can be modelled as a simple accumulator with a potential monitor and reset. This is therefore formulated as:

$$C\frac{dU}{dt} = I \tag{5.17}$$

With both sides multiplied with the membrane resistance R, Equation 5.17 can be rewritten as:

$$\tau_m\frac{dU}{dt} = IR \tag{5.18}$$

This describes that the change of the membrane potential is only influenced by the injected current. τ_m and R are both constants in a defined neuron model that can be tuned for a specific behaviour.

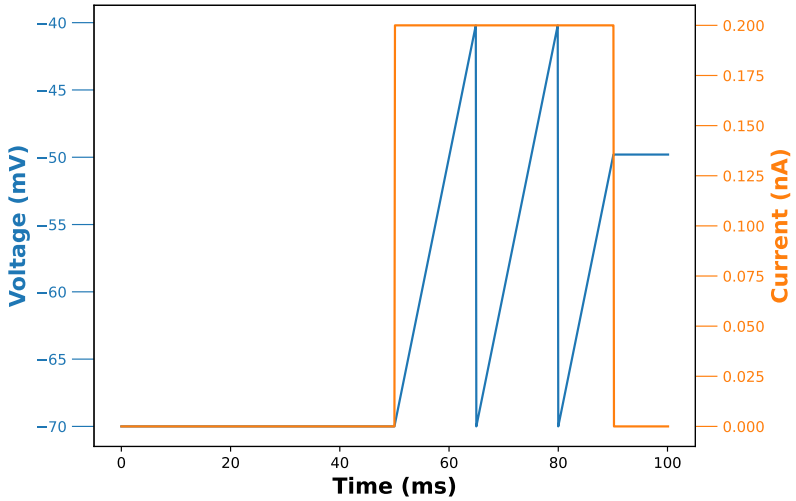


Figure 5.24: Integrate and fire neuron model simulation against the injected current

An exemplary integrate and fire neuron's simulation is given in Figure 5.24. A current injection is given to the IF neuron, where the voltage increases linearly only after 50 ms. Once it reaches the predefined threshold of -40 mV, the voltage will be reset back to the reset potential -70 mV. It could also be noticed that since IF neuron does not have leakage, once the current injection stops, the voltage will remain where it is. No decay is observed for the neuron.

Application and realistic use

A practical implementation of the variable that changes by the rule of the differential equation is Euler's method. This basically can be summarised as:

$$f(x_0 + \Delta t) = f(x_0) + \Delta t \; f'(x_0) \tag{5.19}$$

This approach predicts the value of the next time step by adding the value of the current time step with the derivative multiplied by the time step. This applies to functions that are continuous and also demonstrate continuity in terms of their derivative. This is illustrated in Figure 5.25. Such methodology gives a fairly accurate approximation of the actual value for the variable at a cheaper cost given that each time step for the iteration is small enough so that the error is negligible. Because no solution to the ordinary differential equation (ODE) is needed, the cost to extrapolate based on the derivative is very low.

In previous sections, neuron models have all been assumed to receive current injection as input. However, in the real application, the neurons should be connected through synapses, which are the components to convert spikes into current flow to feed the neuron. Therefore, as for the actual simulation, the neuron is expected to receive irregularly shaped current flows instead of a square wave used in the simulation above.

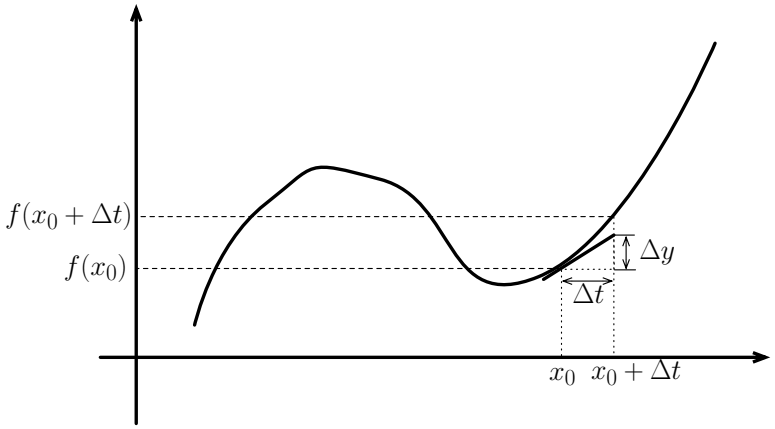


Figure 5.25: Euler's method of extrapolation

In general cases, the intensity of a post-synaptic current to a single neuron should be the sum of the input current it receives from all the synapses:

$$I_i(t) = \sum w_{ij}\alpha_j(t) \tag{5.20}$$

Since the current generated by synapses is a time-dependent function, each current flow generated from pre-synapse is represented as in $\alpha_j(t)$, the efficacy it has on the post-synaptic neuron is annotated as w_{ij}. Each time there is a spike, the generation is triggered and a current is then transmitted just as indicated in Figure 5.26. As for the generation of the current

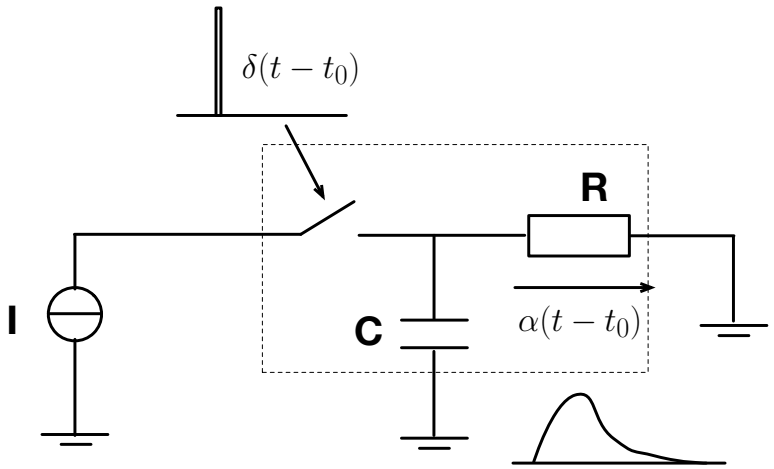
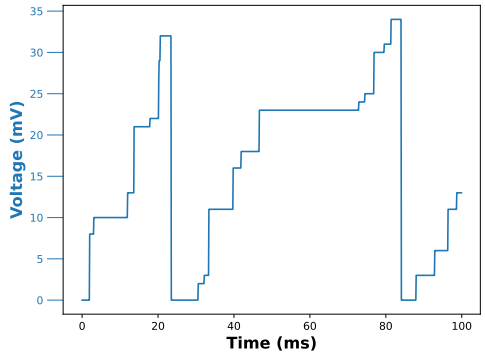


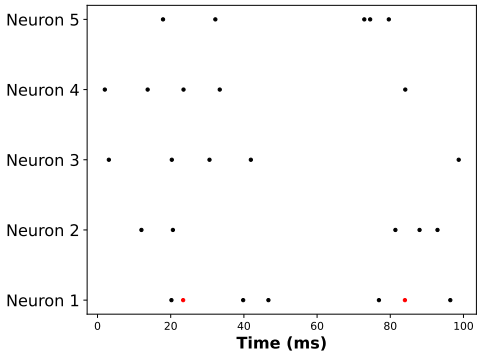
Figure 5.26: Simplified synapses circuit model

from synapses, a conductance-based model is usually applied:

$$\alpha_j(t - t_0) = -g(t - t_0)[U_i(t) - E_{syn}] \tag{5.21}$$



(a) Membrane potential of an IF neuron with simplified synapses



(b) Spikes of the input (black) and output (red) of the IF neuron

Figure 5.27: A simple simulation of an IF neuron

Because the current is only generated when there is an input spike from the pre-synaptic neuron, the current should have the same time property as the spike. If the spike is generated at time t_0, then the switch to generate current will be turned on temporarily at t_0. This explains the timing information of the conductance g described in Equation 5.21.

Besides, how this injection is perceived by the neuron is also dependent on the state this neuron is in. That is to say, the "perceived" current is also dependent on the membrane potential of the post-synaptic neuron. This has been annotated as in the multiplication of the conductance and the potential difference, where $U_i(t)$ marks the membrane potential and E_{syn} is the reversal potential of this excitatory synapse.

The excitatory synapses reflect the positive current injection the afferent spikes incur, the opposite will be called inhibitory synapses. Normally, E_{syn} will be set higher than the threshold value, a point where membrane potential cannot reach. This means that when the neuron is at the resting potential, a single spike will create a bigger impact on the membrane potential than the case where membrane potential is already high before the spike.

This synapse mechanism can sometimes be ignored or simplified even more to produce an IF neuron with step-shaped potential. Assuming that the impact each spike brings to the neuron is the same regardless of the neural state, the synapse can be simplified further. A simple simulation of an IF neuron is given in Figure 5.27. It is clear that the voltage accumulation is purely linear and no leakage is present for the neuron. This neuron has 5 different afferent synapses, each one has a different weight. The spike times for this input are displayed as black dots in Figure 5.27b, while the red dots indicates the firing time for this IF neuron when it reaches the voltage threshold.

This highly simplified neuron and synapse model has enabled much higher applicability for spiking neurons. And they are believed to be the next generation of neuron model since both the computational cost and the power consumption are low.

There have been research work that employs SNNs in computer vision (Rueckauer, Lungu, Hu, Pfeiffer, & Liu, 2017) (Sengupta, Ye, Wang, Liu, & Roy, 2019), robotic control (Gutierrez-Galan, Dominguez-Morales, Perez-Peña, Jimenez-Fernandez, & Linares-Barranco, 2020) and some simple language processing. There have also been attempts to convert the powerful transformer models into their spiking version given that large language model like generative pre-trained transformers (GPT) shows significant potential.

5.4.3 Learning in SNN

Except for cortical neural simulations, people have also been using SNN for deep learning tasks like image classification (Zhou, Chen, & Ye, 2019), object tracking (Luo et al., 2020) etc. However, learning in SNNs is still an open research topic, unlike ANNs where backpropagation and gradient descent are widely used. There are still some training approaches used that prove effective.

By big category, there are mainly supervised and unsupervised training. But a more commonly used way to categorise the training in SNN is by 3 approaches:

- Direct training
- ANN-SNN conversion
- Native SNN training

Direct training

Researchers have been trying to associate spiking neurons with traditional training methods. The big reason why backpropagation and gradient descent are not applicable is because the "activation function" of the spiking neurons are non-differentiable. This is because the reset and firing mechanism in IF neurons cannot provide continuity for the "activation function". Direct training methods aim to bridge the gap between the "non-trainable" IF neurons and the prevalent backpropagation. There are various solutions to this problem, and they all demonstrate some advantages to some extent.

One possible approach is to not take derivatives of the "activation function", but the times the spikes are generated towards the weight (Bohte, Kok, & La Poutré, 2002). Despite the fact that discrete spikes are discontinuous, the time is continuous and differentiable. This is the first proposed approach to apply backpropagation to SNN training and is named *SpikeProp*. Naturally, the error of the loss is formulated with the arrival time of the spike. One can calculate the gradient of the loss with respect to the spike time and then use this to update the neural weights, which results in a change of the membrane potential and ultimately change in the spike time. The drawback of this approach is that each neuron must generate one and only spike to be updated. This is due to the fact that the calculation of the gradient replies to the spike time. In the situation of no spikes generated, the loss of this neuron would be invalid and therefore weights cannot be updated during backpropagation.

Another successful training method looks at spiking neurons from a different perspective by unfurling the network in the dimension of time.

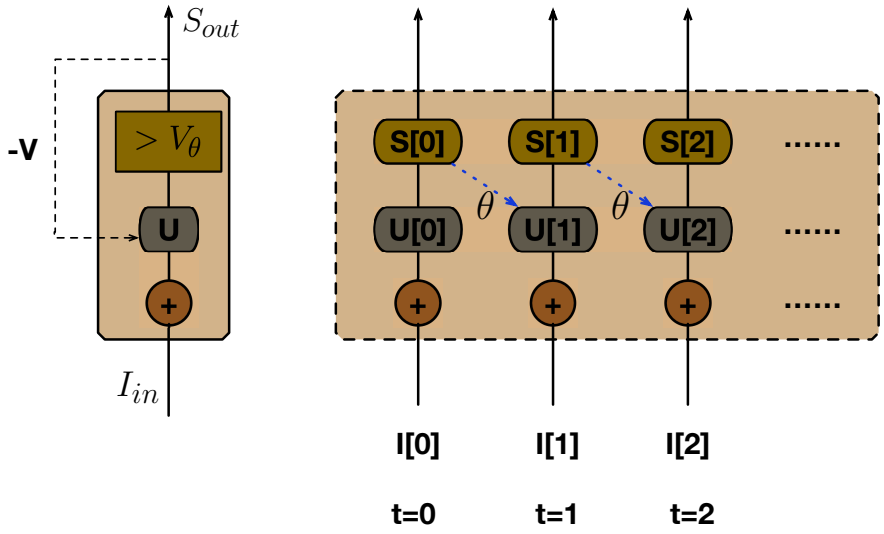


Figure 5.28: Spiking IF neuron recurrent connection unrolled

As depicted in Figure 5.28, an IF spiking neuron accumulates the potential when it receives input. Once that value reaches the threshold V_θ, it will generate a spike, which will deduct the potential with the threshold value. This can be formulated as in:

$$U[t+1] = U[t] + I[t+1] - S[t]\theta \tag{5.22}$$

where $U[t]$ is the potential from last time step, $I[t+1]$ is the input from current step while $S[t]\theta$ shows the voltage subtraction from firing.

This is inspired by how recurrent networks are trained in ANN paradigm. The unrolled big network will then be trained with the generalised backpropagation, which is also known as backpropagation through time (BPTT) (Shrestha & Orchard, 2018), (Esser et al., 2016). Notice that if there is also leakage in the IF neuron, the computational graph should also include the connection between two potentials at the adjacent time steps, i.e. $U[i]$ and $U[i+1]$. This would update the Equation 5.22 with the decay constant β:

$$U[t+1] = \beta U[t] + I[t+1] - S[t]\theta \tag{5.23}$$

During backpropagation, the error should travel from the final output back to each leaf node by the reverse order of the time step.

This is normally combined with the surrogate gradient method to address the nondifferentiability issue of spiking neurons. Looking closely at the function of the spike activity against the membrane potential, it is not difficult to conclude that there are only two cases at a given time step.

$$S = \begin{cases} 1, & U \geqslant V_\theta \\ 0, & U < V_\theta \end{cases} \tag{5.24}$$

That is, the spike is either emitted ($S = 1$) or not ($S = 0$) depending on the membrane potential value. If we are trying to calculate the derivative of spike activity with respect to membrane potential, a Dirac function will be presented as in Figure 5.29. The derivative should satisfy

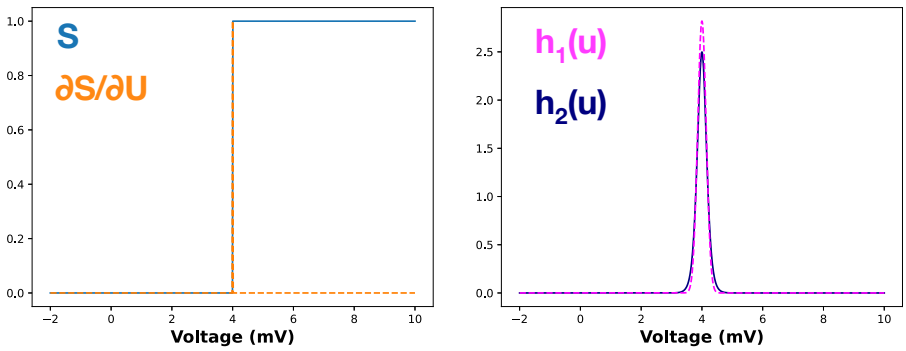


Figure 5.29: Spike activity and its derivative against voltage (left), derivative approximation(right)

$\frac{\partial S}{\partial U} = \{0, \infty\}$, which means the derivative should equal 0 everywhere else except when $U = V_\theta$ where it should equal positive infinity. This should also satisfy the following integration equation:

$$\int_{-\infty}^{+\infty} \frac{\partial S}{\partial U} dU = 1$$

Based on the constraint made in this equation, the approximation of this derivative can be expressed as in Figure 5.29 where 2 approximation functions are given.

$$h_1(U) = \frac{1}{a_1} \frac{e^{\frac{V_\theta - U}{a_1}}}{(1 + \frac{V_\theta - U}{a_1})^2} \tag{5.25}$$

$$h_2(U) = \frac{1}{\sqrt{2\pi a_2}} e^{-\frac{(U - V_\theta)^2}{2a_2}} \tag{5.26}$$

Where coefficients a_1 and a_2 are positive and adjustable to determine how close the approximation can be. In other words, the width of the "peak" is controlled by these coefficients. The closer they are to 0, the "thinner" and "taller" the curve will be. During application, these approximations will only be applied at the backward pass to replace $\frac{\partial S}{\partial U}$, but the forward pass will still be the same. However, this technique cannot solve the issue with dead neurons, which do not fire at all during forward pass. This will lead to the neuron being stuck forever because no updates can be assigned to the relevant weights.

In conclusion, direct training has become one of the most popular techniques when it comes to the learning rules in SNN. This has enabled the user to deploy a deep SNN with the existing dominant training method, backpropagation. The advantages are that one can have a fairly quick-responding SNN which does not need too many time steps (Wu et al., 2019) and securing a good accuracy (Wu, Deng, Li, Zhu, & Shi, 2018). The drawbacks are equally dominant as its strengths. Since the training unrolls the SNN into a recurrent neural network, the training for this can be computationally costly and is limited to a relatively shallow network given restrained resources.

ANN-SNN conversion

By comparing the rate-coded simplified IF neuron and a general artificial neuron with an activation function of ReLU. One may argue that they share a strong resemblance and many similarities.

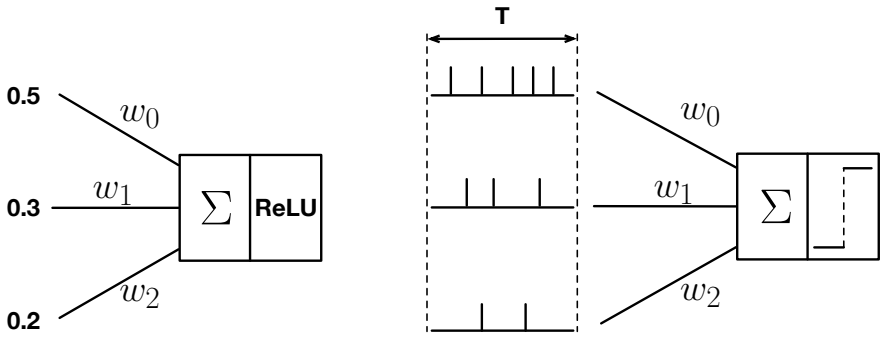


Figure 5.30: A side-by-side comparison of an ANN neuron with ReLU and an IF neuron

Just as illustrated in Figure 5.30, the artificial neuron displayed on the left demonstrates the functionality that is presentable in the following:

$$a = ReLU(0.5w_0 + 0.3w_1 + 0.2w_2) \tag{5.27}$$

where a represents the output activation. While the spiking neuron that runs time period of T can be formulated as follow:

$$s = \frac{5w_0 + 3w_1 + 2w_2}{V_\theta} \tag{5.28}$$

s here stands for the total amount of spikes in the time period of T. If we are looking at an average spike rate S_T over time period T, we would have the following from Equation 5.28 just by dividing both sides of the equation by T:

$$S_T = \frac{s}{T} = \frac{5w_0 + 3w_1 + 2w_2}{T * V_\theta} \tag{5.29}$$

If we happen to have $T = 10$ and $V_\theta = 1$, then Equation 5.29 would transform into:

$$S_T = 0.5w_0 + 0.3w_1 + 0.2w_2 \tag{5.30}$$

Apparently, the parameters setting for T and V_θ here are for demonstration purpose only. One can also set other values for these two values, but it will not change the fact that the average spiking rate S_T is linearly related to the activation value of the corresponding ANN neuron. And since the spiking neuron can only fire when there is positive membrane potential that is bigger than V_θ, S_T has to always be non-negative just as ReLU function requires. Therefore, the strong connection between these two types of neurons has allowed researcher come up with the idea of mapping properly trained ANNs into their SNNs forms with the conditions that all the activation functions should be ReLU except for the last layer.

Following this idea, Diehl et al. proposed the first ANN-SNN conversion algorithm that proves effective with both fully-connected networks (FCN) and convolutional neural networks (CNN). This essentially described two conversion schemes with the weights of a pre-trained ANN that has no biases. One is called weight-based conversion, while the other one is more activation-based or data-based as called in the original paper. It was also shown in the comparison that the activation-based conversion outperforms the weight-based one in terms of performance and latency and therefore is preferred. The activation-based one is basically described in Algorithm 2. They demonstrated that with proper parameters tuning and ample simulation time steps, both converted spiking version of FCN and CNN can deliver high accuracy on the handwritten digit dataset MNIST.

The limitations of this work was later addressed by the extension to this conversion algorithm (Rueckauer, Lungu, Hu, & Pfeiffer, 2016). In this extension, the author included the conversion method for ANN neurons with biases and proposed a more robust conversion scheme. Besides, the conversions for batch normalisation layer, max pooling layer and softmax layer are also included. As for the biases, they are taken as the same parameter as weight for

Algorithm 2 Data-based Weights scaling for Spiking Neural Network Layers

$previous\_factor = 1$
for each $layer$ in $layers$ **do**
    $max\_wt = 0$
    $max\_act = 0$
    for each $neuron$ in $layer.neurons$ **do**
        for each $input\_wt$ in $neuron.input\_wts$ **do**
            $max\_wt = \max(max\_wt, input\_wt)$
        end for
        $max\_act = \max(max\_act, neuron.output\_act)$
    end for
    $scale\_factor = \max(max\_wt, max\_act)$
    $applied\_factor = scale\_factor / previous\_factor$
    for each $neuron$ in $layer.neurons$ **do**
        for each $input\_wt$ in $neuron.input\_wts$ **do**
            $input\_wt = input\_wt / applied\_factor$
        end for
    end for
    $previous\_factor = scale\_factor$
end for

normalisation and converted together with weights:

$$\lambda^l = max[a^l] \tag{5.31}$$

$$\hat{W}^l \leftarrow W^l \frac{\lambda^{l-1}}{\lambda^l} \tag{5.32}$$

$$\hat{B}^l \leftarrow \frac{B^l}{\lambda^l} \tag{5.33}$$

where a^l represents the activation values in the layer l, $\hat{W}^l$ is the converted SNN weights as opposed to the original ANN weights W^l, and same applies to the biases $\hat{B}^l$ and B^l. The robust normalised mentioned above actually means the careful selection of λ^l. Since the distribution of a single layer's activation values across all the datasets are not following a normal distribution or a uniform distribution, this way of normalisation could sometimes bipolarise the corresponding spiking rate. For example, it is normally the case that the max activation in a layer is much higher than the 99-th percentile, and the occurrence of the max activation is highly scarce. This would mean that the max activation cannot represent the max value of the distribution of most data, i.e. the maximum is the outlier. Therefore, to have a more robust conversion, the author proposed using the 99-th percentile or 99.9-th percentile as the key parameter

$$\lambda^l = percentile(a^l, 99) \tag{5.34}$$

This ANN-SNN conversion method works well for most cases and takes advantage of the existing ANN frameworks for shadow training therefore the conversion can be speedy and much less costly in terms of computational resource. There is some accuracy loss from the original ANN, but overall it will not display a huge accuracy gap between ANN and SNN. The main drawback for this method is the overhead one has to go through to get the SNN and the latency of the converted SNN is higher than direct training.

Native SNN training

Native training or local learning methods for SNN mainly look at the relationshiop between spatio-temporal features and weight updates. This is highly attractive for neuroscience to understand how neural networks are trained because biologically there is no backpropogation in neural systems. Another reason why it is preferred is because it offers a hardware-efficient way to train a deep neural network. The training agent does not have to go through other neurons or the entire network for the weight updates.

In general, local learning rules mainly include spike-time dependent plasticity (STDP) (Markram, Lübke, Frotscher, & Sakmann, 1997) and its variations that are derived from Hebbian learning rules (“Hebb, D. O. The organization of behavior: A neuropsychological theory. New York: John Wiley and Sons, Inc., 1949. 335 p. $4.00”, 1950), which mainly summarise as "neurons that fire together wire together". And the function of STDP is highly dependent on the architecture of the SNN. When applied in a competitive fashion, unsupervised STDP is able to perform simple clustering tasks (Masquelier, Guyonneau, & Thorpe, 2009) (Nessler, Pfeiffer, Buesing, & Maass, 2013). Researchers have been inspired by this learning mechanism to install the local learning rule in each layer and extend the learning to the whole network in a layer-by-layer fashion. Recent publications showed that Spiking Convolutional Neural Networks (SCNN) trained by STDP are capable of performing image classification tasks just as well as its counterparts (Kheradpisheh, Ganjtabesh, Thorpe, & Masquelier, 2018) (Lee, Srinivasan, Panda, & Roy, 2019). The combination of reward-modulated STDP (Mozafari, Ganjtabesh, Nowzari-Dalini, Thorpe, & Masquelier, 2019) can also enable the network to learn in a supervised way depending on the specific application. In some recent publications (Thiele, Bichler, & Dupret, 2018), it is event possible to train the whole SNN without going layer-by-layer in SCNN.

Apparently, the advantages of the mentioned native SNN training methods are as pronounced as their disadvantages. They can offer great potential for efficient training of SNN, no extra overhead, no shadow training for another ANN. It is capable of exploiting the spatio-temporal information that none of the other training methods can. If well established, they would be the best choice to deploy as an on-chip learning mechanism that will update the weights as they

see new data. The disadvantage is obviously that there is not enough infrastructure support so it is difficult to implement and takes evidently longer than all the other methods. Besides, these methods cannot deliver accuracy as high as their conventional counterparts like ANN or CNN.

5.4.4 Neuromorphic computing infrastructure

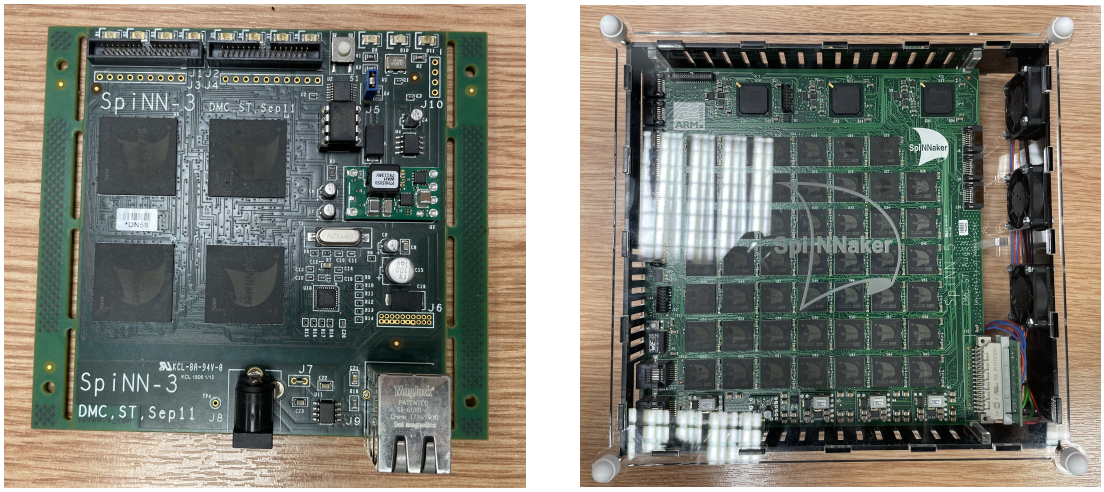
To carry out simulation and research work for SNN, one must have proper infrastructure support either in software or hardware or both. There is hardware that is specifically designed for SNN simulation and training. Some of the prominent candidates will be named and briefly explained in this section.

Hardware development

SpiNNaker (Painkras et al., 2013), which is short for Spiking Neural Network Architecture, is a substantially parallel computing system designed to perform cost-efficient and reconfigurable neuroscience experiments simulation. The massive million-core machine is capable of simulating billions of neurons and a trillion synapses thanks to the scalability of a single multiprocessor chip. Each chip encapsulates 18 ARM968 processors, which are surrounded by the light asynchronous communication system. One single chip only costs 1 W at its peak consumption when all 18 processors operate simultaneously at 180 MHz. They are compatible with the proposed Python package sPyNNaker to deliver custom SNN simulation, which shall be discussed later. An example of two SpiNNaker boards with different size is presented in Figure 5.31. The 4-node board has 72-processors in total and can simulate up to 10,000 neurons. While the 48-node board is capable of simulating up to a few 100,000 neurons with the 864 processors embedded in.

TrueNorth (Akopyan et al., 2015) is an IBM project that produces a non-von Neumann, low-power, highly-parallel, scalable, fully digital chip, which incorporates 4096 neurosynaptic cores. One package of a chip covers 1 million digital neurons and 256 million synapses tightly interconnected by an event-driven routing infrastructure. It offers a 65 mW power capsule, which is impressive due to its mixed asynchronous-synchronous design. The computation of the neuron is synchronous while the communication and transmission of events between cores are designed in an asynchronous fashion. This chip also offers programmable capability to configure custom SNNs needed for specific applications.

Loihi (Davies et al., 2018), the neuromorphic computing chip from Intel, features 128 neuromorphic cores and 3 embedded x86 processor cores in a mesh, which can be extended by connecting more Loihi chips through the off-chip communication interfaces. All the cores inside one chip are connected by Network-on-Chip (NoC) system, which asynchronously



(a) A 4-node SpiNNaker board **(b)** A 48-node SpiNNaker board

Figure 5.31: 4-node SpiNNaker board and 48-node SpiNNaker board

delivers the messages and events in packets. Each neuromorphic core implements 1,024 spiking neuron units as well as a learning engine that is microcode programmable to support local learning rules including pair-wise STDP, triplet STDP and reinforcement learning with synaptic tag assignments.

SpiNNaker 2 (Mayr, Hoeppner, & Furber, 2019), as the successor to SpiNNaker project, is still under research and development. Even though no detailed floor-planning or chip die design is revealed, the specification claims that every chip will still be an ARM-based processors system with 144 Cortex-M4F cores in each. The goal is to build a big neuromorphic computing centre with 10 million cores. Some publications have already been using the early prototype of SpiNNaker 2 for learning tasks. For example, a radar hand gesture recognition system was implemented on a SpiNNaker 2 prototype with an FPGA (J. Huang et al., 2022). A commonly used training method called E-prop was also implemented on the SpiNNaker 2 prototype to test its resource utilisation and performance (Rostami, Vogginger, Yan, & Mayr, 2022).

Software development

Unlike the well-developed infrastructures for ANN, SNNs do not have a mature framework like TensorFlow or PyTorch. Fortunately, there are ample open-source simulation packages available in Python or C++ to facilitate researchers' experiments and simulations. They vary from pure software solutions to software tools for neuromorphic hardware or graphic processing units (GPU).

NEST (Gewaltig & Diesmann, 2007), as one of the earliest spiking neural network simulation tools, was proposed in 2007 and stands for NEural Simulation Tool (NEST). The tool itself was first proposed (Plesser et al., 2007) to solve the challenges encountered during simulating large-scale neuronal networks:

- Large amount of connections need to be stored in a distributed way
- Spikes buffering before transmission to make sure the latency and information is correct
- Simulation needs to be deterministic and reproducible
- The object-orient implementation should allow user to perform network and machine level optimisation

This simulator tool is written in C++ and offers increasing numbers of interfaces over the years for its usability. The native language to use this tool is by SLI, a scripting language that can be typed in terminal in one session to construct model structures for the simulation. To enrich the user experience, a python package called PyNEST was introduced (Eppler, Helias, Muller, Diesmann, & Gewaltig, 2009) to facilitate quick network construction. It basically acts as a compiler to translate python scripts into data and SLI commands to NEST simulator. It is a powerful simulator that offers highly parallel simulation on multicore and multiprocessor platforms. With the distributed computing established, it is also easy to scale the network across platforms. It is still being actively maintained and supported by its community and researchers despite its age.

Brian (Goodman & Brette, 2008) is a purely software-based simulator written in Python. The biggest advantage of this tool is its flexibility. It allows users to define customised neuron model by stating the variable dynamics using ordinary differential equations. In the meantime, it offers a reasonably fast simulation speed despite that it is a clock driven simulator meaning every state variable gets updated at each time step. Its successor Brian 2 (Stimberg, Brette, & Goodman, 2019) embraces a complete rewrite of Brian and specifically addresses the simulation speed with the support for multi-thread computation. Even though its high flexibility and accessibility are desirable, the code still runs on CPU and cannot be accelerated with parallel computation. However, it performs very well on light training or simulation tasks for experiments.

PyNN (Davison et al., 2009) is technically not a simulator but a simulator-independent SNN modelling package that standardises the syntax of constructing SNNs. In other words, a same model that constructed following PyNN API can run on different simulators without changing the code. Currently supported simulators include NEST, Brian, SpiNNaker etc. This has been achieved by providing a standardised set of high-level abstraction, which promotes code reuse and sharing. But in the meantime, it still allows access to the low-level neurons and synapses for flexibility.

Nengo (Bekolay et al., 2014) is a simulator following the Neural Engineering Framework proposed by Eliasmith and Anderson, which defines computational framework for understanding and modelling neural systems. It was previously drafted in Java till version 1.4, the new simulator is now written in Python. Similar to the implementation of PyNN, Nengo also separates the modelling and simulation. This means Nengo offers a standard set of interface to build neuronal network model. Once all the information needed is encapsulated into the model object, it will be passed to the simulator which is independent from the front-end interface. Though a reference simulator is implemented within the package, one can migrate the simulation to other highly parallel computing platform such as GPU or neuromorphic hardware. For example, in the paper introducing Nengo 2.0 (Bekolay et al., 2014), the author developed Nengo for OpenCL to run benchmark on an NVidia GTX280 GPU, which significantly boosts the simulation efficiency. The official support for other simulators now include Loihi, SpiNNaker, OpenCL, Message-Passing Interface (MPI) and even FPGA. This simulator offers both frontend and backend interface and has strong extensibility and flexibility. It is still being actively developed and now has an extensive coverage of various projects and applications.

sPyNNaker (Rhodes et al., 2018), the official simulating package for the SpiNNaker, has been existing since the advent of the neuromorphic hardware itself, which is around 2014. The sPyNNaker API consists of two stacks: one primarily written in Python for the host system, and the other in C for the SpiNNaker machine. Users utilise the PyNN interface to develop an SNN model, which is then processed by the Python-based sPyNNaker software to a format compatible with SpiNNaker machines. Simulations run on the SpiNNaker machine using an event-driven operating system. Once the simulation is finished, the results data are retrieved back to the host system for further processing and visualisation using the PyNN API. This software package is now under the migration work for the upcoming SpiNNaker 2, the successor of SpiNNaker. But it is still available for public use on the intended SpiNNaker.

BindsNET (Hazan et al., 2018) is another Python package that offers SNN simulation framework. However, this one is specifically steered towards machine learning and reinforcement learning. It allows for quick construction and simulation of spiking networks, with a user-friendly and succinct syntax. What is special about this work is it is developed based on PyTorch, a well received deep learning framework. This would mean that users can employ BindsNET just like PyTorch, and implement SNN on CPUs and GPUs efficiently. It can even switch to other computing backends like TensorFlow and SpiNNaker if required. Just like other solutions, it is running in C++ at the lower level for good performance and exposes the friendly frontend python interfaces to users.

SpikingJelly (Fang et al., 2023) is yet another open-source Python package that is based on PyTorch designed for SNN simulation and training. One can use this tool to simulate SNN on both CPUs and GPUs thanks to the CUDA kernels from PyTorch and the developer research team. It incorporates both ANN-SNN conversion and surrogate gradient descent modules for users to either do shadow training or direct training through time. Apart from the GPU and CUDA support, it also provides the module that converts defined SNN object into a format that is supported by neuromorphic hardware. So far the supported hardware include Intel Loihi and Lynxi KA200. It is claimed that other neuromorphic platform can be also supported with necessary exchange module developed.

snnTorch (Eshraghian et al., 2023) shares the similar genes as the two counterparts mentioned above. It is also an SNN simulator that is based on PyTorch to employ the GPU accelerated matrix computation. Moreover, snnTorch is optimised for GraphCore's Intelligence Processing Unit (IPU) acceleration. It was initially developed by University of Michigan and now maintained by University of California, Santa Cruz. As a new candidate, it is gaining more attention quickly for potential alternatives when it comes to SNN computation.

5.5 Why spiking neural networks?

It has been well explained in this chapter about how event-based computing works as a totally different paradigm. But why? This is because the nature of this computing aligns very well with our previously mentioned objectives for this project.

In chapter 3, we reviewed the RIID algorithms and concluded that most algorithms are focused around the gamma-ray spectroscopy where gamma-ray spectra are taken as the input for radionuclides analysis. Since gamma-ray spectra essentially represent decay events at different energy levels, the whole spectrum can be considered as a cluster of decay events. Besides, the collection of the spectrum needs a period of time to accumulate these events, and it is updated by pre-defined time step, i.e. every second or 0.5 seconds. In other words, these events also have temporal information. These features can therefore be associated with the encoding of address-event representation or AER naturally. If we encode different energy levels as the "address", the occurrence of an decay event at a certain level as the "event", the scintillator and front-end pre-processing (photomultiplier tube etc.) can be perceived as an event-based sensor.

In chapter 4, we put forward the realistic case where energy consumption can be a problem for the hand-held RIID device. In the existing technology, the analysis was performed by a software-based implementation. This has to execute in a frame-based fashion and normally requires the complete collection of a whole spectrum in a long period of time. The comparison between frame-based and event-based computing can be found in Figure 3.12 . However, if

an event-based implementation can be done, the "analyse-as-collection-goes" will eventually benefit the power saving. This is because the computing can stop earlier than normal time required by the frame-based algorithm and also the computing resources can stay idle when there is no event.

SNN, as a representative event-based algorithm, has demonstrated strong potential for this task for its capability of learning as reviewed in this chapter. What's more, it is easy to implement the algorithm with the support of both available software and hardware infrastructure. Therefore, we have chosen SNN as the main direction of implementation for this application.

5.6 Conclusion

In this chapter, the background knowledge for event-driven processing, data acquisition and encoding were discussed at the beginning, where level-crossing sampling was mainly explained and its potential circuitry implementation. There are also other types of sampling mentioned where the mechanisms were explained.

This is followed by the explanation of address-event representation (AER), which has been widely used in different even-based computations. The addition of address to the representation indicates clearly the source of the event, which facilitates the computation flow.

One representative event-based processing paradigm is neuromorphic computing, where reconstruction of biological neurons communicates through spikes instead of real values. This fashion of computation has already inspired the invention of neuromorphic sensors like visual sensors and auditory sensors. The most fundamental composing unit, spiking neurons have also been mathematically modelled into different models in terms of the bio-plausibility. Constructing these neurons can demonstrate a certain level of learning capability. Brief introduction about training approaches for SNN was also given. To better support the SNN training and simulations, both new hardware and software infrastructures were built.

To elaborate why SNN has been chosen, the connection between SNN and RIID has been discussed. This justifies the choice of SNN as the main direction and demonstrates that SNN, as a candidate, has a great potential for saving power and increasing the assessment speed on the hand-held device.

Chapter 6

Unsupervised SNN for RIID

It has been well reviewed in the last chapter about the learning for spiking neural networks (SNNs). Here we present one way of implementing SNN for the RIID task, which is the unsupervised training method which employs the native SNN learning mechanism: spike timing dependent plasticity (STDP). This implementation used STDP to perform feature extraction from the processed histogram data. With this established, the accuracy could reach 80% during training and overall testing accuracy of 72%. The whole network was implemented on SpiNNaker, a spiking neural network emulation platform. This work shows that unsupervised STDP, an SNN native training method, can be applied to the classification task of RIID to provide event-based training as well as inference.

6.1 Introduction

Thanks to the constant development of deep learning technologies and algorithms and increasingly powerful high-performance computing hardware such as high-throughput graphics processing unit (GPU) and dedicated machine learning microchips (Reuther et al., 2019), artificial neural networks (ANNs) have made breakthroughs on a range of tasks in many different applications (Esteva et al., 2021) (Katare & El-Sharkawy, 2019) (Siniscalchi, Svendsen, & Lee, 2014). Commonly used methods of training and using these networks have been widely criticised for being notoriously energy intensive (Strubell, Ganesh, & McCallum, 2019). By comparison, spiking neural networks (SNNs) are believed to be the next generation of neural networks (Ghosh-Dastidar & Adeli, 2009). They have attracted significant amounts of attention for their potential energy efficiency in event-based tasks (Pfeiffer & Pfeil, 2018), in part due to their biological plausibility from being computational neuroscience models. In ANNs, each value needed for the neural activity is normally represented in 32 or 64 bit floating point precision. From hardware point of view, ANN is heavily multiplication addition calculation oriented, high precision would increase the power consumption greatly. Unlike their counterparts, SNNs work in a way where communication between neurons is through binary

spikes over time instead of multi-bit static activation as ANNs do. This could enable the whole network to perform in an event-based style. That is to say, the processing unit will not do the computing when there is no event. Consequently, the potential power savings are promising compared to the traditional computing as the computation takes place in a frame-based style.

Radioisotopes identification (RIID) is an area that has seen work carried out to understand. There are approaches like expert interaction (Knoll, n.d.), region-of-interest matching (P. A. Aarnio, Routti, & Sandberg, n.d.) and template matching (Göttsche, Schirm, & Glaser, n.d.), which in part or entirely has to be carried out manually. In addition, methods that use ANNs to do RIID have been introduced recently (Kamuda et al., 2017). Naturally, the application of machine learning to RIID is appealing given the fact that many processes are not entirely automated. For example, in (Kamuda et al., 2017), a three layer ANN was used to perform relatively accurate inference on 32 different radioisotopes. On the other hand, RIID could be considered an event-based task because of the way scintillators work. They give off individual photons when triggered by gamma-ray. These photon events later on captured by photon detector could be encoded as stochastic binary spikes with timing information. However, using event-based computing to address this specific topic has not been fully explored, especially using SNNs.

As mentioned above, the SNN may be the next generation of neural network and a promising candidate for low-power edge artificial intelligence (AI). This is the motivation for their use in this work.

6.2 Related work

Spiking neural network research has been active for several decades. After the birth of the ANN, there has been attempts to build neuron models with capability of processing temporal information. This is because from the bio-mimicry point of view, the ANN neuron model has been oversimplified, not faithfully modelling the biological neuron. Spiking neuron models were developed incorporating more detail from neuroscience, notably the temporal behaviour of neurons. This greater closeness to biology has led to spiking neurons as being referred to as the third generation neuron models (Maass, 1997). Numerous research works on the neuron modelling and network architecture have been published ever since (Tavanaei, Ghodrati, Kheradpisheh, Masquelier, & Maida, 2019). For example, in (Diehl & Cook, 2015), Diehl and Cook proposed the unsupervised SNN for hand-written digits recognition. This is a highly classic model that shows the organic combination of SNN and spike timing dependent plasticity (STDP) for application. It has inspired a series of publications that employed similar

SNN structures (Amirshahi & Hashemi, 2019) (Kulkarni & Rajendran, 2018) (Iyer & Basu, 2017) (Saunders, Siegelmann, Kozma, et al., 2018). However, most of these works have been focusing on network optimisation and used the network simply for hand written digits classification.

To date, there have been few attempts to apply SNNs to the task of RIID. Huang *et al.* implemented a field-programmable gate array (FPGA) design of a simple three-layer densely connected SNN to finish the task with overall accuracy of 87.6% (X. Huang, Jones, Zhang, Xie, et al., 2020) and implemented of convolutional SNN on an FPGA (X. Huang et al., 2021) to achieve the classification of 8 different radioisotopes, reaching an accuracy of 90.6%. These SNNs were generated through ANN-SNN conversion, which trains a restricted ANN and then does a conversion to an SNN. This training approach has proven to be working, but the overrun to update the network could be cumbersome.

This work was built upon the similar architecture and training approach in (Diehl & Cook, 2015) but with the application of RIID on the same data set used in (X. Huang, Jones, Zhang, Xie, et al., 2020). Advantages of such approach is that it could provide a less complex alternative training procedure. Meanwhile, this local training rule offers the possibility of on-device training. The objective is to address the applicability of applying SNNs on RIID in a purely event-based way by presenting an SNN trained with unsupervised STDP.

6.3 Methodology

The whole architecture of the neural network implemented in this work is a three-layer SNN with one input layer, one excitatory layer and one inhibitory layer. The input layer consists of 400 neurons that could generate Poisson spike trains according to the input rate, which will be explained in details later. This layer encodes the data information in the rate-based encoding, but this layer would be adapted if ever used in a production system with events coming from an event-based sensor. What comes after are the excitatory layer and inhibitory layer, which both have 81 neurons. All the neurons are modelled as leaky integrate and fire (LIF) neurons with conductance based synapses connection.

As illustrated in Figure 6.1, the projection from each population to the other could be listed as follow:

- Input layer to excitatory layer, all to all connection with STDP applied
- Excitatory layer to inhibitory layer, one to one connection with a fixed weight
- Inhibitory layer to excitatory layer, one to many connection with a fixed weight

The one to many connection starts from one neuron in the inhibitory layer back to all neurons in the excitatory layer except the one it receives the stimulus from. This connection applies to all the neurons in this layer. Since the purpose of this layer is to bring lateral inhibition, the synapses in this projection are defined as inhibitory synapses unlike the rest of this SNN, which are excitatory connections.

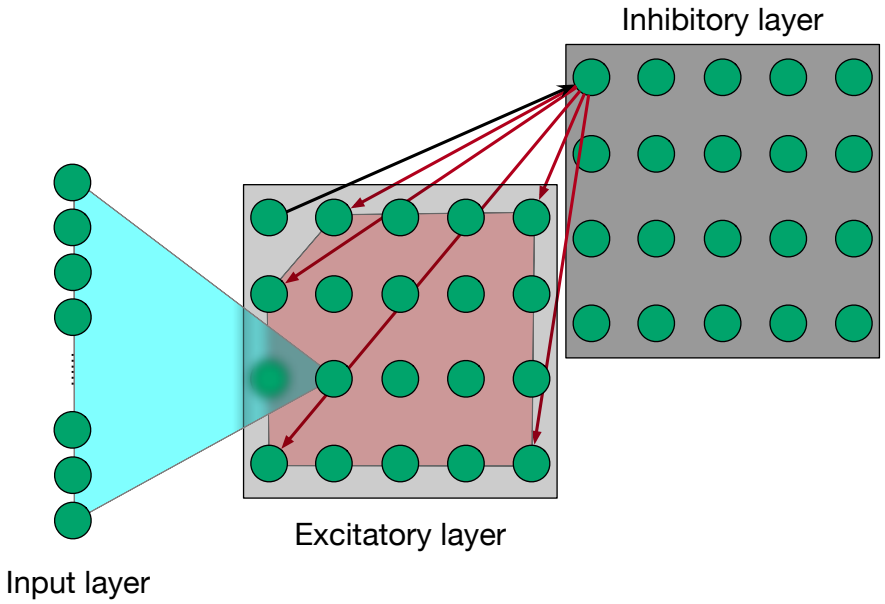


Figure 6.1: Architecture of the unsupervised SNN

6.3.1 Leaky integrate and fire (LIF) neuron with conductance-based synapses

As the most basic composing unit of the SNN, a conductance-based leaky-integrate-fire (LIF) neuron model was employed. The model could be mathematically described as in the Equation 6.1. In the formula, the membrane potential V is following the exponential decaying rule on the decay time constant τ. V_{rest} is the resting potential of the neuron, which is the potential level when the neuron is at rest and has been set at -65 mV. V_{exc} and V_{inh} are the reversal potential for excitatory and inhibitory synapses and have been set at 0 V and -100 mV respectively, which are specific to the conductance-based model. For conductance based model, the effect of the input varies for different values of the membrane potential, and this effect needs to be converted into the form of current and multiplied by the membrane resistance R_{mem} to be added to the membrane voltage. At the same time, it could be inferred from the expression that the bigger membrane potential is, the more it may rely on the conductance g_e to have the same level of voltage change.

$$\tau \frac{dV}{dt} = (V_{rest} - V) + [g_e(V_{exc} - V) + g_i(V_{inh} - V)]R_{mem} \tag{6.1}$$

For the integrate or accumulating process, the key state variables, conductance g_e and g_i for the excitatory and inhibitory synapses respectively change with the input and have an exponential decay on their own. This could be modelled as in Equation 6.2 and 6.3. As part of the working mechanism, the conductance follows the exponential decay as well with time constants of τ_e and τ_i respectively. At the same time, it sums up the weighted value from spikes on all the afferent synapses in the given time period. Under such definitions, the function $\delta(t-t_k)$ in Equation 6.2 represents a spike event at time t_k, w_i is the weight for the ith afferent synapse in the range from 1 to N_e. Here N_e stands for the number of synapses the neurons has been connected with.

$$\tau_e \frac{dg_e}{dt} = -g_e + \sum_{i=1}^{N_e} \sum_k w_i \delta(t-t_k) \tag{6.2}$$

$$\tau_i \frac{dg_i}{dt} = -g_i + \sum_{i=1}^{N_i} \sum_k w_i \delta(t-t_k) \tag{6.3}$$

6.3.2 STDP mechanism and homeostasis

STDP mechanism

STDP is a type of Hebbian learning rule that operates based on the temporal information of the spike timing of pre and post-synaptic neurons. It will strengthen the connection between two neurons when the pre-synaptic spike travels before the post-synaptic spike. This is because under such condition, it could be inferred that the spike event of the pre-synaptic neuron contributed the spike event of the latter one. Consequently, such connection should be encouraged and vice versa. A typical STDP learning curve obtained from SpiNNaker could be observed in Figure 6.2 where t_j and t_i are the spike timings for pre and post-synaptic neurons respectively. The simulation is an approximation to the real case, which explains why the curve is not smooth. Meanwhile, it could be concluded from the curve that the closer those two spikes are temporally, the stronger the bond either as in potentiation or depression (Markram, Lübke, Frotscher, & Sakmann, 1997).

As the essential learning mechanism applied into the network, STDP actually has a lot of variations including pre-and-post STDP, power-law weight dependence plasticity. Following the original paper this architecture was proposed, triplet STDP was chosen for the average higher accuracy and more stable performance (Diehl & Cook, 2015). Triplet STDP is capable of handling cases where multiple spikes are involved. For example, pair-based STDP may lead to undermined long term potentiation (LTP) or long term depression (LTD) due to the high frequency of repetition of a spike pattern. This could be illustrated in Figure 6.3. In cases like in the figure where it is obvious that the two neurons' synapse should be potentiated, i.e. the weight of the synapse should be enhanced since the pre-synaptic spikes happen

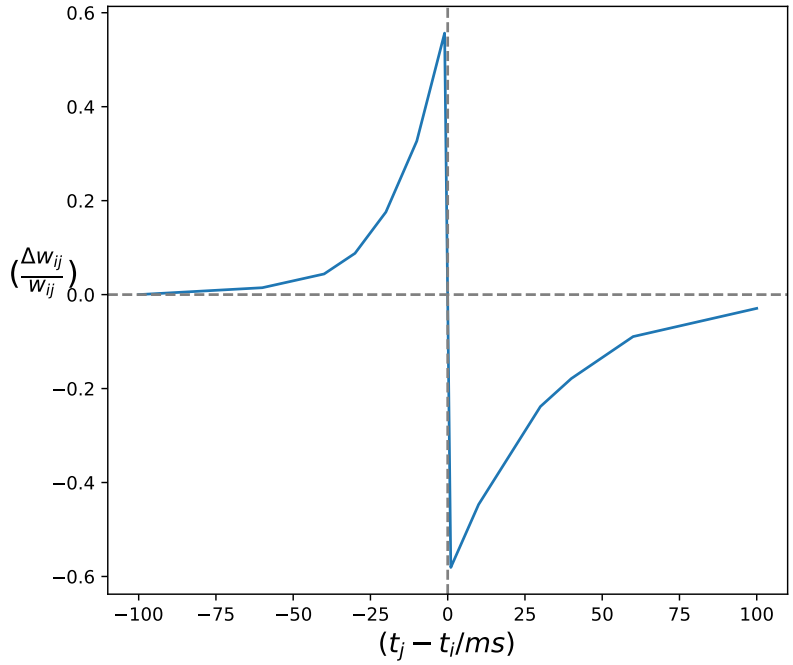


Figure 6.2: An STDP learning curve example

before their corresponding post-synaptic spikes. However, if this is described by the pair-based STDP where weight is updated on both pre-synaptic spikes and post-synaptic spikes, this potentiation (demonstrated in red curved arrow) would be partially countered by the potential depression (represented by the blue curved arrow) caused by the unwanted pair of the first post-synaptic spike and the second pre-synaptic spike. Additionally, this effect is more substantial when the frequency of one side spike increases. In other words, the smaller T is , the less negligible this effect will be. This is because, smaller T gives less space for Δt_1, which boosts the LTD according to the classic STDP curve. Triplet STDP has been proposed to correctly model this behaviour (Pfister & Gerstner, 2006).

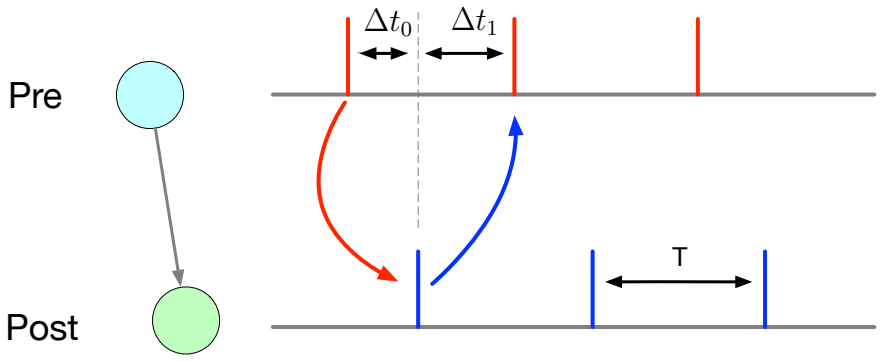


Figure 6.3: Undermined LTP situations

It was suggested by Pfister and Gerstner that the weight update should be considered among three or four spikes that are near to each other. And the effect the next near spike has should also be measured in an eligibility trace. As presented in Figure 6.4, LTD and LTP triplet combinations are illustrated as pre-post-pre case and post-pre-post case.

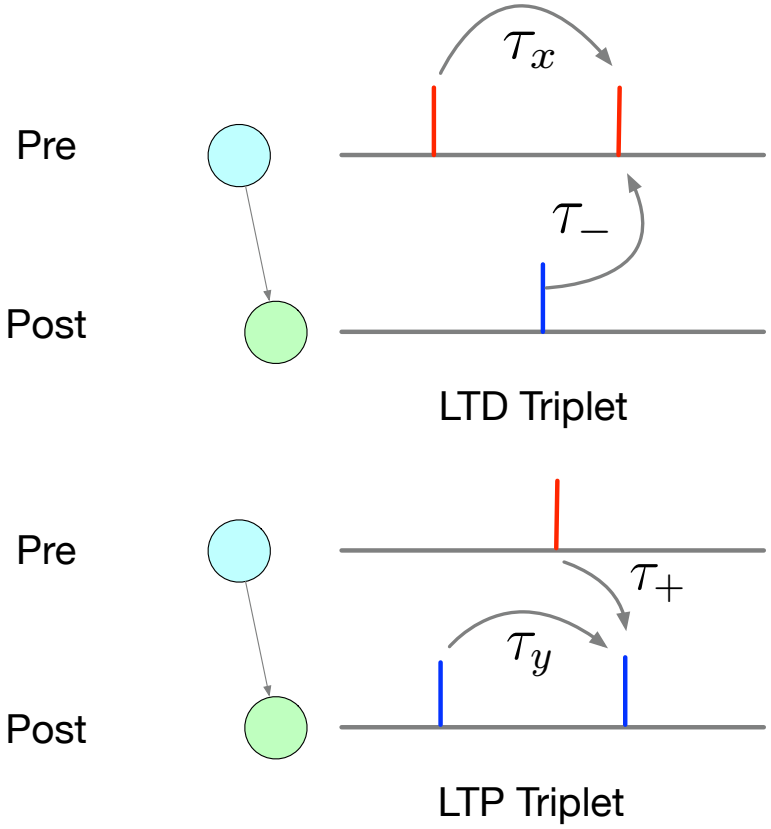


Figure 6.4: LTP and LTD triplets

For each spike event, the timing trace is expressed as follows:

$$\frac{dr_1}{dt} = -\frac{r_1}{\tau_+} \tag{6.4}$$

$$\frac{dr_2}{dt} = -\frac{r_2}{\tau_x} \tag{6.5}$$

if a pre-synaptic spike happens, $r_1 \leftarrow 1, r_2 \leftarrow 1$.

$$\frac{do_1}{dt} = -\frac{o_1}{\tau_-} \tag{6.6}$$

$$\frac{do_2}{dt} = -\frac{o_2}{\tau y} \tag{6.7}$$

if a post-synaptic spike happens, $o_1 \leftarrow 1, o_2 \leftarrow 1$.

It could be seen that for each spike, there will be two timing trace with different decay constants. For example, for the trace r_1 and r_2, the constants are τ_+ and τ_x respectively. One records the eligibility trace for its effect on this neuron when it emits a spike, the other one records its effect on the next pre-synaptic spike.

As the weight updating rule, it could be described as follows:
if a pre-synaptic spike happens at time t:

$$\Delta w = -o_1(t)[A_2^- + A_3^- r_2(t-\varepsilon)] \tag{6.8}$$

if a post-synaptic spike happens at time t:

$$\Delta w = +r_1(t)[A_2^+ + A_3^+ o_2(t-\varepsilon)] \tag{6.9}$$

Since every time a pre-synaptic spike happens, trace r_2 will be reset back to 1. A small difference ε is used here to extract the trace value at last time step. Same applies for the case when post-synaptic spike is observed.

Homeostasis mechanism

Another topic needs mentioning is the homeostasis mechanism applied in the architecture during training. Two different mechanisms were introduced into the system: an adaptive threshold and weight normalisation.

Since there is lateral inhibition implemented in the excitatory layer, where competition between each neuron is introduced at the training process. there might be the scenario when one neuron takes constant lead in every pattern and depresses all its counterparts at the same layer. This could lead to the imbalanced over-excitation, which consequently causes patterns of different classes to overlap on one neuron's synapses. This is highly unwanted during training. To tackle this, the homeostatic mechanism of an adaptive threshold is implemented for every neuron in the excitatory layer. In the normal case, the threshold of a neuron is static. Whenever the membrane potential reaches this level, a spike will be generated. However, a neuron with an adaptive threshold will have its threshold increased for a certain amount θ_m with every spike it emits.

$$\theta = \theta + \theta_m \tag{6.10}$$

$$\frac{d\theta}{dt} = -\frac{\theta}{\tau_\theta} \tag{6.11}$$

This approach effectively discourages the leading neuron from dominating at every sample, which could be seen as a regularisation mechanism. It works by increasing the threshold of the neuron so that for each spike it makes, the more difficult it will get for the next spike. Therefore, more opportunities will be spared to the rest of the population to learn different patterns. Therefore only the highly resembling input pattern that is salient with the already trained weight of the synapses will boost the neuron to fire again. However, the threshold will not indefinitely increase. The decaying of this variable helps prevent when one neuron's threshold get too high to a point not even one single spike could be generated.

Homogeneity is a valuable trait to have for the excitatory layer, which in other words means level of activity of each neuron in the excitatory layer remains almost the same on average. This has been achieved by weight normalisation of all the synapses that connect to one neuron. The weight normalisation operation is performed at the beginning of every sample's training. Furthermore, such implementation enables the neuron to be more resilient to noise in the sample.

$$w_{ij} = \frac{w_{sum}}{\sum_i w_{ij}} w_{ij} \tag{6.12}$$

6.3.3 Input encoding and data pre-processing

The original dataset is a set with 9600 training samples of simple rebinned energy histogram with 100 channels. However, unlike hand written digits, the sample itself was not highly distinct from each other, which may cause classification degradation for this methodology. As presented in Figure 6.5, a typical histogram for each class was plotted, some of which are fairly centralised on one or few channels. This could deteriorate the classification efficacy, since rate-based STDP underperforms at taking diagnostic information in a pattern (Vigneron & Martinet, 2020). Instead, it will try to memorise the whole typical pattern in the synapses. The range of the data distributed on each channel varies greatly from class to class. At the same time, the way each sample is represented does not have the overall presentation from the point of view of the whole dataset.

To address this issue, the Gaussian group encoding was used to bring the homogeneity and a more balanced range to the training set. Gaussian group encoding is a commonly used encoding in SNN. It presents one value in a higher dimension with each coordinate represents where this value lies in the whole range of the channel that this value belongs to. For example, an input vector, originally N-dimensional will be transformed into a vector of a higher dimension $N * M$ as in Equation 6.13. Each component x^i in the vector is expanded into M components $g(x^i, \mu_0), g(x^i, \mu_{M-1})$. In this work, $N = 100, M = 4$, it means the original sample with 100 channels were upscaled to 400 channels using this representation.

$$\{g(x^0, \mu_0), ...g(x^0, \mu_{M-1}), .., g(x^{N-1}, \mu_0), g(x^{N-1}, \mu_{M-1})\} \tag{6.13}$$

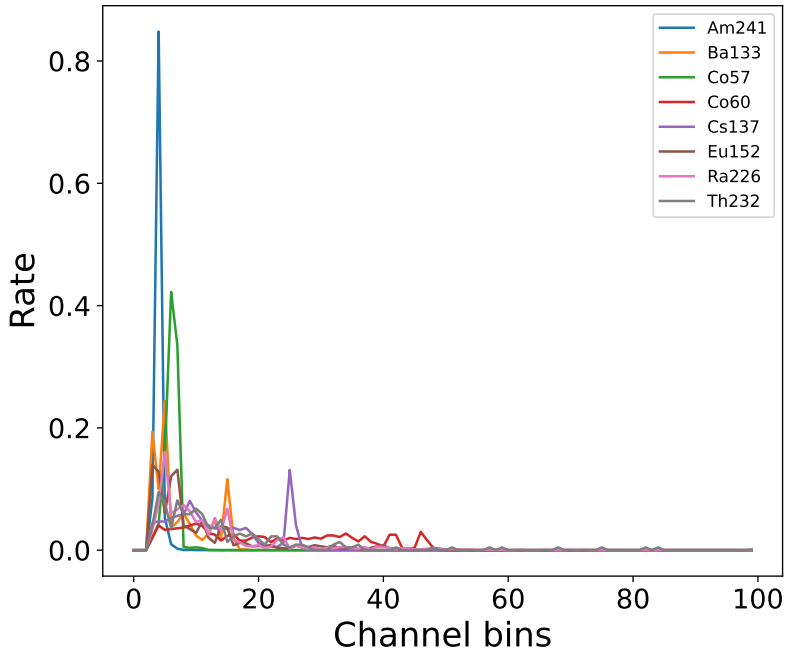


Figure 6.5: Typical histogram for each class

where

$$g(x^i, \mu_j) = exp(-\frac{(x^i - \mu_j)^2}{2\sigma_j^2}) \tag{6.14}$$

As suggested in (Sboev, Serenko, Rybka, & Vlasov, 2020), parameters should be calculated as:

$$\mu_j = X^i_{min} + (X^i_{max} - X^i_{min})\frac{j}{M-1} \tag{6.15}$$

and

$$\sigma_j = (X^i_{max} - X^i_{min})\frac{2}{3(M-2)} \tag{6.16}$$

After this encoding, the 400 input channels were filtered with a difference of Gaussian (DoG) filter together with thresholding to keep the input intensity within [0,128] Hz. By doing this, more effective and essential features get to be retained while the redundant and insignificant ones in the sample will be left out. This filtering does not change the input size. A rate-based coding scheme was used with input spikes being generated at an average frequency proportional to the rate given.

6.3.4 Simulation on SpiNNaker and training approach

Simulation on SpiNNaker

SpiNNaker is an emulation platform for SNN developed at the University of Manchester that employs ARM968 cores on the chip (Furber, Galluppi, Temple, & Plana, 2014). The simulation in this work is completed by a 4-node SpiNN-3 board, which is able to simulate up to 10,000 neurons. For this application this size of system is more than sufficient.

As for the dataset we used, it is the simple re-binned dataset discussed in Chapter 4. There are 9600 samples in total, and they were shuffled and evenly divided into 5 batches with each containing 1920 samples. Each batch will be delivered evenly among 8 duplicate unsupervised networks. Therefore, during an epoch, there will be 5 batches of training, and each duplicate network will be fed with 1200 samples. As for the verification, we took 30% of the total sample for testing.

During the training, adaptive threshold neurons will be used. Each sample will be presented to the network for 350 ms, followed by a 150 ms blank window where no stimulus is given to the network. This relaxes neurons back to the resting potential so that when the next sample arrives every neuron will start accumulating from the same level of membrane potential. The network architecture has been described in Python scripts using the PyNN API from SpiNNaker, using the sPyNNaker tool chain (Rhodes et al., 2018). The low-level neuron model, synapse model and triplet STDP are described in C for a faster compiling and running on the machine. Key scripts and code could be accessed through the github link[1].

The key parameters used for training this unsupervised network include: membrane decay time constant $\tau = 100\,ms$, membrane resistance $R_{mem} = 1\,M\Omega$, the neuron threshold voltage $\theta = -52\,mV$, reversal voltage for excitation $V_{exc} = 0\,V$, for inhibition $V_{inh} = -100\,mV$, resting voltage $V_{rest} = -65\,mV$, conductance delay time constant $\tau_e = 1\,ms$, $\tau_i = 2\,ms$ and time constant for threshold decay $\tau_\theta = 4 \times 10^5\,ms$

Minibatch processing

With all the architecture of SNN built up, one problem arises, and it is the training speed. Fairly speaking, time in simulation and actual simulation time differs due to the fact that there might be other factors taking time beside actual simulation. Statistically, with each sample's actual simulation time lasting for 500 ms, including the network configuration, loading and routing, it takes approximately 4 seconds for each round. As a result, the training process for all 9600 samples in one run will take longer to finish if these are done sequentially. The solution to this is a new technique called minibatch processing in SNN, which was originally proposed

1. The github link to the project could be found through the github repository: `https://github.com/sx4n18/EBCCSP_2022_unsupervised_net`

by Saunders (Saunders, Sigrist, Chaney, Kozma, & Siegelmann, 2020). He proposed that training of SNNs could be carried out similarly as in ANN, where weight convergence update could be aggregated by few batches of independent training results. Thanks to the parallelism available with SpiNNaker, multiple networks running independently could be achieved easily.

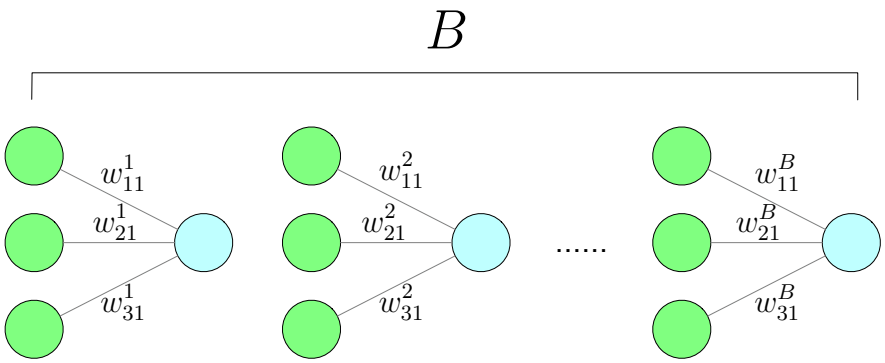


Figure 6.6: Minibatch processing

For example, as presented in Figure 6.6, the same network were duplicated multiple times. These B individual networks have the same initial parameters including weight and neuron state variables like membrane potential, threshold value. At each round of training, B different samples will be fed into these B networks. After training, the weight parameter could be aggregated in a parameter reduction method such as minimum, average or maximum. Here in this work, averaging was used, which is described in Equation 6.17.

$$w_{ij} = \frac{\sum_{m=1}^{B} w_{ij}^m}{B} \tag{6.17}$$

After aggregation, this parameter is applied to every network in the next round of training as the starting parameter. As simple as this technique may seem, it improved the training speed significantly. At the same time, the training accuracy was not significantly affected. By the testing of the handwritten digits classification task in (Diehl & Cook, 2015), the accuracy difference with minibatch size of 8 saw a drop by less than 1%.

Post-processing filtering

Training of this unsupervised network has been carried out in a straightforward fashion. First, 8 samples will be encoded in Poisson spike train and fed into the network for 500 ms as one round. After this, the script will extract the spike count of each neuron as a record and also essential state variables (mainly the membrane potential at the last time step and threshold value). With the parameter reduction method used, 8 sets of different parameters will merge into one and be applied to the next round of training as the starting point. However, during training, there might be unwanted spike behaviour at the excitatory layer which has a deleterious effect on the parameter reduction and update. Even with lateral inhibition, there

still are cases where no clear winner could be found, or even there is a winner, it is not leading the competition by much. That is to say, what could observed in the excitatory layer is that almost all excitatory neurons have the same number of spikes during training. As a result, every neuron will pick up the input pattern and try to memorise it in the synapses. Clearly, this homogeneity is detrimental and not wanted in the training process. Therefore, a post-processing filtering method was implemented during the parameter reduction

The filtering method works as a remedy during the training process to pick out the data points that the network does not respond well and leave them out of the weight update process. Since the ideal behaviour of the excitatory layer should be one or two clear winner neurons taking the lead (many more spikes than the rest), what is expected is that the statistical description of the spike counts in the layer should hold a relatively low average value and a large standard deviation. These can be easily implemented using Numpy.

6.4 Results

6.4.1 Convergence and accuracy

There have been multiple tests about the accuracy concerning the size of the excitatory layer. For the accuracy evaluation, this was done at every stage of the dataset during training. The whole dataset was divided into 5 small stages with each holding 1920 samples. As the training goes, there could be one set of assignment concluded from the last 1920 samples. Based on that, a real time accuracy at each stage could be calculated as presented in Figure 6.7. Details about assignment and inference will be discussed in the next subsection.

The experiments that were carried out in the figure are running on the same parameters except the excitatory size. It could be noted that except number 9 for parameter n_e, all the rest runs could reach around 70% of accuracy at the end of first epoch of training. This is because when n_e is 9, it is just slightly bigger than the number of classes of the dataset. Consequently, the assignment of the neurons in this layer might not make a complete class set. That is to say, there is at least one class that could not be inferred at all from this network. On the other hand, when the size of this layer keeps increasing, a more gentle yet less steep accuracy curve could be observed. This is because the assignment process during training happens at a similar rate with different size of excitatory layer. The bigger this layer is, the more training it will need to finish the complete assignment of every neuron in it. However, unlike the conclusion made in the paper (Diehl & Cook, 2015) where the wider the excitatory layer is (more neurons in the layer), the higher accuracy, this work has different results. In this work, the widest excitatory layer does not give the highest accuracy after a small comparison.

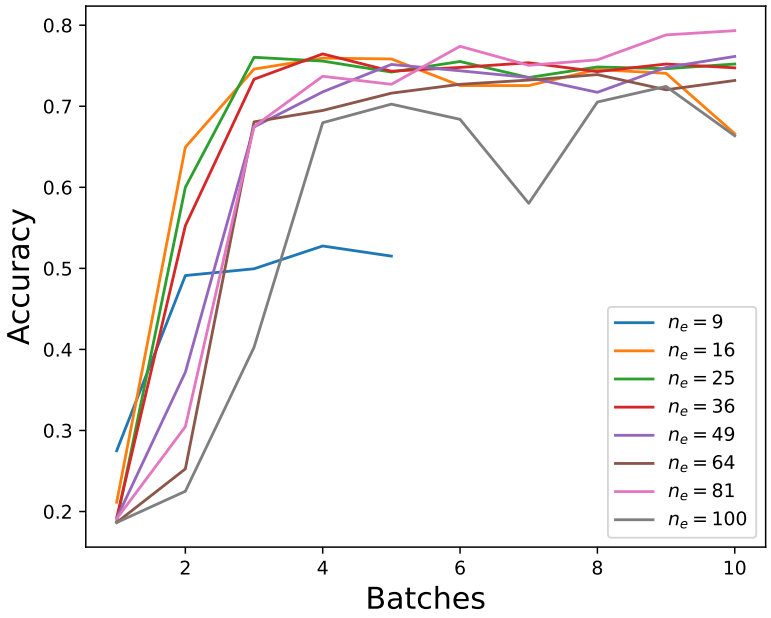


Figure 6.7: Accuracy and the excitatory size

It could be noticed from Figure 6.7 that it is not entirely true that the wider the better in this application. The experiment on the size shows that under the same configuration, as n_e grows, the average accuracy will increase first and then slowly decline. For cases when n_e values are 16, 25, 36, 49 and 64, the average accuracy does not vary much with the difference of only 1% at most. However, for $n_e = 81$, cases have been exceptional with an average accuracy of 77% and the highest accuracy of 80%. However, contrary to the conclusion from (Diehl & Cook, 2015), the biggest excitatory size in the test run led to one of the lowest performances of 67.7%.

The reason why the biggest size $n_e = 100$ has one of the lowest training accuracy may be resulted by a combination of reasons. First, it may still take some extra training to reach the convergence. So far, we have seen that the convergence speed varies depending on the size of the net and it generally takes longer for a bigger network to converge than a smaller one. This would mean that the accuracy obtained from the size of 100 is not the final accuracy. Second, we observed strong homogeneity while training size 100. This is likely because of the quality of the training set. And we noticed that the assignment of a single neuron during training keeps changing at different batches. This is because the difference between classes are not prominent enough to be captured. And this will leave the network in a negative

behaviour mode where it is either "everyone is a winner" or "no one is a winner" during the winner-takes-all mechanism. And this poses a bigger impact on a wider network than a narrower one. This also explains why the accuracy curve for 100 is more rigged, because assignment for neurons cannot be stabilised.

Following what was mentioned above, some other tests were carried out. A random search for the threshold delay time constant τ_θ (this defines how "forgetful" excitatory neurons are) was done. Among all combinations, the set when $n_e = 49$ and $\tau_\theta = 7 \times 10^5$ together with $n_e = 81$ and $\tau_\theta = 4 \times 10^5$ stood out and made the highest average training accuracy around 77%.

To verify the effectiveness of implemented post-processing filtering, a comparison test was performed. Given the same parameters of $n_e = 49$ and $\tau_\theta = 7 \times 10^5$, two SNNs were trained. The only difference is the presence of filtering. It could be observed from Fig 6.8 that the filtering mechanism offers stability for training and faster convergence speed.

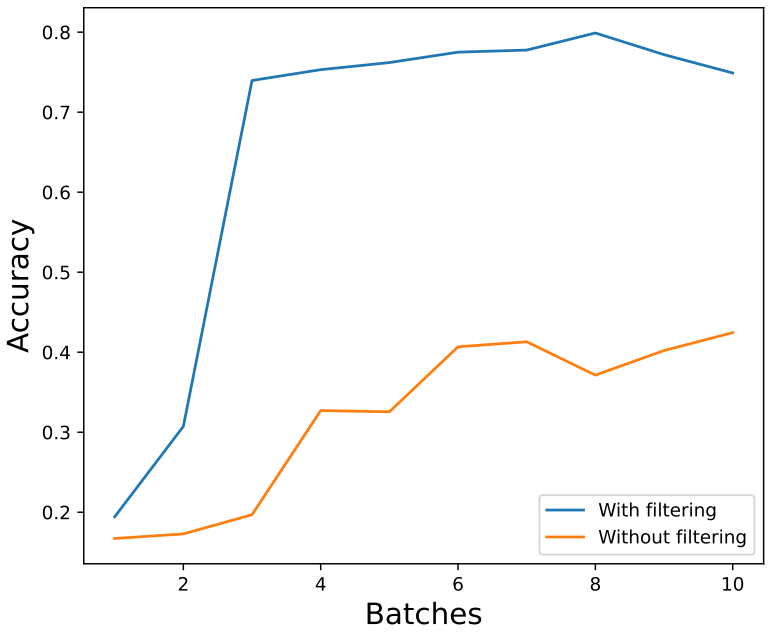


Figure 6.8: Impact of the post-processing filtering

6.4.2 Inference and assignment

The way inference works for this network is that after each epoch of training, there will be an assignment given to each of the excitatory neuron according to their statistical behaviour throughout the past training samples. On average, the neurons that give out the most spikes for a specific class will be labelled as that class. This assignment, however, is updated at each epoch. For the final testing, the whole set of trained parameters like weight, threshold values and neuron assignments from the configuration of $n_e = 81$ and $\tau_\theta = 4 \times 10^5$ is chosen. The assignment map for this network by the end of training is shown in Figure 6.9, where each class has been encoded in a label number ranging from 0-7 and they are arranged as $^{241}\mathrm{Am}$, $^{133}\mathrm{Ba}$, $^{57}\mathrm{Co}$, $^{60}\mathrm{Co}$, $^{137}\mathrm{Cs}$, $^{152}\mathrm{Eu}$, $^{226}\mathrm{Ra}$, and $^{232}\mathrm{Th}$ respectively. It could be concluded from the map that the training of this network has been entirely random due to the unsupervised methodology and homogeneous lateral inhibition among the excitatory layer.

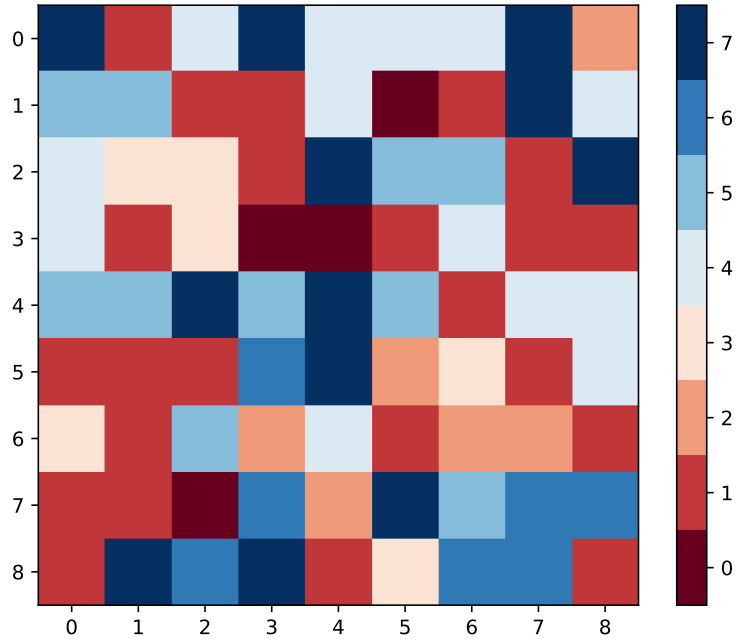


Figure 6.9: Assignment map

At the inference stage, the network is reconstructed with the trained weight frozen and threshold values initialised for each neuron. However, the neuron model will be replaced with normal conductance based LIF neuron instead of the one used during training. What's more, the threshold values used will be 30% less. This is to activate neurons to give a more contrasting spike count distribution and encouraging a clearer result. After each testing sample fed into the network, the recorded spike count for every neuron is evaluated. The neuron that makes

the most spikes is recognised as the winner and its corresponding assignment is presented as the final result for this inference. In cases where there are more than one winners, a voting mechanism implemented will list out the label that was reported most as the winner. If there are still multiple winners, the label with smallest index will be considered as the winner.

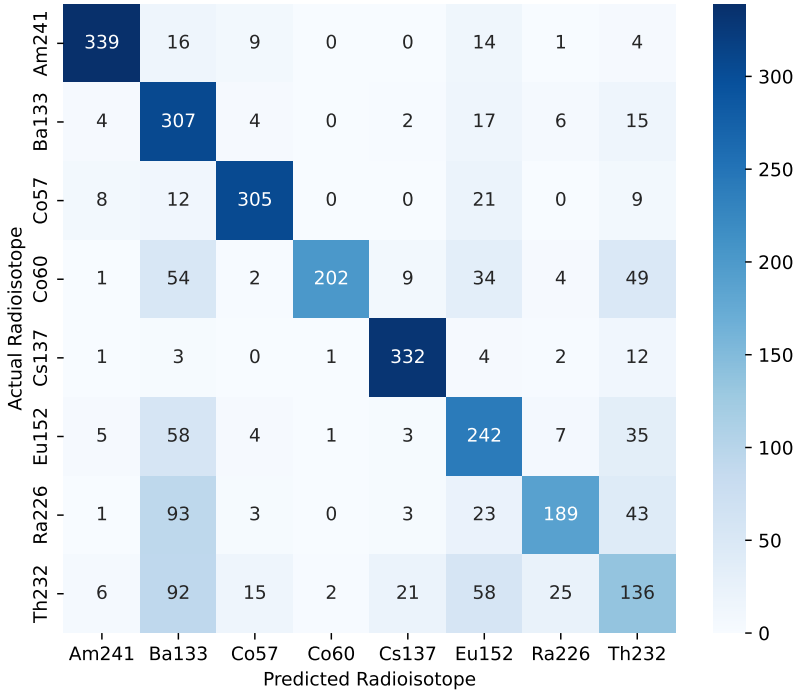


Figure 6.10: Confusion matrix

The whole dataset was tested, which contains 2868 samples. The testing results were organised and presented as in the confusion matrix. It could be seen that this model is doing fairly well in most cases. But in the cases for few troublesome isotopes including ^{60}Co, ^{226}Ra and ^{232}Th, the accuracy on those isotopes has been below 60% with the worst being even below 40%. The reason for this might be the way the data has been pre-processed. The group encoding gives a global point of view to each sample from the whole dataset. Moreover, pattern for ^{232}Th has been quite random when the distance gets further. This could explain the confusion of ^{232}Th with other isotopes. On the other hand, the similarity in the histogram of ^{133}Ba and ^{226}Ra suggested in (X. Huang, Jones, Zhang, Xie, et al., 2020) could also explain the low accuracy on ^{226}Ra.

6.5 Conclusion

In this chapter, an unsupervised STDP based SNN application on the RIID has been implemented on SpiNNaker. This involves the detailed parameter selection, data pre-processing and multiple mechanisms including neural homeostasis, filtering and minibatch processing. Unlike other SNNs, this network has been trained with the training approach that is entirely native to SNN. This is the first unsupervised STDP based SNN implementation for RIID by our knowledge. Using this network, the classification on 8 different radioisotopes could be achieved with accuracy of 72%. This could be potentially increased with other techniques such as dynamic range compression, logarithmic data pre-processing or more advanced SNN architecture (e.g. convolutional SNN) and training approach. Future work may include more dataset, other types of STDP (e.g. neuromodulated STDP (Mozafari, Kheradpisheh, Masquelier, Nowzari-Dalini, & Ganjtabesh, 2018)) or deeper networks.

However, when compared to the approach adopted in the work (X. Huang, Jones, Zhang, Xie, et al., 2020) where a shallow ANN was trained first and then converted to SNN, the unsupervised STDP approach has the following drawbacks:

- Accuracy is discernibly lower than ANN conversion
- Training costs much more resources and longer even with less overhead
- The network architecture is not suitable for hardware implementation

First, when it is compared with other implementation sharing the same dataset, this approach can only achieve 72% of accuracy. And this is with all the preprocessing added to the data before the unsupervised net. However, the testing accuracy from (X. Huang, Jones, Zhang, Xie, et al., 2020) claims almost 90 %. Second, even with the speed up training technique applied, the training for this unsupervised network still costs more than 10 hours with the 4-node SpiNNaker board. Last but not least, to deploy this implementation, neuromorphic hardware might inevitably be the only reasonable choice. This is because the model is not easy to implement on FPGA or other hardware with the conductance based neuron and highly recurrent model architecture.

Chapter 7

Bin-ratio ensemble SNN for RIID

From the last chapter, it can be concluded that biologically plausible training so far is not an efficient choice for deep learning tasks when using SNNs. However, the drawback of shadow training still exists with the long simulation time needed to reach a satisfying accuracy.

In this chapter, an updated conversion method shall be discussed and explained in detail. This training will be employed to train an ensemble of 20 simple 3-layer SNNs for a more challenging but more realistic dataset on the application for RIID. This algorithm aims to deliver high accuracy in identifying 18 different radioisotopes with the assistance of mild preprocessing identified as "Bin-Ratio".

7.1 Introduction

Just as concluded in chapter 1, it can be learnt from the history that potential harm from radioactive isotopes like $^{137}\mathrm{Cs}$ or $^{131}\mathrm{I}$ is substantial. Moreover, the threat of scenarios like dirty bombs, and smuggling of radioactive materials can be mitigated with proper actions. Detection of hazardous isotopes on a handheld device and quick strategic countermeasures play a fundamental role in national security, especially in populated areas such as airports and stadiums. For example, dirty bombs normally consist of explosives and radiological materials like $^{137}\mathrm{Cs}$. A portable device that is capable of detecting key radioisotopes could quickly identify the presence of dangerous materials and prevent dirty bomb attacks or illicit trafficking. This has been well established in chapter 4 with the real application device form Kromek. However, such handheld devices are usually constrained by the battery. Consequently, power-efficient solutions for radioisotope identification (RIID) are strongly needed to extend the battery life.

Thanks to the significant progress in the machine learning infrastructures, new solutions to problems like medical diagnosis (Ozkan, Koklu, & Sert, 2018) (Abubakar & Olatunji, 2020), warehouse automation (D. Zhang, Pee, & Cui, 2021) and autonomous vehicle navigation (Shreyas, Bharadwaj, Srinidhi, Ankith, & Rajendra, 2020) have been proposed. Similar cases also apply to RIID.

Over the years, RIID algorithms evolved gradually in terms of the level of autonomy just as reviewed in chapter 3. The most manual way to decipher the existence of a certain radioisotope is to rely on an expert's interpretation of the spectrum (Knoll, n.d.). As the algorithm progresses, more systematic approaches like region of interest (ROI) matching (P. A. Aarnio et al., n.d.) and template matching (Yan & Glaser, 2015) were introduced. Inevitably, machine learning algorithms were also applied in this field for their high level of automation and robustness. Successful applications like artificial neural networks (ANNs) (Kamuda et al., 2017), and convolutional neural networks (CNNs) (Kamuda, Zhao, & Huff, 2020) have proven the efficacy of this ideology. To push this further, spiking neural networks (SNNs), the promising next-generation of neural networks, have also been applied in RIID (X. Huang, Jones, Zhang, Furber, et al., 2020) (X. Huang et al., 2021) (Xie et al., 2022). This shift could build an end-to-end event-driven signal processing system, which offers low power consumption (Kim, Park, Na, & Yoon, 2020) and fast inference (Bu et al., 2023). Such a solution aligns perfectly with the requirement of the portable detector: a power constraint RIID application with the need for a robust and automated algorithm.

This chapter will introduce a new ensemble SNN implementation inspired by the ensemble ANNs with a bin-ratio pre-processing technique. In the later sections, the original bin-ratio ensemble ANN will be discussed, and then the conversion pipeline and hardware-oriented optimisation techniques will be illustrated in detail.

7.2 Related work

7.2.1 Bin-ratio ensemble ANN

A scintillator is a type of sensor that emits photons when receiving ionising radiation. These photons are then translated into a spectrum that spreads over different energy levels. RIID algorithms mainly process these spectra and give the analysis of the constitution of possible radioisotopes. However, noise is a big issue when carrying out RIID tasks. There are mainly 2 different types of noises: background noise and gain shift.

Bin-ratio ensemble ANN (BEANN) was proposed (S. Zhang, Marsden, & Goulermas, 2022) as a resilient machine learning algorithm to tackle the noise issue. It processes the ratio of bin values instead as the input and takes the averaged judgement from an ensemble of ANNs. In detail, the implementation itself is really simple. The key for this algorithm is the pre-processing where bin-ratio values were computed instead of directly using the original values from the spectra.

Normally the values across a spectrum can be considered as a 1-D vector, and the values in this vector are non-negative integers. BEANN proposes a division matrix where each column and row corresponds to the bin value from the original spectrum. This matrix can be illustrated

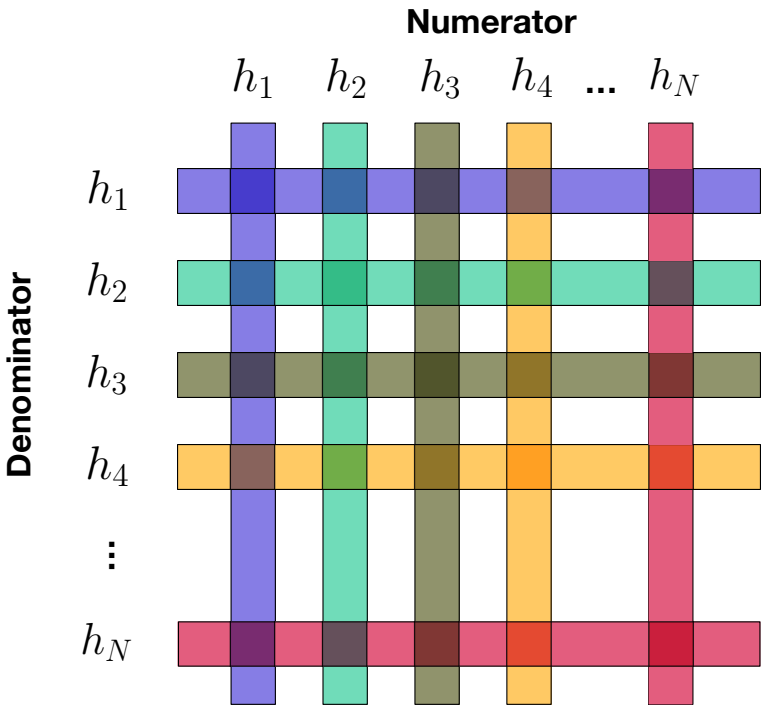


Figure 7.1: Construction of the bin-ratio matrix

as in Figure 7.1, where values of bins are represented in different colours. Imagine a spectrum with N different bins, there should be N elements in this 1-D vector and can be represented as h_i. By having this 1-D vector in two different dimension, a bin-ratio matrix is constructed with each row representing denominator and each column representing numerator. Naturally, the $N \times N$ matrix should look as the representation in Figure 7.2:

$$\begin{bmatrix} \frac{h_1}{h_1} & \frac{h_2}{h_1} & \frac{h_3}{h_1} & \frac{h_4}{h_1} & \cdots & \frac{h_N}{h_1} \\ \frac{h_1}{h_2} & \frac{h_2}{h_2} & \frac{h_3}{h_2} & \frac{h_4}{h_2} & \cdots & \frac{h_N}{h_2} \\ & & \vdots & & & \\ \frac{h_1}{h_N} & \frac{h_2}{h_N} & \frac{h_3}{h_N} & \frac{h_4}{h_N} & \cdots & \frac{h_N}{h_N} \end{bmatrix}$$

Figure 7.2: Detailed maths representation of the bin-ratio matrix

In the data used in the original paper, there are 1024 elements in one spectrum, and 9000 samples in total. This would create 9000 1024×1024 matrices. It can be observed that elements of the central diagonal of this matrix are all 1s, which have no contribution to describing the characteristic of the spectrum. All the other diagonals describe the ratio of two bins with the distance of 1 to $N-1$. In other words, they reflect the comparative relationship between bins within a spectrum. Therefore, each diagonal contains information that correlates to a specific isotope from one perspective. Theoretically, $N-1$ ANN models can be trained to provide "opinions" about one single spectrum from different perspectives.

Inspired by this, the ensemble model of 20 ANNs that are trained on each diagonal's data was constructed. Each ANN model simply has 3 layers: input layer, hidden layer with 40 neurons and the output layer with 18 classes. Having experimented various machine learning algorithms, BEANN is by our knowledge the best solution that both excels in performance and simplicity of architecture. Having BEANN in its spiking version would smooth the deployment of such robust algorithm in the embedded platform.

7.2.2 Specific spiking neurons for mapping

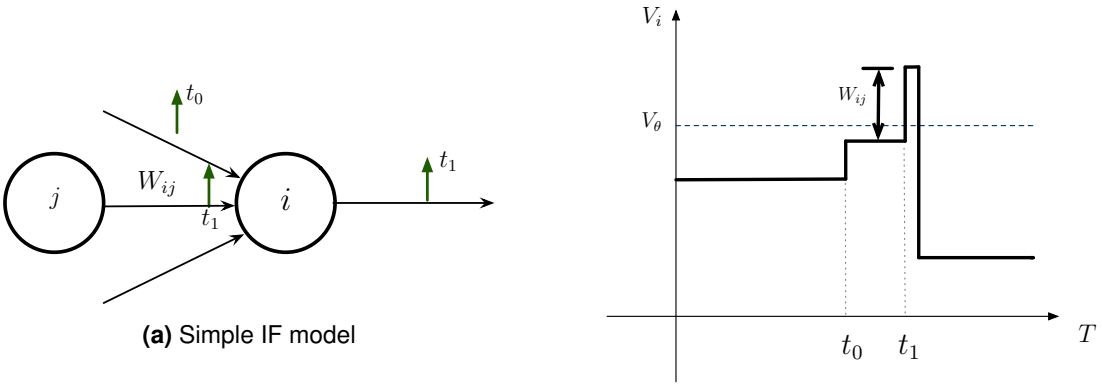
SNNs work similarly to ANNs but with more biologically plausible neurons. Take the most commonly used spiking neuron, integrate and fire (IF) neuron, as an example, the behaviour of the neuron could be basically described as follows:

$$\frac{dV_i^l}{dt} = \sum_{j=1}^{N} S_j^{l-1} W_{ij} \tag{7.1}$$

$$\begin{aligned} if\, V_i^l > V_\theta, V_i^l = V_i^l - V_\theta, S_i^l = 1; \\ else\, S_i^l = 0 \end{aligned} \tag{7.2}$$

This highly simplified neuron is used because of its high resemblance to an ANN neuron with ReLU activation function. The traditional mapping from ANN to SNN is intuitively straightforward and has been reviewed in chapter 5.

In Equation (7.1), the integration stage is described, V_i^l signifies the membrane voltage of neuron i in layer l, S_j^{l-1} symbolises the spike coming from neuron j in the last layer $l-1$, where 1 means a spike and 0 means no spike. W_{ij} denotes the weight on a particular synapse connection from neuron j to i. During processing, the neuron will pick up spikes coming from the last layer and accumulate at the membrane voltage according to the weight at each connection. On the other hand, fire and reset logic is formulated in Equation (7.2) where V_θ is the threshold voltage, once the membrane voltage surpasses this value a spike



(a) Simple IF model

(b) Voltage change along time

Figure 7.3: Integrate and fire model

shall be generated by setting S_i^l to 1 and reset by subtracting V_θ from V_i^l. Otherwise, V_i^l will stay unchanged and S_i^l stays 0 by default. This whole mechanism can be demonstrated in Figure 7.3. In the sub-figures, Figure 7.3a depicts the IF neuron connection and spike transmission while the corresponding voltage change for neuron i is plotted in Figure 7.3b.

7.2.3 Conversion noises and solutions

However, there can be some disagreement or errors between the behaviour of an IF neuron and an artificial neuron. These are mainly where the accuracy drop come from during conversion. The sources of the noises can mainly be summarised as in (Diehl et al., 2015):

- Input noise or encoding noise
- Sub-threshold noise
- Supra-threshold noise

Encoding noise

Encoding noises mainly contributes to the error due to the fact that input real values cannot be fully represented by discrete spikes. On the other hand, Poisson encoding, the most commonly used encoding scheme for input, is totally stochastic. This would inevitably introduce errors due to the randomness during input encoding. Poisson encoding spike generation is demonstrated in Figure 7.4.

This process simply takes a random number generated from a generator and compare it with a specified rate, is this number is smaller than this rate, a spike is generated, otherwise no spike is produced. When this random number generator gives random numbers from a uniform distribution from 0 to 1, then on average the generated spikes over time will be proportional to the rate. This can also be easily implemented in a programme following the Algorithm 3.

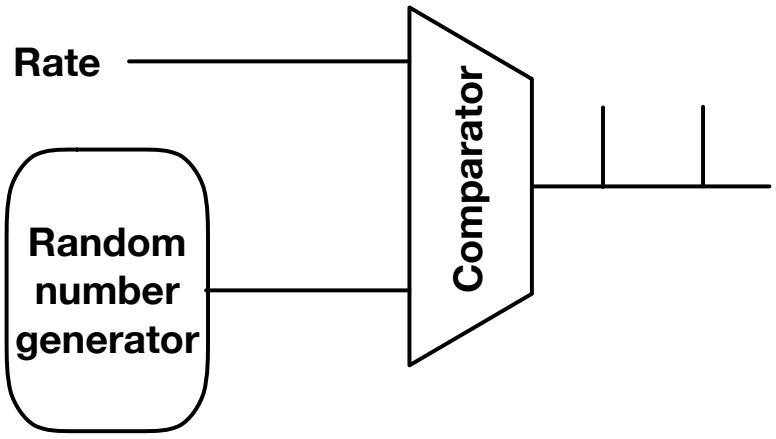


Figure 7.4: A simple Poisson encoding spike generator

Algorithm 3 Poisson encoded Spike Generation from Rate

```
function GENERATESPIKE(Rate)
    RN ← RandomNumberGenerator(0, 1)        ▷ Generate random number from uniform distribution
    if RN < Rate then
        Spike ← 1                           ▷ Generate spike if random number is less than rate
    else
        Spike ← 0                           ▷ No spike if random number is greater than or equal to rate
    end if
    return Spike
end function
```

Following this algorithm, errors can be observed from a simple Python simulation [1]. It can be clearly seen from the simulation in Figure 7.5 that a bar-code like spike train was generated with the given rate 0.23. The simulation time is 1000 ms and a single time step is 1 ms, therefore the max spike count should be 1000. However, the actual spike count is 253, this would give us an average rate of 0.253 instead of 0.23 as specified.

This can be addressed by using analogue input instead of Poisson encoding. The proposed technique simply merges the encoding process and the first hidden layer and treats the input real value as the weighted constant current injection. Even though this approach may sound less "spiky", the implementation is still biologically plausible.

1. Scripts can be found in the GitHub repository

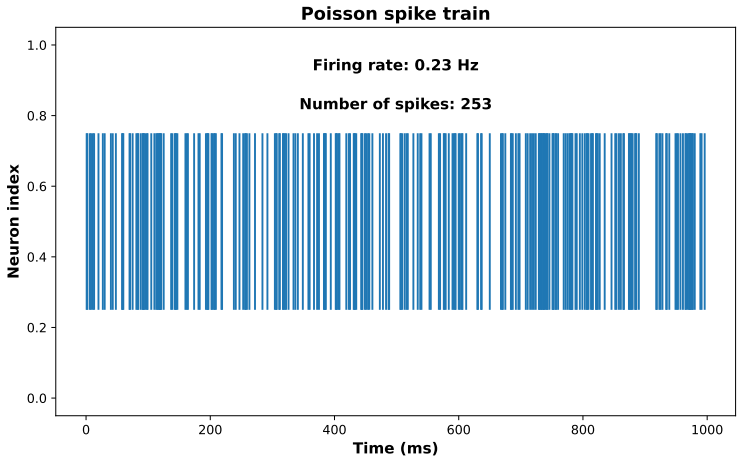


Figure 7.5: Poisson spike train generated from the Python script

Supra-threshold noise

The supra-threshold noise, just as the name suggests, is brought by the fact there is the scenario when accumulated membrane potential surpasses the threshold and the "unused" potential cannot be represented by the spike. This is an easier problem to solve and can be addressed by saving that "unused" bit into next time step's accumulation (Rueckauer et al., 2017). The "unused" potential shall then give the neuron a slight "headstart" at the beginning of the next time step. Just as indicated by Equation 7.2, the application of a "soft-reset" or reset-by-subtraction mechanism instead of reset-to-zero recovers the error caused by supra-threshold noise. This can be demonstrated by Figure 7.6, where the supra-threshold noise V_ε is compensated by reset-by-subtraction.

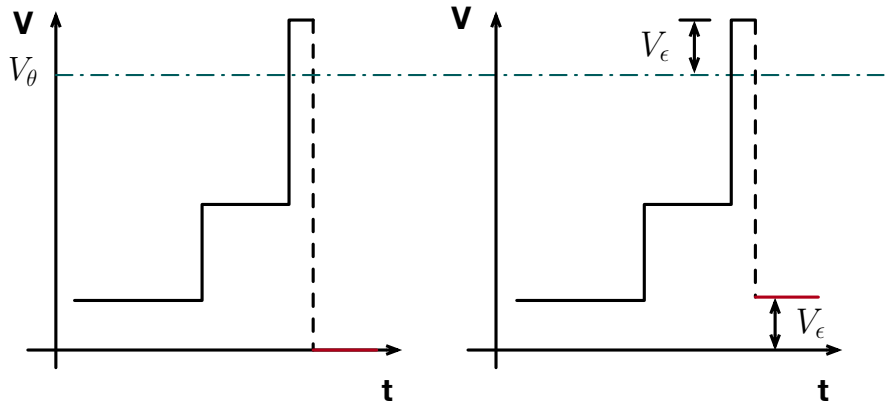


Figure 7.6: Hard reset (left) and soft reset (right)

Sub-threshold noise

Sub-threshold noise, on the other hand, is a more tricky problem. It mainly describes the error brought by the potential residual by the end of simulation. For example, Figure 7.7 shows 3 different cases of IF neurons' potential along with the simulation time. Apparently,

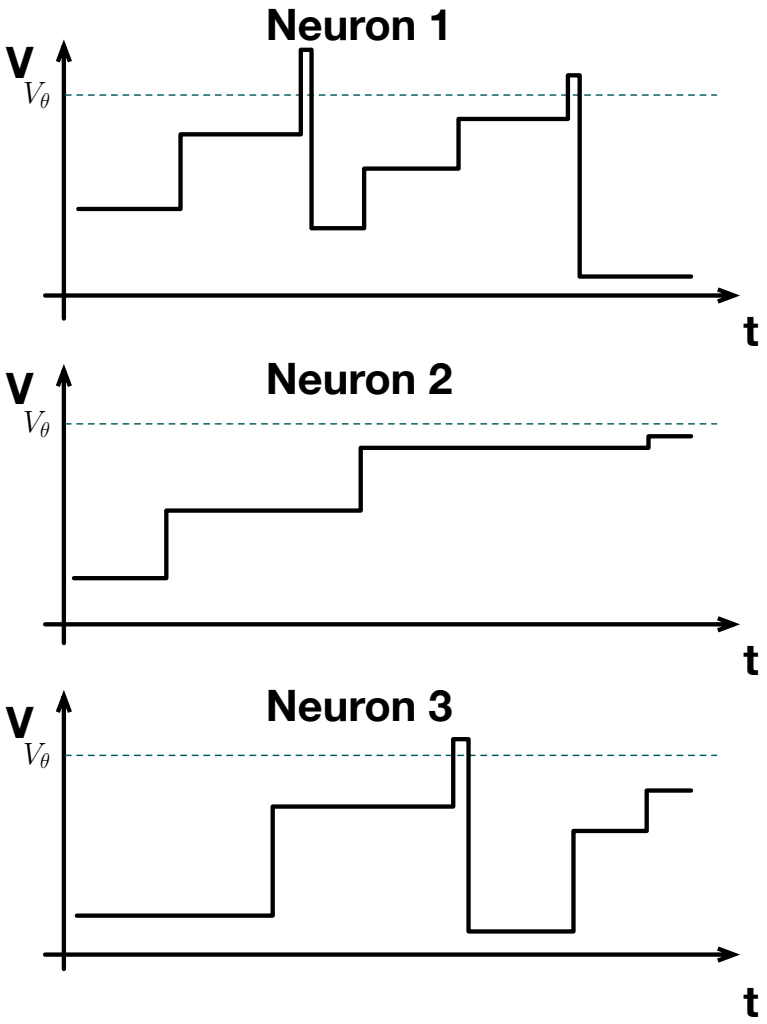
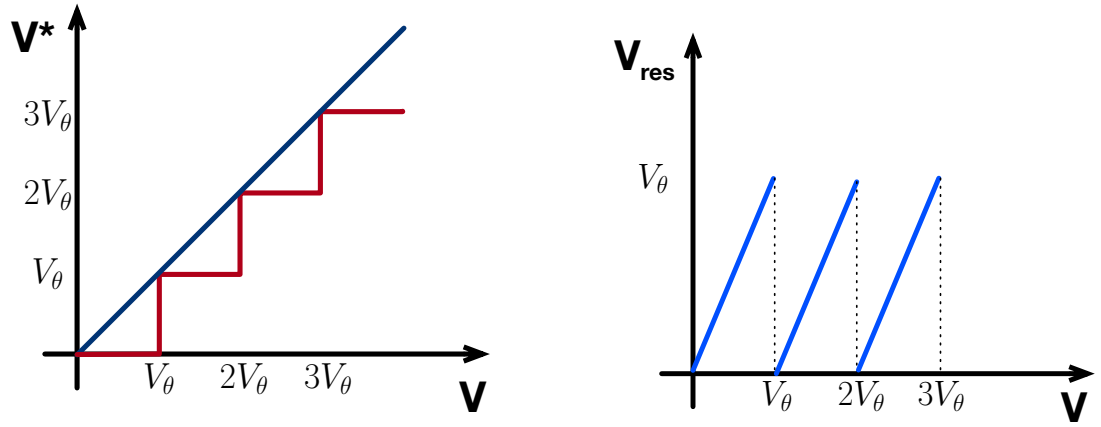


Figure 7.7: Different cases with neuron's potential by the end of simulation

the impact of the potential residual has towards neurons is different. Neuron 1 in the figure only has very little potential residual by the end of simulation. 2 Spikes were generated during simulation for neuron 1, if the final residual left is regarded as V_ξ. The total accumulative potential is $2V_\theta + V_\xi$, which is closer to the actual ANN neuron activation it should represent, but the actual activation represented by this neuron would simply be $2V_\theta$. The error here $V\xi$ is negligible in this case given that $V_\xi \ll V_\theta$. However, for the case of neuron 2, the final



(a) Actual response function (light cayenne) of the IF neuron and the ideal response function (smokey ocean)

(b) Sub-threshold noise along the accumulative potential

Figure 7.8: Mathematical analysis of the response function for IF neurons

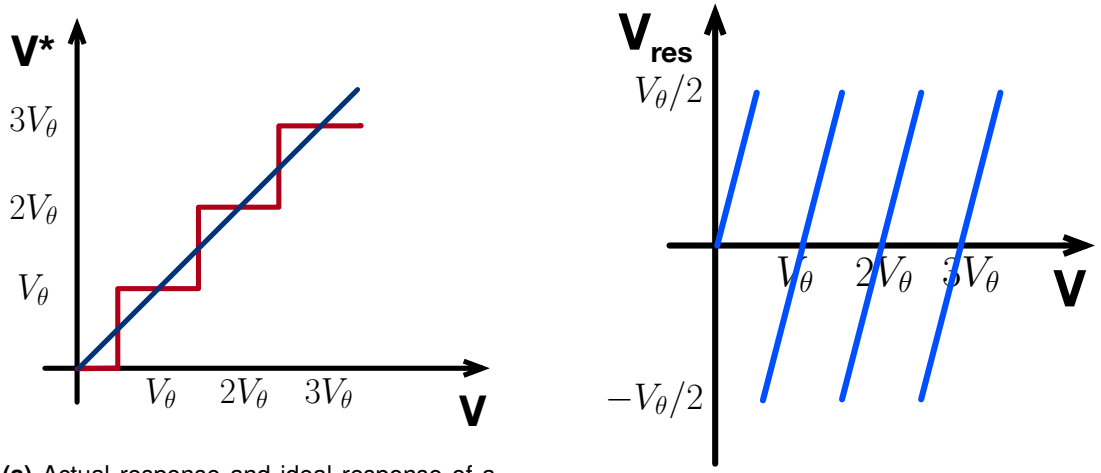
residual V_ξ, which is also the accumulative potential, is much higher and very close to V_θ. But the final representation of this IF neuron is 0, this is far from the correct activation it is trying to represent. Similar scenario can be observed in neuron 3 where the accumulative potential and the actual IF neuron representation are highly unbalanced, where $V_\theta + V_\xi > V_\theta$

The essence of this noise is due to the fact that IF neurons are operating flooring function. If the final spike count is N_s, the total accumulative potential is V_{acc}, the threshold is defined as V_θ, we can conclude the correlation as follow:

$$N_s = \lfloor \frac{V_{acc}}{V_\theta} \rfloor \tag{7.3}$$

Mathematically, the relationship between the accumulative potential and the actual represented potential by IF neurons can be depicted by Figure 7.8a. The light cayenne curve expresses the perceived activation represented by the IF neuron against the actual accumulative potential. It presents a step-shaped response function due to the flooring function. The curve in smokey ocean represents an ideal response function. By calculating the difference between these two functions, it is easy to get the sub-threshold noise as represented in Figure 7.8b.

This noise can be mitigated by pre-charging the IF neuron at the beginning of the simulation (Hwang et al., 2021). One may choose to pre-charge the membrane potential to the half of the threshold to have a balanced sub-threshold noise, by which a zero mean distribution is shown just as Figure 7.9 By having the sub-threshold noise changed as in the form in Figure 7.9b, one may assume the impact the positive residual can be compensated by the negative residual on an overall average.



(a) Actual response and ideal response of a pre-charged IF neuron

(b) Sub-threshold noise with pre-charged IF neuron

Figure 7.9: Mathematical analysis of the pre-charged IF neuron's response

7.2.4 Learned step size quantisation and ANN-SNN conversion

ANN quantisation is a technique that is widely used to compress the model size and compute with low precision. There are simple post-training quantisation (PTQ) and more elaborate quantisation aware training (QAT). PTQ simply runs through the whole training dataset or part of it to find the corresponding parameters to scale the activations and weights into 8 bit integers. This is essentially an arithmetic scaling where no more training will be done after the standard process.

However, QAT performs extra training with scaling factors inserted in the fake-quant node (Jacob et al., 2017). It first introduces a new set of parameters into the model with within fake-quant nodes, which is illustrated in Figure 7.10 It can be seen from the figure that a fake-quant

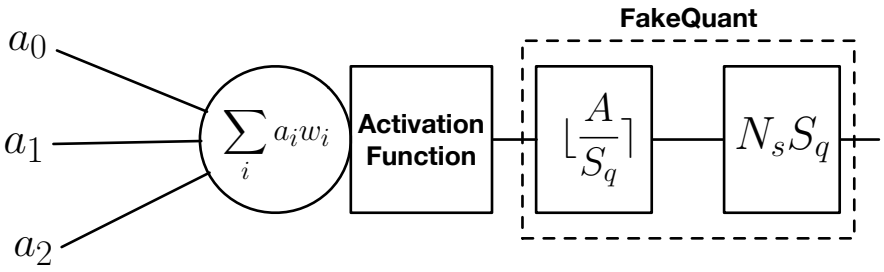


Figure 7.10: Fake-quant node in quantisation aware training

node is basically a combination of quantisation and dequantisation. Quantisation stage, which is simplified here with a scaling and rounding, while dequantisation aims to scale the quantised value back to the original activation in floating point number. For QAT, the forward pass will

include this fake-quant process to expose the quantisation error so that the backward pass can update the weight to compensate the error induced by the quantisation process. The general QAT will only update the floating point weights and does not update scaling parameters like S_q.

Learned step size quantisation (LSQ) uses similar approaches, the difference is that LSQ supports the update for the scaling factor S_q (Esser, McKinstry, Bablani, Appuswamy, & Modha, 2020). This would allow the user to restrict the quantisation range. In other words, one may train the network's weights and scaling factors so that activation values can be represented with even lower precision unlike the traditional INT8 representation. The author showed that even 3-bit integers are enough to reach the full precision baseline full-precision model accuracy.

Inspired by this LSQ quantisation scheme, a quantisation framework for fast SNN conversion was introduced (C. Li, Ma, & Furber, 2022). Since the essential idea of conversion is to map activation values to spike rates, the author showed that the bit-width needed to represent all necessary activation values is the key factor that determines the SNN inference speed or time steps needed to be specific. However, spiking neurons could only produce one spike at a time step. Representing all the possible real-valued activation needs a high degree of space. LSQ has demonstrated an effective way to quantise ANN activation values into a much smaller space that is representable with as low as 2 bits, i.e. only four possible values (Esser et al., 2020). Simply following LSQ by inserting another set of parameters, namely quantisation step size, into the fake-quant node, a pre-trained model can be fine-tuned together with the training of these new parameters. Subsequent conversion just follows the typical steps but substituting the conversion parameters with the scaling parameters.

For example, if an ANN has been trained on N bits range of LSQ, this would mean that LSQ has figured out a scaling parameter S_q^l for a certain layer l so that the network can still work with N-bit precision without losing much accuracy. Then the conversion from this quantised ANN to SNN can be done with the following calculation:

$$\tilde{W^l} = V_\theta^{l-1} \cdot W^l \tag{7.4}$$

$$\tilde{B^l} = B^l \tag{7.5}$$

$$V_\theta^l = (2^N - 1) \cdot S_q^l \tag{7.6}$$

Where $\tilde{W^l}$ is the converted weights for spiking neurons at layer l, V_θ^l represents the threshold voltage of layer l, $\tilde{B^l}$ is the new bias value. The needed parameters are W^l, B^l and S_q^l and number of bits N.

7.3 Spiking bin-ratio ensemble neural network

The mentioned methods above were utilised for our BEANN conversion. In addition, the following optimisations for this SNN deployment are included:

- Instead of using division, a rounded base-2 log-subtraction encoding was employed
- Pruning was added during the SNN conversion to reduce the computational overhead
- Quantisation of SNN weights with normalisation offers a fixed-point algorithm to be implemented on FPGA

The overall flow for this algorithm is shown in Figure 7.11. There are 20 different degrees of

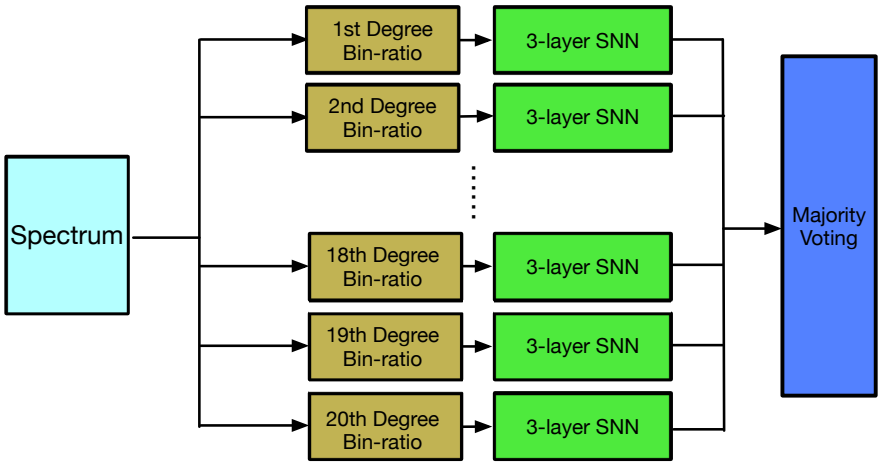


Figure 7.11: Bin-ratio spiking ensemble data flow

classifiers in this ensemble, each classifier consists of a bin-ratio encoding (pre-processing) and a 3-layer fully connected SNN. As for each SNN, the architecture stays consistent as in (S. Zhang et al., 2022). That is input-FC40-FC18, (FC means fully connection) where input is the processed bin-ratio data, and the number of input bins depends on the degree of the classifier, this will be explained further in the following section. The spectrum in this application has 1024 energy bins used as input for the ensemble. The majority voting by the end takes in 20 verdicts and makes the final inference. Each part of this ensemble flow will be discussed in detail.

7.3.1 ANN ensemble re-implementation

The original algorithm by S. Zhang et al. was implemented in Matlab. However, the later conversion stage needs a Python compatible weight format and model structure. A Python re-implementation of the original model was then performed.

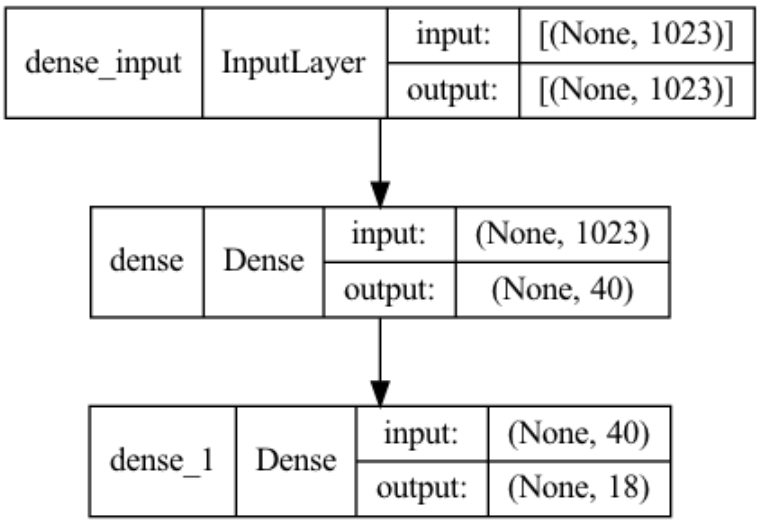


Figure 7.12: Structure of a single ANN network

There are two main frameworks in ANN construction: TensorFlow and PyTorch. The ensemble ANN implementation was constructed in TensorFlow because of its highly abstract interface, which allows simplicity at the methodology verification stage. Each net has a similar structure shown in Figure 7.12. But what was shown in the figure is the ANN model for the first diagonal, because there are 1023 input bins in the bin-ratio dataset. There are 9000 data samples, with a partitioning of 15%:15%:70% representing test set, validation set and training set.

Regularisation

This is still a reproduction, therefore, the activation function was set as "sigmoid", the same one as the Matlab implementation. The final output was normalised by softmax function and the loss function was cross-entropy. Adam optimiser was used with learning rate at 0.001 and beta_1 at 0.9, and beta_2 at 0.999. L2 regularisation was used to prevent overfitting with the rate set at 0.001. However, the first diagonal model suffers a slightly strong overfitting issue 7.13. The gap between the accuracy of the training and validation is nearly 20%, which is a clear sign of overfitting.

Tried another stronger L2 regularisation, however, this has mitigated the overfitting at the cost of model's performance. The validation accuracy dropped from 74.4% to 62.8%, even though the accuracy gap between training and validation has shrunk from 21% to 14%.

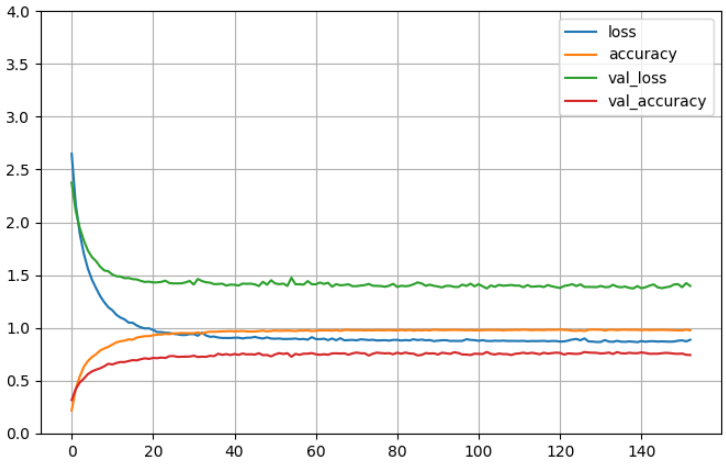


Figure 7.13: Clearly overfitted ANN model, metrics V.S. epochs (L2 rate = 0.001)

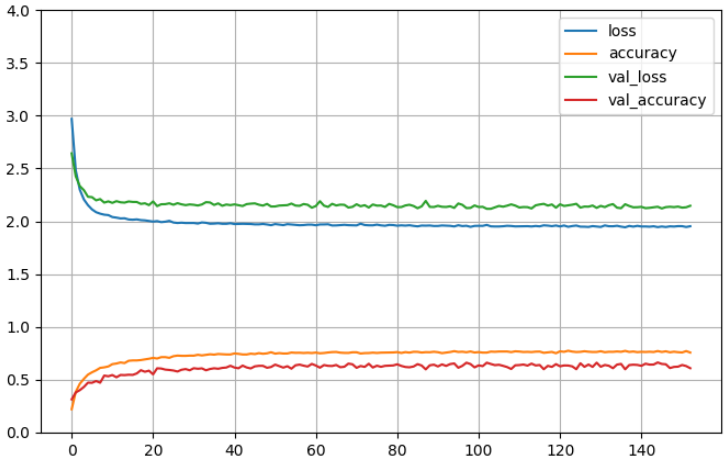


Figure 7.14: Improved overfitting at the cost of performance, metrics V.S. epochs (L2 rate = 0.005)

Given that L2 regularisation was not working well on TensorFlow implementation, another regularisation technique was attempted: dropout. This technique randomly masks out part of features or neurons during training at a single iteration, so that a subset of features or neurons will not contribute to the loss and will not be updated. This method eliminates the overly heavy reliance the network on a specific feature or neuron by spreading the association to basically every neuron. This seems to be working well for our model as well.

An experiment of dropout regularisation was carried out. First a dropout rate of 0.2 was applied to both layers in the model:

```
model = keras.Sequential([
  keras.layers.Dropout(rate=0.2),
  keras.layers.Dense(40, activation='sigmoid',
  kernel_initializer=initalizer),
  keras.layers.Dropout(rate=0.2),
  keras.layers.Dense(18, activation='softmax',
  kernel_initializer=initalizer)
])
```

Listing 7.1: Dropout inserted ANN model

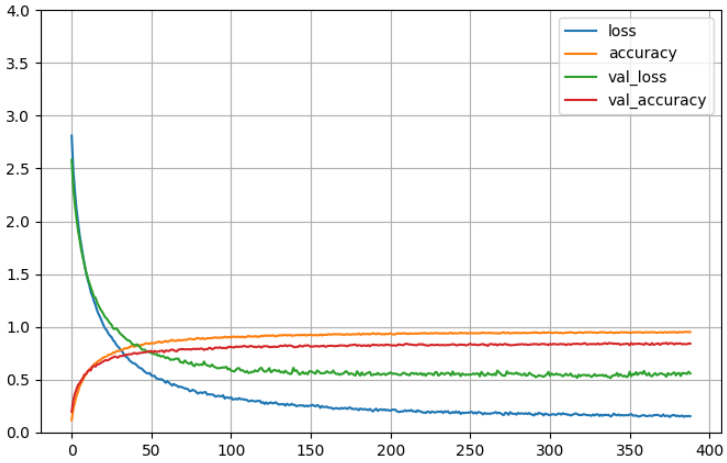


Figure 7.15: Dropout experiment shows improvement in both performance and overfitting (dropout rate=0.2)

This resulted in a model performance with higher accuracy and much lower overfitting as shown in Figure 7.15. Statistically, the accuracy gap between training and validation is now simply 11% and the overall validation accuracy is 84.2%, the highest so far.

But what might be the best dropout rate? A simple parameter search was operated on the first diagonal ANN. The search field starts from 0 to 0.8 with the search step of 0.02. Plot the test accuracy and loss against the dropout rate it can be seen that the optimal choice of dropout rate is around 0.2/0.3. A compromised 0.25 dropout rate was used for the following implementation.

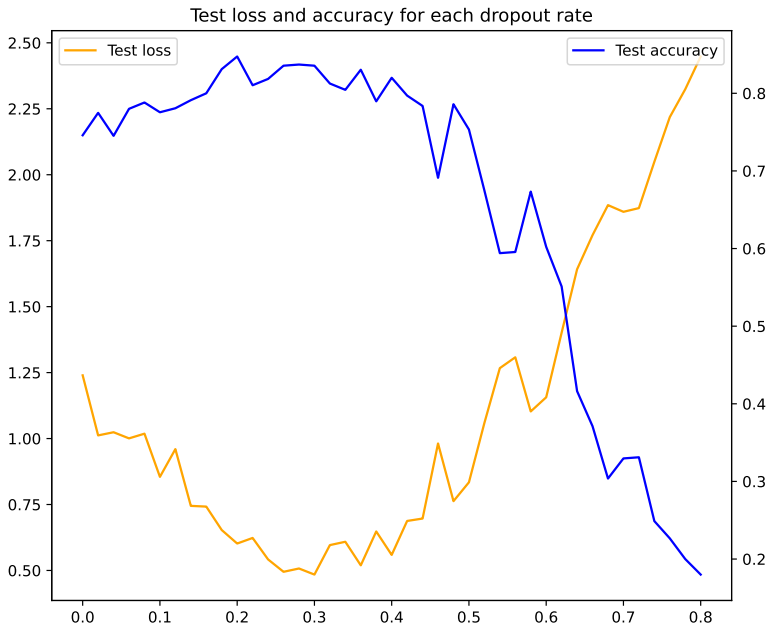


Figure 7.16: Dropout rate V.S. test loss and accuracy

Ensemble building

By extending the configurations to other 19 networks for each diagonal, we can have a whole ensemble of bin-ratio ANN for RIID. If counting each inference form a single net as one vote, and simply check the votes. The final implementation of the original bin-ratio ensemble was successfully reproduced in Python. This shows the final accuracy of the BEANN re-implementation at 97.3%. This result is endorsed by the confusion matrix illustrated in Figure 7.17.

The whole ensemble model has been later changed slightly in the configuration settings where "sigmoid" activation was replaced with "ReLU" for better conversion preparation. The following sections after this successful reproduction have all been following the "ReLU" activation instead.

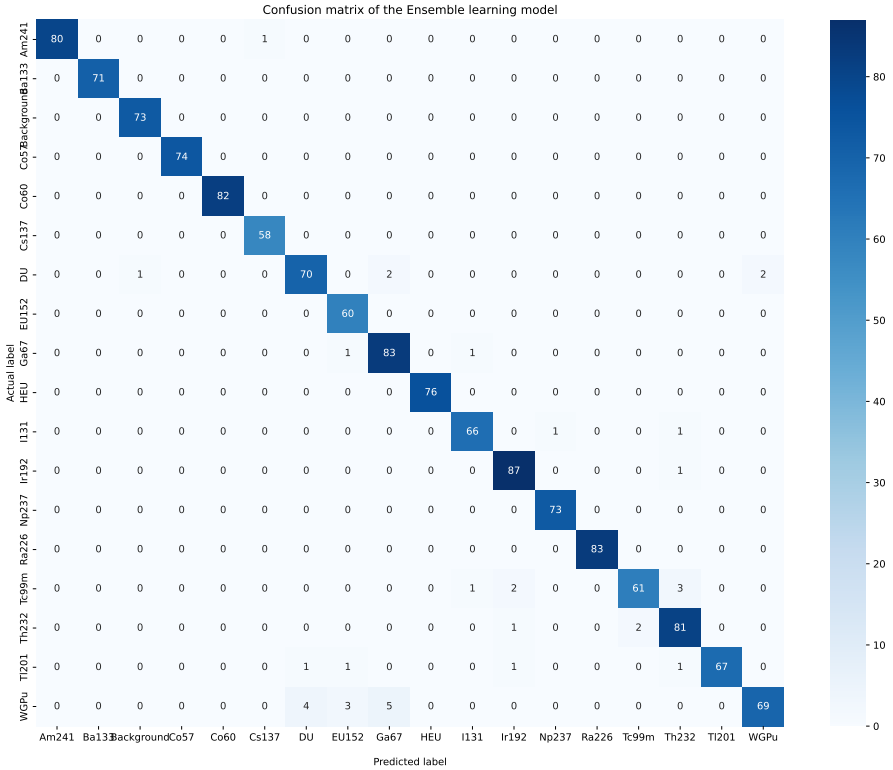


Figure 7.17: Confusion matrix for the BEANN re-implementation in Python

Majority vote

There are two different majority vote, hard voting and soft voting. Hard vote assigns each member in the ensemble the same weight, with one vote coming out each network. By counting how many votes a specific class receives from the ensemble, the final verdict can conclude the inference as the class that receives the most votes. Soft vote works slightly different by adding up all the final activation from each net and conclude with the highest activation values as the winner.

There has not been a strong evidence showing that one specific vote performs better than the other. For example, the results concluded in Figure 7.17 using hard voting give 97.3%, but the soft voting presents an accuracy of 97.5%. These two results show a small advantage using soft voting, but this is not always true. This will be shown in the later section.

7.3.2 Dynamic range compression re-implementation

Even though the original bin-ratio proves to be feasible and a competitive algorithm for RIID. The final goal for this algorithm is hardware deployment. Division is not an easy option for hardware compared to addition and subtraction. Therefore, in this subsection, an equivalent alternative pre-processing shall be explored.

Natural logarithmic pre-processing

According to logarithmic computation rule, the logarithm of a division is equal to the subtraction of these two logarithm.

$$log_a(\frac{A}{B}) = log_a(A) - log_a(B) \tag{7.7}$$

Applying logarithm to bin-ratio values can convert the original division into subtraction effectively.

However, the original values in the dataset range from 0 to 255k. Given they are all integer numbers, a natural logarithm projection is visualised in Figure 7.18 The only incalculable

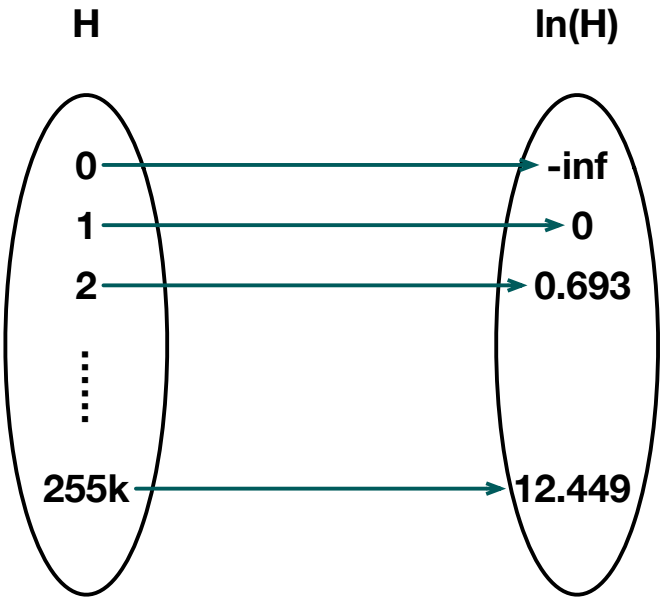


Figure 7.18: Natural logarithm projection from bin value to the new field

element brought by the projection is $0 \rightarrow -\infty$. Here we replaced $-\infty$ with 0 after logarithm, followed by subtraction between bins. The newly pre-processed dataset is ready for the BEANN algorithm. This has transformed the original dataset into a more compressed range.

Following the same configuration as before, an ensemble was trained and implemented in TensorFlow. This implementation shows the confusion matrix as in Figure 7.19. This implementation presents a soft voting accuracy of 97.33% yet produces hard voting accuracy of 97.41%.

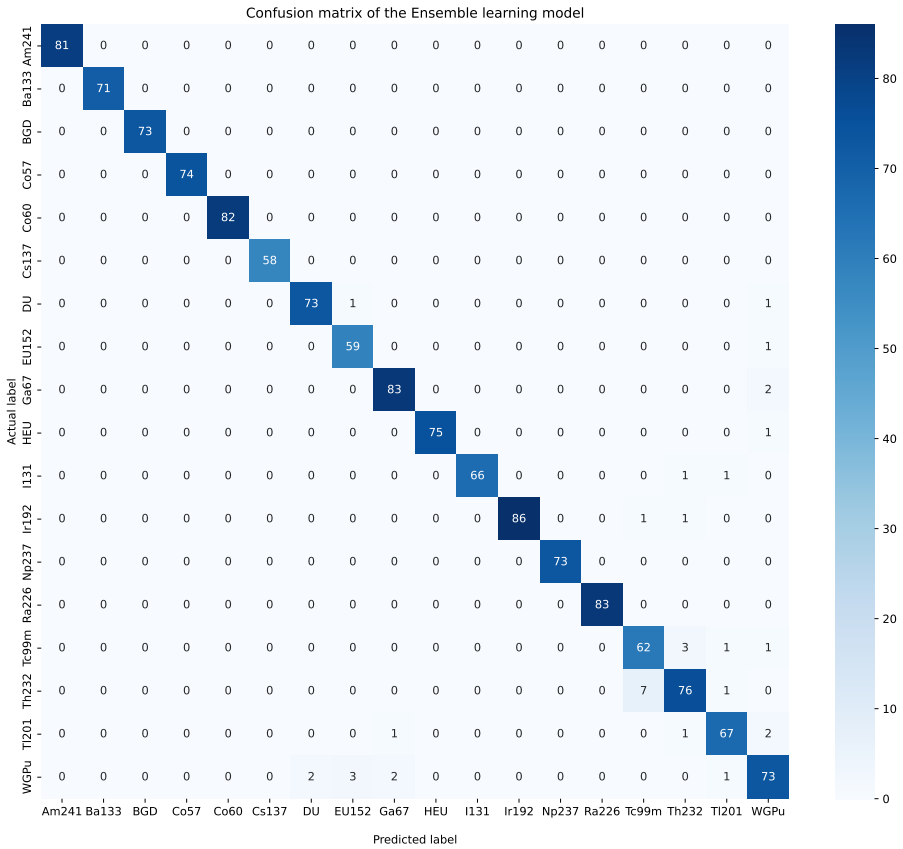


Figure 7.19: Confusion matrix for the BEANN on natural logarithmic data

Non-negative logarithmic pre-processing

To eliminate the negative infinity that needs extra attention before feeding into neural network for training, a slightly different pre-processing was applied so that the transformation will not have negative values in the target field. This is illustrated in Equation 7.8.

$$ln(\frac{A}{B}) \approx ln(\frac{1+A}{1+B}) = ln(1+A) - ln(1+B) \tag{7.8}$$

By using this type of pre-processing, another BEANN was trained. This gives us a slightly boosted accuracy of hard-voting and soft-voting both at 98.52%.

7.3.3 Towards more hardware-friendly pre-processing

The previous implementation has shown that the ensemble model demonstrates better performance for range-compressed data. Besides, it demonstrates the robustness of this algorithm towards differently pre-processed data. To move towards a more hardware friendly algorithm, some optimisation techniques were applied. These were designed to round output of the pre-processing to integers so that hardware can compute at a low cost. Therefore, the following optimisations were introduced:

- A base-2 logarithm computation for easier hardware implementation
- A shifted logarithm to eliminate incalculable numbers
- Rounding to the closest integers after logarithm for fixed point arithmetic

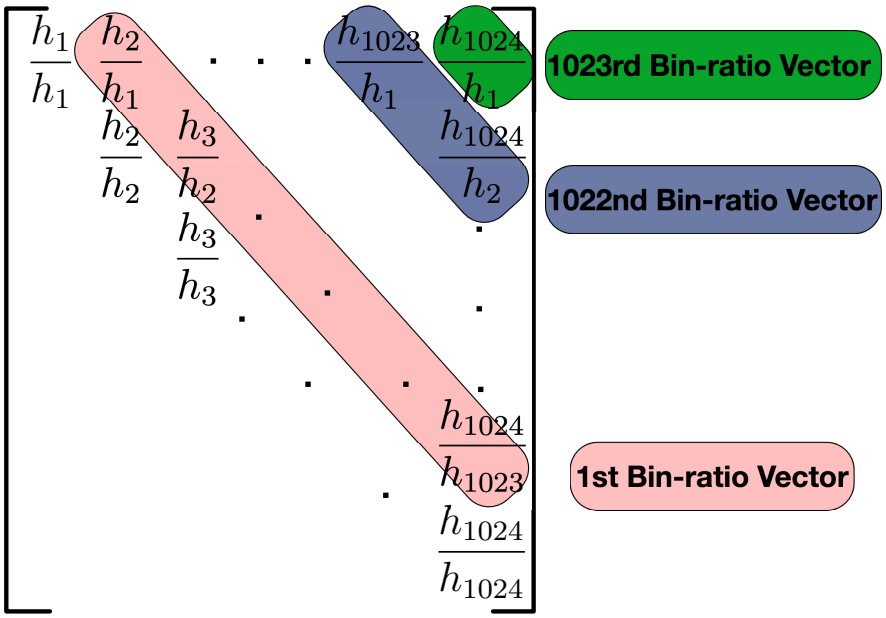
New bin-ratio encoding

First we have made a small change to the original bin-ratio matrix, where the ratio were replaced with base-2 shifted logarithm to get vectors for different degrees.

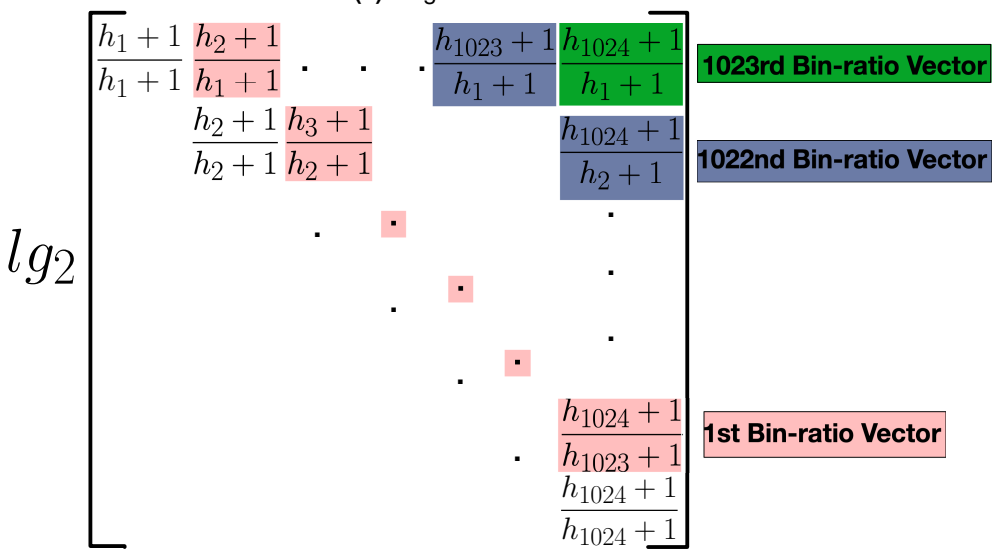
As for the original bin-ratio matrix, it is explained in Figure 7.20a. In a spectrum, there are values located at each bin, they represent the intensity of the gamma radiation at the corresponding energy level. They are denoted as in h_i, where i is the index of the bin. The generation of this matrix is straightforward. Simply divide 1024 values by each individual value h_i at row i. After this process, different diagonals of this matrix are taken as the input for training and testing of the ANN. Therefore, each ANN has a different input size. Diagonal 1 has 1023 inputs, diagonal 2 has 1022 features and so on. This method is a good strategy to tackle common issues with RIID since the division of bin values is not susceptible to noise. For example, gain shift and background noise of the spectrum will be eliminated in the division of the data. This is because bin-ratio reflects the correlation between each bin instead of the absolute value.

As presented in Figure 7.20b, an equivalent logarithm was applied in our work. This is because it permits a large dynamic range within a small number of bits while maintaining granularity in the representation of lower-magnitude values. Additionally, the logarithmic compression makes the many division operations simpler to compute.

First, our dataset has a substantially big range with a maximum value of around 256k. To represent such a big range, 20 bits are preferably needed throughout the whole computation even after division. Logarithmic processing is capable of compressing the range drastically. Thanks to this pre-processing, 8 bits are enough for our case. On the other hand, through the laws of logarithms, when values are compressed, multiplication or division of the raw values can be carried out using addition/subtraction of the compressed representations. Our



(a) Original bin-ratio matrix



(b) Bin-ratio matrix in this work

Figure 7.20: Bin-ratio matrix in the original algorithm and this work

approach only needs a fixed 3 operations to finish, that is two logarithms and one subtraction. This is much less than a typical division operation, which iteratively does subtraction and multiplication. The number of operations is therefore not fixed but at least 3: multiplication, subtraction, comparison of the remainder and divider.

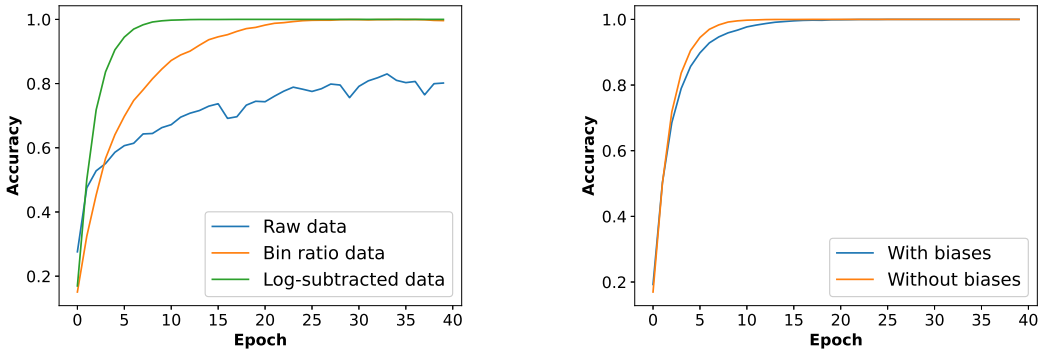
However, there might be an out-of-range issue with the logarithm when the input is 0, which should give negative infinity theoretically. The logarithm function used here has been replaced by $log_2(x+1)$, so that all the input integer values can be processed properly. To conclude, as for the diagonal i, all the features in this should be:

$$h' = \{log_2(h_m+1) - log_2(h_{m-i}+1) \mid i+1 \leq m \leq 1024\} \tag{7.9}$$

In a set indexed diagonal i, there should be $1024-i$ elements.

Integer input

Furthermore, we have the pre-processed data rounded to the closest integer to keep as much precision as possible. To prove the validity of this pre-processing, training on different ways of pre-processing was done. 40 epochs of training on raw data, bin-ratioed data and our log-



(a) Training accuracy on different datasets **(b)** Training accuracy with or without biases

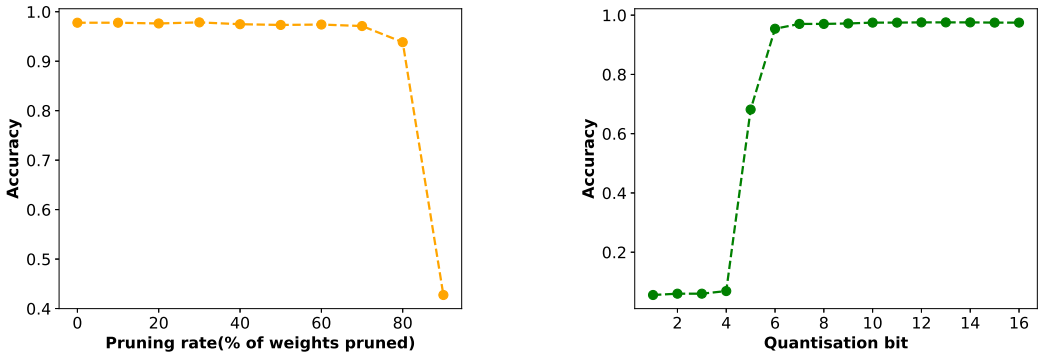
Figure 7.21: Training accuracy v.s. epoch

subtracted data is shown in Figure 7.21a. The simple ANN used in this case has the same architecture as the individual net in the ensemble. For the ease of the later conversion stage, we trained the nets without biases. This proves to be feasible since the trained model without biases gives similar accuracy to the one with biases as shown in Figure 7.21b.

7.3.4 Conversion and optimisation

Pruning stage

After the training of each net, a pruning process was performed, during which weights were progressively pruned by magnitude until it reached the level of sparsity that was set. A few epochs of training will be done right after to fine-tune the unpruned parameters so that accuracy will be retained. This could be done with the offered API from TensorFlow, which is what we used. After pruning, all the models in the ensemble are significantly reduced to



(a) Pruning rate v.s. ensemble accuracy

(b) Quantisation bit v.s. ensemble accuracy

Figure 7.22: Optimisation techniques: pruning and quantisation

just 30% of the original parameters. This is the highest sparsity before the accuracy drops noticeably as indicated by Figure 7.22a. The total parameters of the ensemble were reduced from 825,200 to 247,600.

Conversion stage

To do the conversion, the LSQ training needs to be done. This is to find the learned step size for activation and later used to calculate the threshold. A bit-width of 2 was used in our case, this is the least bits possible before pushing the nets into a binary network. Consequently, there are four different levels of activation values after quantisation. If the learned step size can be annotated as S_q, the activation values could only be as follows:

$$\{S_q * b \mid b \in \{0,1,2,3\}\} \tag{7.10}$$

With the step size, conversion and normalisation can be done with the formulas provided below:

$$V_\theta^l = (2^N - 1) \cdot S_q \tag{7.11}$$

$$\tilde{W}^l = V_\theta^{l-1} \cdot W^l \tag{7.12}$$

N stands for the bit-width set during quantisation training, V_θ^l symbolises the threshold for layer l, S_q is the learned step size and $\tilde{W}^l$ represents the weights after normalisation.

Following as in the original paper (C. Li et al., 2022), analogue input was used at the first layer to generate spikes. Technically, they could be considered as the current injection with different weights. As a result, rate coding noise could be eliminated. This type of noise mainly comes from the stochasticity when using rate-based encoding. Theoretically, our 2-bit LSQ conversion will have the max accuracy with 4 time steps, since 4 steps is the time it needs to represent 4 different rates.

One thing that needs to be mentioned is that the pruned model needs to stay pruned during LSQ training. In other words, the weight parameters that are 0 need to stay 0 and nonzero values should be fine-tuned during this quantisation-aware training. Inspired by the methodology of pruning, a simple prune-all-zero filter was applied before fake-quant node insertion. This pruning class would create a mask to keep weights that are 0 before training from being updated. The combination of the pruning class and fast SNN conversion technique has allowed us to provide SNNs with more sparse connections.

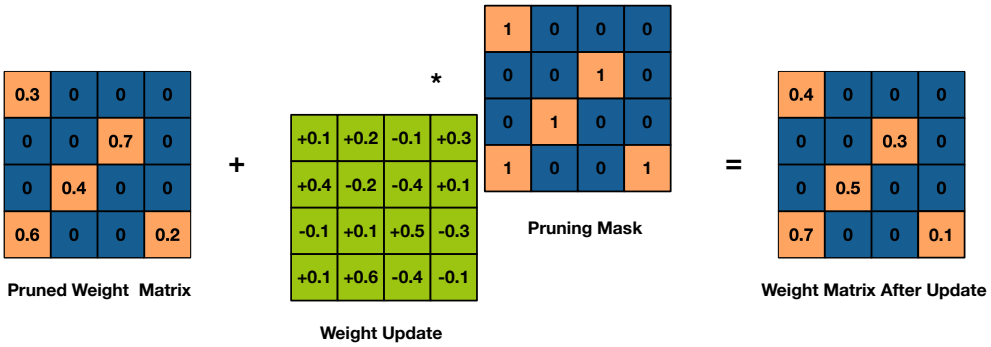


Figure 7.23: Pruning masks applied during weight updates

Quantisation stage

Quantisation is the key stage where all parameters are quantised into integers for the ease of hardware implementation. Since the computation unit for our algorithm is IF neurons, when the parameters in one layer are scaled simultaneously, the behaviour should stay unchanged. The parameters here include weights and thresholds. They should be multiplied by the same coefficient. In the meantime, the scaled parameters should stay in a range that is representable with 8 bits. Therefore, normalisation should be done to find the closest integer that gives the best approximation. The quantisation flow is described in Algorithm 4.

The quantisation bridges the floating-point algorithm and fixed-point calculation. An 8-bit representation was chosen based on the trade-off between accuracy and bit-width. From Figure 7.22b, it could be observed that the accuracy of our network remains lossless after 9 bits compared with the floating point algorithm. With 8 bits, there is an acceptable accuracy loss under 0.5%. This means that the weights and threshold have a range between -128 and 127. Besides, the 8-bit format is a more commonly used bit-width for fixed-point arithmetic.

Algorithm 4 SNN normalised quantisation

Input: $W = [w^l], V_\theta = [v_\theta^l], bit = m \in Z^+$
Output: $\tilde{W} = [\tilde{w^l}], \tilde{V_\theta} = [\tilde{v_\theta^l}]$

$\tilde{W} = [\,], \tilde{V_\theta} = [\,]$
$L \leftarrow len(W)$
$N_{max} \leftarrow 2^{m-1}$
$N_{min} \leftarrow -2^{m-1}$
for $i \leftarrow 0, L-1$ **do**
    $Param \leftarrow \{w^i, v_\theta^i\}$
    $Param \leftarrow round(norm(Param) * N_{max})$
    $clip(Param, N_{min}, N_{max} - 1)$
    $\tilde{W}.append(Param[w^i])$
    $\tilde{V_\theta}.append(Param[v_\theta^i])$
end for

7.4 Dataset

The dataset employed in this work is derived from real-world isotope spectra collected by Kromek Group PLC and Oak Ridge National Laboratory and referenced as enhanced CLLBC synthetic dataset in Chapter 3. The data acquisition instrument is a medium-resolution scintillation detector with a full-width at half maximum (FWHM) resolution of less than 4% at 662 keV. The isotope library contains 18 radioisotopes that meet American National Standards Institute performance standards for handheld instruments for the detection and identification of radionuclides (ANSI N42.34-2015 (*American National Standard Performance Criteria for Handheld Instruments for the Detection and Identification of Radionuclides, ANSI N42.34-2015 (Revision of ANSI N42.34-2006)*, IEEE Standard, 2016)) including ^{241}Am, ^{133}Ba, ^{57}Co, ^{60}Co, ^{137}Cs, ^{152}Eu, ^{226}Ra, ^{232}Th, ^{67}Ga, ^{131}I, ^{192}Ir, ^{99m}Tc, ^{201}Tl, ^{237}Np, depleted uranium (DU), highly enriched uranium (HEU), weapons-grade plutonium (WGPu), naturally-occurring radioactive materials (NORM). To simulate real scenarios in the synthetic dataset, background noise and gain shift noise were added. Four different types of background spectra were mixed in random proportions to generate multiple background spectra. Additionally, randomly generated gain shift noise was introduced into the final dataset.

As for the dataset partitioning, we followed closely with the original BEANN algorithm implementation where 70% of the dataset was used for training, 15% of the data was used for validation and 15% of the data was for testing. Given there are 9000 samples in total, 6300 of them were used for training, 1350 were used for testing.

7.5 Conclusion

In this chapter, we introduced the newly proposed spiking version of bin-ratio ensemble algorithm for RIID. This is based on the original bin-ratio ensemble ANN with hardware-oriented optimisation and converted by quantisation framework for fast SNN. This is combined with the pruning method which keeps the SNN with sparse connections as low as 30% of the original parameters. Following this, a normalised quantisation scheme was used to quantise SNN weights into 8-bit integers from 32-bit floating point numbers. All the techniques mentioned above has made SNNs ready for hardware implementation. The entire algorithm flow is listed in Figure 7.24.

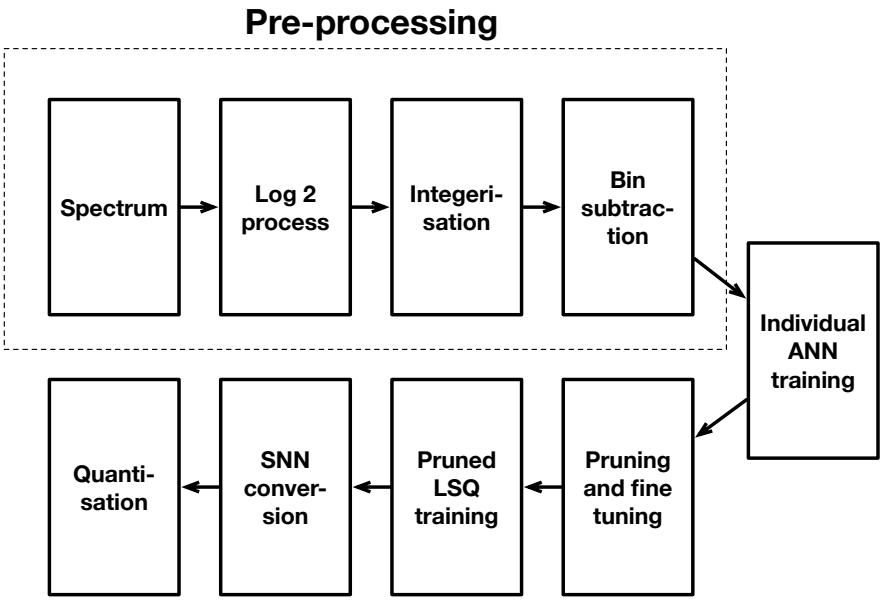


Figure 7.24: Algorithm flow of this work

Chapter 8

Hardware implementation

Following last chapter where the original BEANN was successfully converted into an 8-bit quantised spiking bin-ratio ensemble neural network, this chapter will be focused on the hardware architecture design. The main top down architecture will be discussed, where an overview of the whole hardware design is illustrated. At each hierarchy, the detailed design methodology and constitution will be explained. The main objective is to implement the hardware on an FPGA with hardware description language (HDL) on Xilinx FPGA using Vivado, an electronic design automation tool, to make sure the consistency between SNN software simulation and hardware simulation.

8.1 A top-down overview

The whole top-down hierarchy overview of this bin-ratio ensemble SNN (BESNN) is depicted in Figure 8.1. The whole ensemble network is made up of multiple single nets. Within one

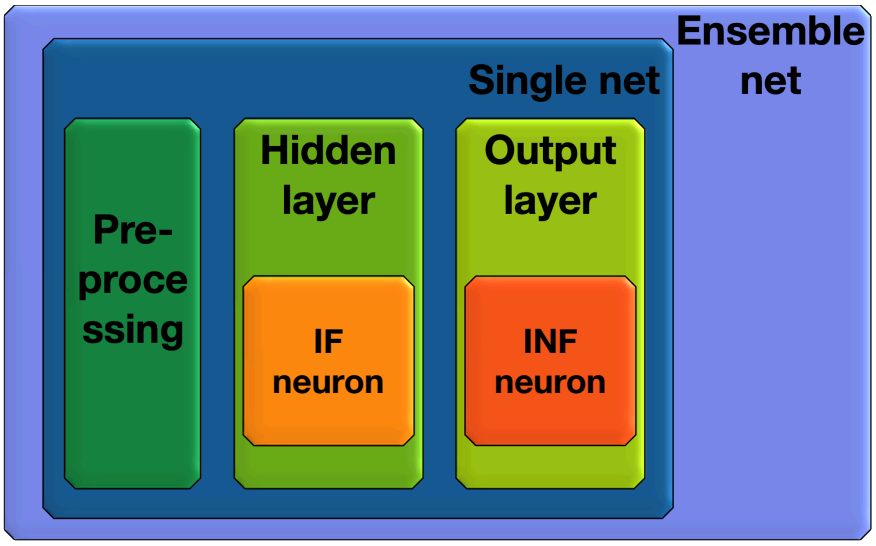


Figure 8.1: Top-down hierarchy of BESNN

single net, there are pre-processing layer, hidden layer and output layer. Pre-processing layer is mainly used to convert raw data into base-2 shifted log-subtracted data. Since the original

single net in the BEANN is simply made up of 3 layers, all we need for the SNN computation is the hidden layer and output layer. The core to hidden layer is an IF neuron, which is used in a time-division multiplexing (TDM) way. While the hidden layer itself combines the analogue input and encoding into one encapsulation. The output layer, on the other hand, simply accumulates but does not fire and therefore incorporates an integrate-and-no-fire (INF) neuron. This is originally designed in the algorithm to retain accuracy of the converted SNN. Each module in this hierarchical view has an accompanying state machine to operate. Details for each module will be explained in the following section

8.2 Pre-processing layer

Just as discussed in the last chapter, the pre-processing mainly operates the procedure described below: Where the algorithm 5 takes in the raw data array with the size of 1024 and

Algorithm 5 Process Raw Data

```
1: procedure ProcessRawData(raw_data_array, diagonal)
2:     new_data_array ← array(size = 1024 − diagonal)
3:     log_data_array ← round(log2(1 + raw_data_array))
4:     for i ← 0 to 1024 − diagonal − 1 do
5:         new_data_array[i] ← log_data_array[i + diagonal] − log_data_array[i]
6:     end for
7:     return new_data_array
8: end procedure
```

diagonal, where diagonal is an integer that represents the diagonal of the bin-ratio matrix. The output should be a new data array with the size of $1024 - diagonal$. The third line describes the input array being logarithmic computed with the shifted base-2 log function, the product is then rounded to the closest integer. The log data array still has the same size as input with 1024 elements. The following is the division alternative—subtraction. Where the subtracted data is then saved into the new data array of the size $1024 - diagonal$, this subtraction is illustrated in Figure 8.2.

8.2.1 Logarithm module

The integerised shifted base-2 logarithm could not be natively implemented using fixed-point arithmetic, therefore we proposed the equivalent fixed-point logic to be implemented. The computation flow can be found in the Algorithm 6. The whole idea is to identify whether or not to use the maximum bit-width or one bit less than that. By looking at Algorithm 6, it can be seen that the input is basically the value in a single bin. This is an integer with the range between 0 and 256K, therefore the input raw data value can be represented with at least 18 bits. To give the design an even bigger range, 20 bits were used. By binary representation, this range has

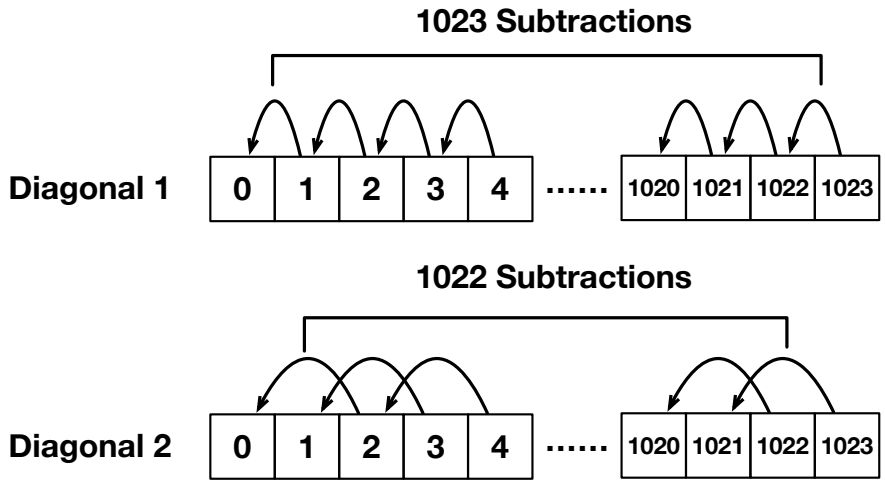


Figure 8.2: Diagonal subtraction in an array

Algorithm 6 Rounding base-2 logarithm

Input: Integer to be processed $N \in [0, 2^{20}-1]$
Output: Processed integer n

$BW \leftarrow 0$
$M \leftarrow N+1$
while $M > 0$ **do** ▷ Get the maximum bit-width it takes to represent N+1
    $BW \leftarrow BW+1$
    $M \leftarrow M \gg 1$
end while
if $[(N+1) \ll (20-BW)] \geq (\sqrt{2} \ll 19)$ **then** ▷ If $log_2(\frac{N+1}{2^{BW-1}}) \geq 0.5$
    $n \leftarrow BW$
else
    $n \leftarrow BW-1$
end if

the minimum of 0 and maximum of $2^{20}-1 = 1,048,575$. In this algorithm, there are mainly two parts. One is to calculate the maximum bit-width needed to represent the input plus 1. The other part is to do the rounding to the closest integer. The first part is easily understandable because of the binary representation. For example, number 1_d in binary is 1_b, which needs 1 bit, while number 123_d in binary is 111_1011_b, which takes 7 bits. This is achieved by right shifting the number until the number is 0. The second part is to calculate the result of the rounding function. The implementation is also simple maths equivalent transformation.

$$log_2(M) = log_2(\frac{M}{2^{BW-1}} \cdot 2^{BW-1}) \tag{8.1}$$

$$= (BW-1) + log_2(\frac{M}{2^{BW-1}}) \tag{8.2}$$

$$= (BW-1) + log_2(M \gg (BW-1)) \tag{8.3}$$

BW is the maximum bit-width while $M = N + 1$, where N is the input raw integer value. Apparently, in Equation 8.3, the integer part of $log_2(M)$ should be $BW - 1$ and the fractional part is $log_2(M \gg (BW - 1))$. To round this number, all we need to compare is the fractional part and 0.5. Therefore, if $log_2(M \gg (BW - 1)) > 0.5$, rounded value of $log_2(M)$ should be BW, otherwise it should be $BW - 1$. In other words, if $M \gg (BW - 1) \geq \sqrt{2}$, output should be BW, otherwise output $BW - 1$. Equivalently, this comparison can be done between $M \ll (20 - BW)$ and $\sqrt{2} \ll 19$, or $1011\_0101\_0000\_0100\_1111_b$.

This implementation has proven to be highly accurate. For example, as for the boundary number 90 ($0000\_0000\_0101\_1010_b$) and 91 ($0000\_0000\_0101\_1011_b$) both with $BW = 7$, base-2 logarithm for them should be 6.492 and 6.508, which should round to 6 and 7 respectively. As for number 90:

$$\begin{aligned} 0000\_0000\_0000\_0101\_1010 \ll (20 - 7) &= 1011\_0100\_0000\_0000\_0000 \\ &< 1011\_0101\_0000\_0100\_1111 \end{aligned}$$

Therefore, the output for 90 should be $7 - 1 = 6$. As for number 91:

$$\begin{aligned} 0000\_0000\_0000\_0101\_1011 \ll (20 - 7) &= 1011\_0110\_0000\_0000\_0000 \\ &> 1011\_0101\_0000\_0100\_1111 \end{aligned}$$

Consequently, the output for 91 is 7.

8.2.2 Shift register module

After logarithmic processing, the subtraction could be easily done with shift registers where the length corresponds to the diagonal of the network. This design ensures the subtraction between numbers is equally intervaled. Number of registers needed N_{Reg} equals the diagonal number plus 1. That is $N_{Reg} = i + 1$.

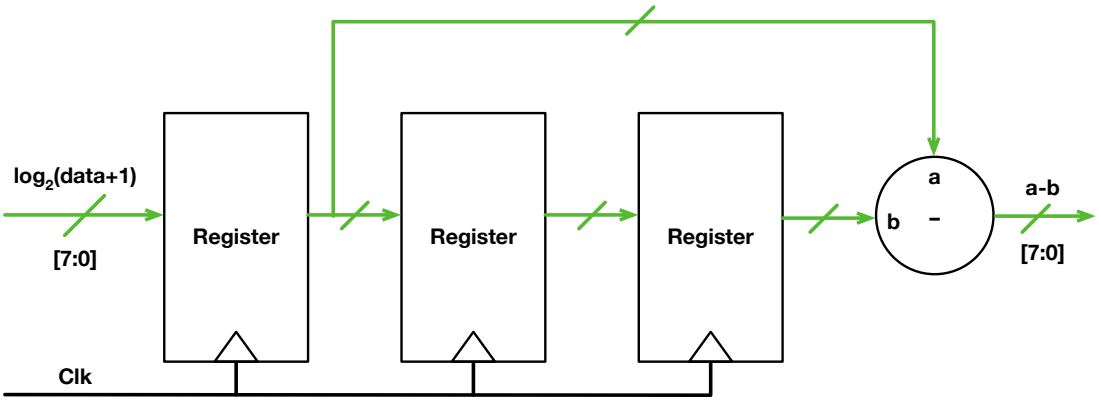


Figure 8.3: An exemplary subtraction design for $2nd$ diagonal network

8.2.3 State machine 1

This pre-processing module is made up of the logarithm module, the configurable shift register module and a state machine to control the flow and communication. The finite state machine (FSM) diagram is described in Figure 8.4. The state machine itself is also fairly simple with

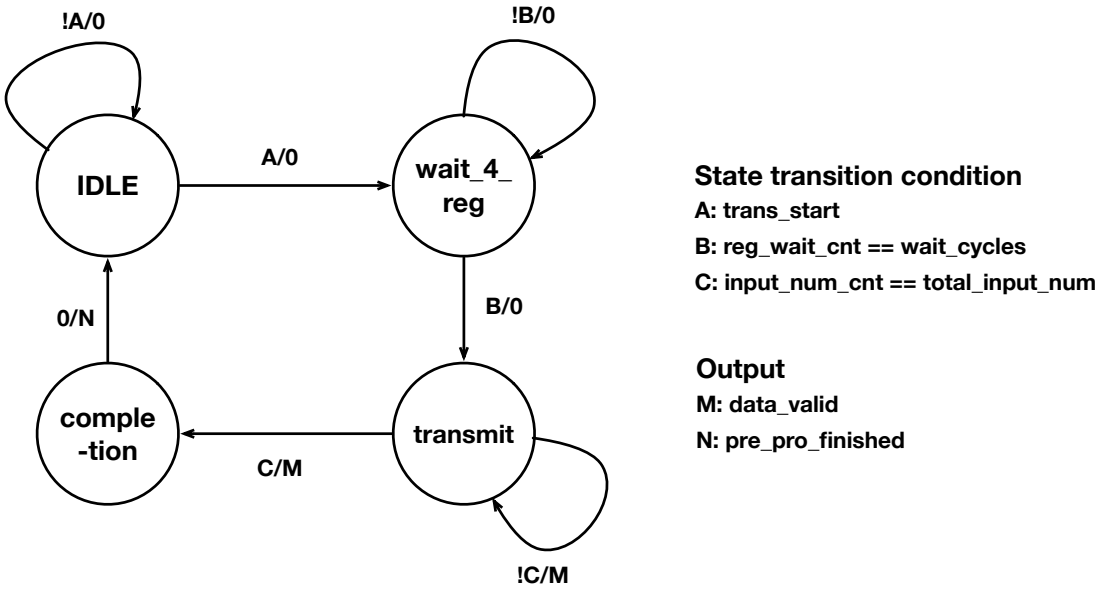


Figure 8.4: FSM diagram for pre-processing module

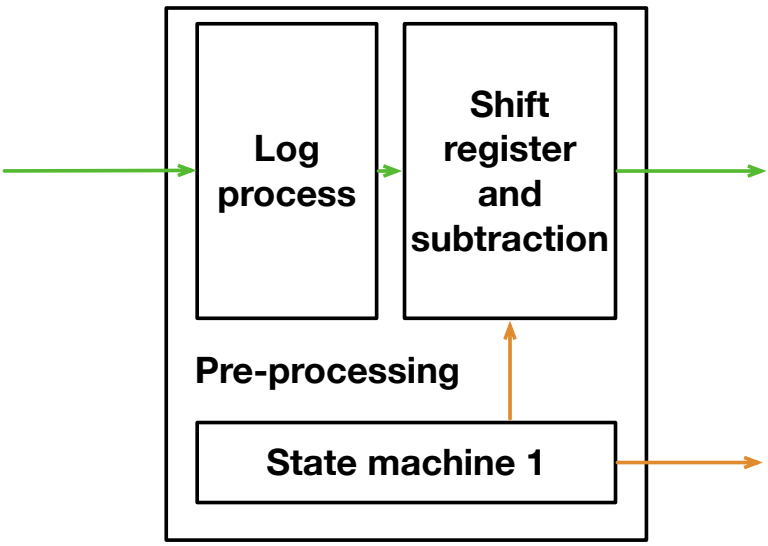


Figure 8.5: Pre-processing layer architecture

only 4 states: ***IDLE***, ***wait_4_reg***, ***transmit*** and ***completion***. This FSM starts at ***IDLE*** state, and will not move on to the next state until it receives a start signal, ***trans_start***. The second

state was designed to wait until the shift registers are all filled up with valid data. The ***transmit*** state simply outputs valid signal to inform the next module along the data flow and makes sure correct numbers of processed data was produced. Last state is a flag state that indicates pre-processing is finished. This layer's overall architecture is summed up as in Figure 8.5.

8.3 Hidden layer

The hidden layer takes over the pre-processed data from the pre-processing layer and use it as the analogue input for the hidden neurons. The hidden neuron accumulates the input as the current injection and fires once this potential reaches the threshold. This process involves related memories to store the input value, weight matrices and membrane potential, a state machine and an IF neuron.

8.3.1 Overall architecture

Overall architecture has been depicted as in Figure 8.6. This has been organised in this

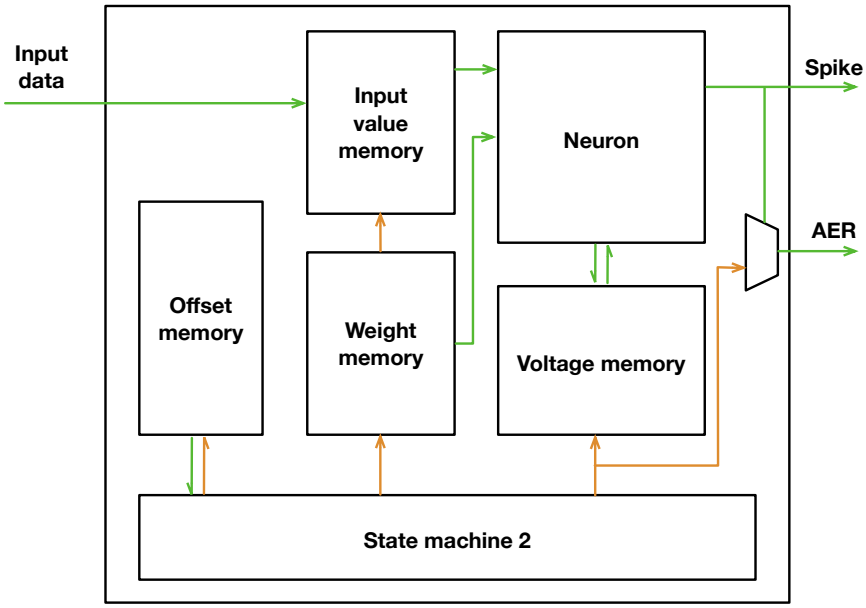


Figure 8.6: Hidden layer architecture

fashion because there is sparse connectivity in the SNN implementation where the sparsified weight matrices are represented in compressed sparse row (CSR) format. Therefore, what needs to be considered while designing this layer is how to efficiently do the sparsified matrix multiplication. Here we used CSR to represent all the non-zero values in the matrix, that includes 3 main vectors:

- All non-zero values

- Index of the values on its row
- Number of non-zero values in each row

As for a matrix with the shape of (m,n), when the sparsity is p. The first two vectors shall have the same dimension as in $(1,m*n*p)$, while the last vector will have the dimension of $(1,m)$. Details of the data structure will be discussed in later subsections.

8.3.2 IF neuron

The neuron design is essentially a signal processing unit that does multiply-accumulate (MAC) operation or simply just accumulate (ACC). MAC is really straightforward to understand because the first layer is basically doing matrix multiplication, which needs a lot of MAC operations. The reason why ACC is needed is because the analogue input computation can be optimised after the first time step is finished. This is simply because the potential increase at each time step is a constant due to the nature of constant current injection. Once the first loop of MAC is finished, the potential increase can be saved for the following steps' computation in a simple accumulative way.

At the initialisation stage, the neuron shall be pre-charged to half of its threshold before MAC computation. Since the threshold is set at a value of 127, the initial pre-charge is set at 63. Thresholds are all set to 127, by the end of the MAC or ACC, if the potential is over the threshold, a spike is generated and the reset-by-subtraction shall return the subtracted voltage back to the memory for next time step. The comparator by the end shall compare the ultimate voltage and output spike (1 or 0) and the final membrane voltage.

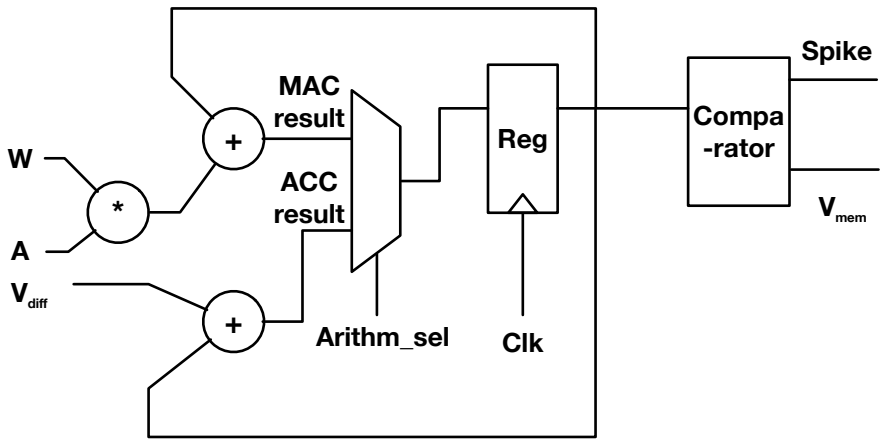


Figure 8.7: IF neuron schematic

8.3.3 All the memory modules

To save up all the necessary weights in the CSR format and the input activation values, various memories are encapsulated to save different information. According to Figure 8.6, there are 4 different types of memories implemented. The content they are enclosing and the memory format will be explained.

First, we shall briefly talk about CSR representation. In a sparse matrix, there are a lot of zero valued elements. 3 vectors are normally used to represent all the needed non-zero values. In Figure 8.8, an example of CSR was given. The example sparse matrix was represented

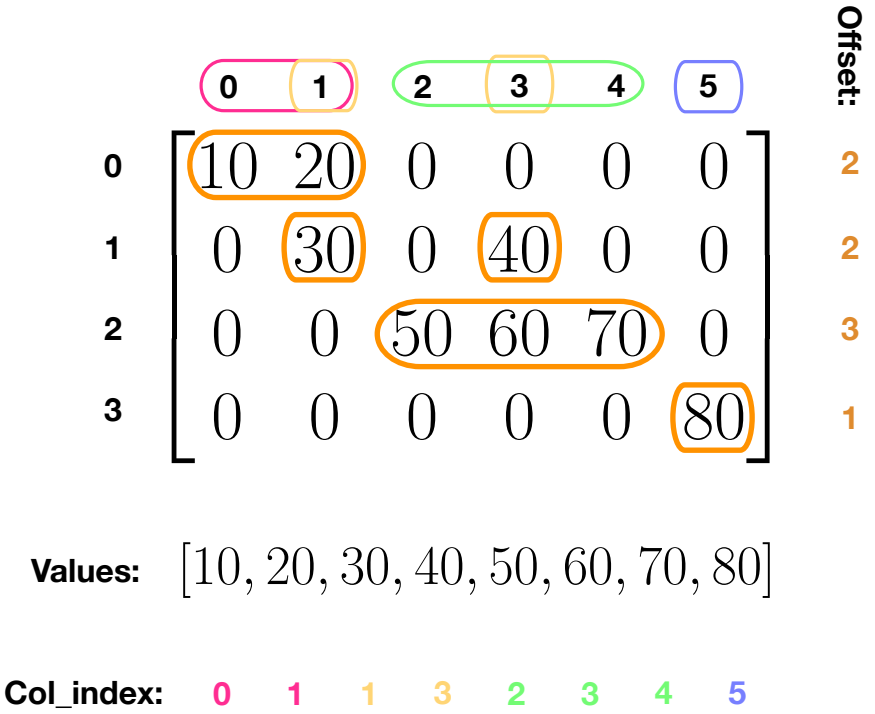


Figure 8.8: CSR representation example

by ***Values***, ***Col_index*** and ***Offset***. Each element in ***Offset*** tells how many non-zero values there are in a row, therefore the length of this vector is also the number of rows for this matrix. ***Values*** vector is filled with all non-zero values extracted from the matrix. Correspondingly, each element in ***Col_index***, who has the same size as ***Values***, indicates the column position within a row.

Similarly, CSC works by compressing the matrix in a column-by-column fashion. It is imaginable that there should be also three vectors: ***Offset***, ***Values*** and ***Row_index***. Sometimes, for the simplicity of counting or quick selection, ***Offset*** can be replaced with a index range to indicate exactly where the corresponding values are in ***Values*** for a specific column. This enables random access, which might be needed sometimes.

Offset memory

This memory straightforwardly saves all the number of non-zero values in each row. Since the first layer of the first diagonal have the sparse weight matrix of size $(40, 1023)$, the biggest possible element in this vector shall be 1023, which is representable using 10 bits. And because the length is 40, 6 bits is enough for all the case. Therefore, the memory should be a read only memory (ROM) with data depth of 40 and data width of 10 bits.

Weight memory

To complement the use of offset memory, the weight memory should save up the rest 2 vectors. And since they have the same size, they can therefore be concatenated into the same ROM.

Thanks to quantisation, weights have all been compressed into 8 bits. However, the index has a bigger range that can be defined with 10 bits. On the other hand, thanks to the pruning process, the sparse weight matrix now only has 30% of the original parameters, and that is $1023 * 40 * 0.3 = 12,276$. This is representable with 14 bits. To conclude, weight memory should be a ROM with data depth of 2^{14} and data width of $10 + 8 = 18$ bits.

Input value memory

Due to the nature of TDM use of the only IF neuron in the module, temporary save of the input value is needed to operate multiple time steps' computation. This has led us to the employment of a random access memory (RAM) to save the input activation values.

The input data is dumped by the pre-processing layer, which should have up to 1023 elements. Each value is represented in 8-bit. This allowed us to temporarily host these data in a 1024×8 RAM, depth:1024, width: 8.

Voltage memory

This memory keeps track of the neuron voltage after each time step and voltage difference after first time step. Therefore it should have depths of 40, each corresponds to a neuron, and width of double the voltage bit-width. By looking at the IF unit, it can be noticed that MAC employs the multiplication of 2 8-bit numbers. This should yield a 16-bit product, i.e. membrane potential. Same applies to the voltage difference.

By concatenating two voltage values, it is not hard to conclude the implementation of the voltage memory should be a RAM with width of 32 and depth of 40.

8.3.4 State machine 2

With all needed components in place, the controller should be designed to coordinate the correct data flow in them. This is a bigger FSM with many more states.

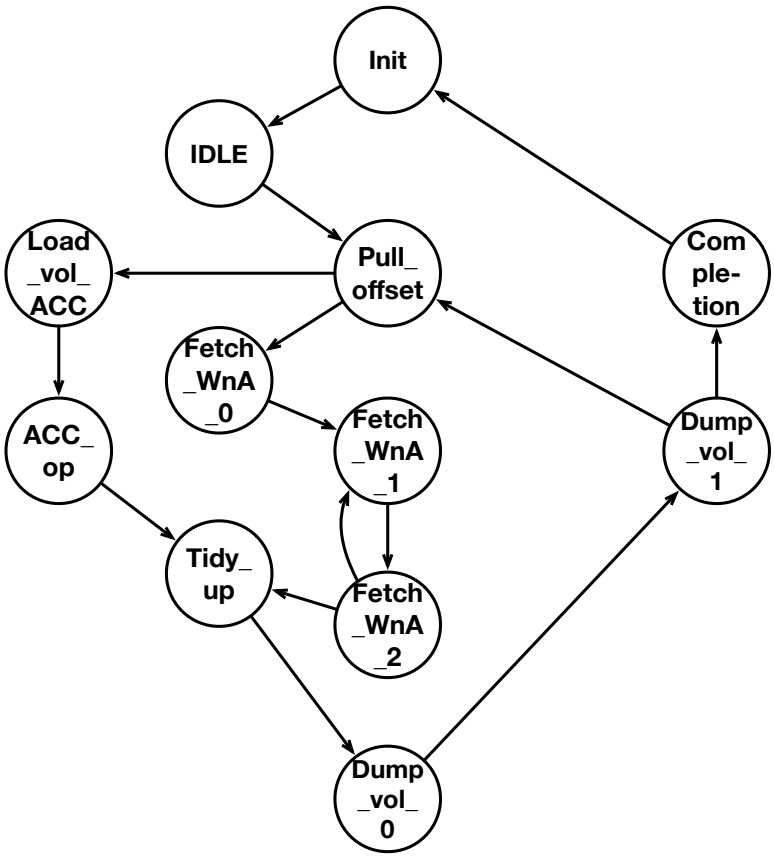


Figure 8.9: FSM chart for the hidden layer module

The state machine 2 follows the computational loop formatted in Algorithm 7. Clearly, the state machine needs to loop through 4 time steps. However, the computational flow is slightly different at the first time step. In the first time step, MACs are operated for every neuron as many times as the connections this neuron possesses. Once this neuron is finished, the voltage difference and the voltage after MACs are exported and saved in the voltage memory. Repeat this process until all neurons are finished, which concludes the current time step. In the following time step, computations are much easier with the neuron voltage and voltage difference. It simply loads the neuron with its last state and accumulates the voltage difference. Therefore, it takes much longer to finish the first time step. But once that is finished, all the rest shall be at least 40 times faster.

Algorithm 7 Hidden layer state machine computational flow

```
1: for time_step in range(4) do                                   ▷ 4 time steps in total
2:     if ¬time_step then                                         ▷ if it is the first time step
3:         weight_id ← 0
4:         for neuron_id in range(40) do                          ▷ loop through all hidden neurons
5:             MAC_cnt ← pull_offset(neuron_id)                   ▷ see how many MACs needed
6:             for mac_num in range(MAC_cnt) do                   ▷ loop through all needed MACs
7:                 w,a ← pull_WnA(weight_id)
8:                 Neuron_MAC(neuron_id,w,a)                      ▷ MAC in IF neuron
9:                 weight_id ← weight_id + 1
10:            end for
11:            if Neuron_fire(neuron_id) then                     ▷ if neuron fires with a spike
12:                Neuron_reset(neuron_id)
13:            end if
14:            output_V_diff(neuron_id)    ▷ output the neuron's voltage and its difference
15:        end for
16:    else
17:        for neuron_id in range(40) do
18:            load_neuron_state(neuron_id)                       ▷ load the voltage back
19:            Neuron_ACC(neuron_id,V_diff)                       ▷ accumulate V with V_diff
20:            if Neuron_fire(neuron_id) then
21:                Neuron_reset(neuron_id)
22:            end if
23:            output_V(neuron_id)                                ▷ output only neuron's voltage
24:        end for
25:    end if
26: end for
```

8.4 Output layer

The output layer, the consecutive layer following the hidden layer, operates on the incoming spike events or AER instead of input activation values. This results in a different micro-architecture design. What is more, this layer retains the highest accuracy from conversion by replacing the output spikes with membrane voltage accumulation. Therefore, we have the INF neuron that only accumulates but does not fire. What is more, since the output layer focuses more on the spike coming from a specific neuron instead of generating spikes, the computational flow should process all related neurons in the output layer that will receive this spike. This leads to the change in the weight matrix representation from CSR to CSC.

8.4.1 Slightly different architecture

This layer's schematic can be explained in Figure 8.10. The design consists of an index

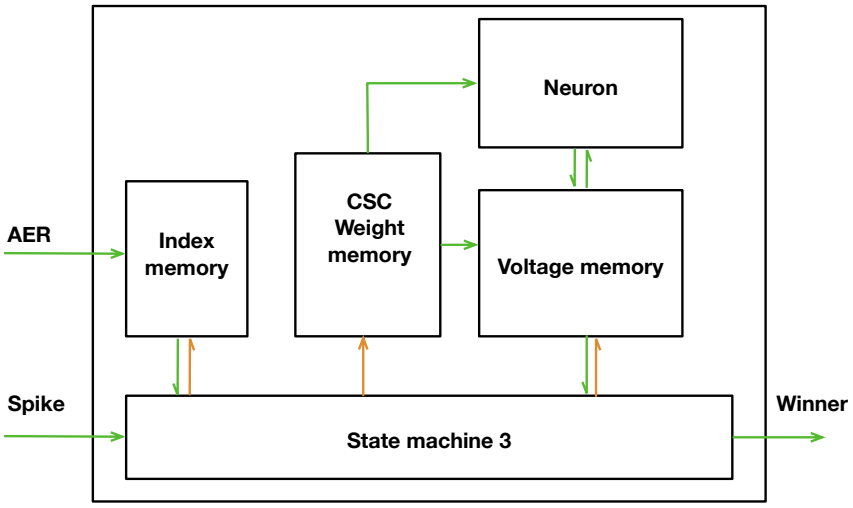


Figure 8.10: Output layer schematic diagram

memory, an INF neuron, a CSC weight memory, a voltage memory and a state machine. With the absence of an input value memory, the output layer module has a much simpler logic. But the memories are designed in a similar fashion, which will not be explained here.

Incoming spikes trigger the state machine to start the computation, which starts by checking how many neurons need updating from the output of the index memory. The index memory mainly records the CSC weight memory address. The neuron module is a simple INF neuron that only accumulates and outputs the voltage. By the end of the cycle, the state machine shall talk to the voltage memory and conclude with the winning neuron with the biggest membrane potential.

8.4.2 INF neuron

Integrate-and-no-fire neuron has a really simple data flow, all it needs is to load the voltage back in when the load signal is high and accumulate when input_valid signal is high. The schematic diagram of this neuron is shown in Figure 8.11. At the end of the accumulation,

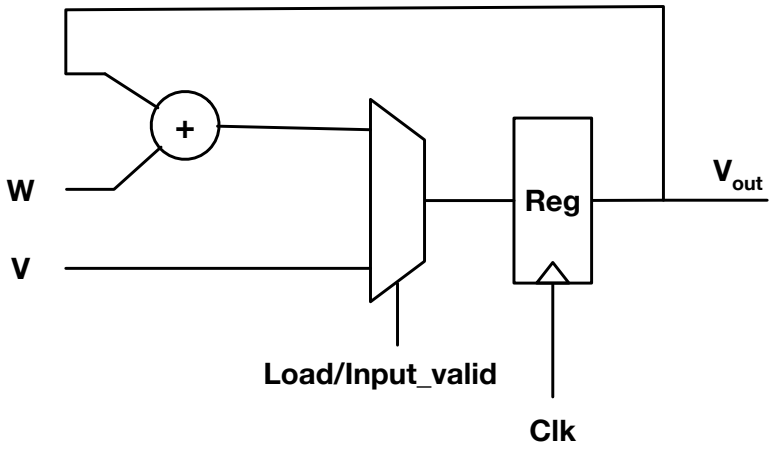


Figure 8.11: INF neuron schematic

the voltage shall be output back to the voltage memory.

8.4.3 State machine 3

The state machine in this layer can be seen as a simpler version of the last layer and it has the FSM chart depicted in Figure 8.12. The flow can be described in Algorithm 8. The state

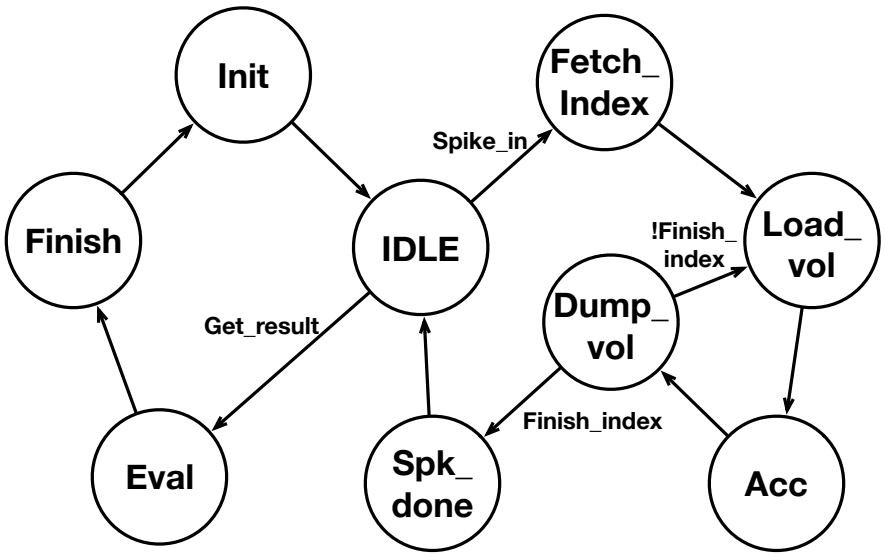


Figure 8.12: Output layer state machine chart

Algorithm 8 Spike-In and Winner Evaluation Algorithm

```
1: if spike_in then                                          ▷ Spike comes in
2:     begin_i, end_i ← fetch_index(AER)                     ▷ check all the neurons that need updates
3:     for w_addr in [begin_i, end_i] do                     ▷ loop through relevant weight memory
4:         neuron_id, w ← fetch_w(w_addr)                    ▷ get the corresponding neuron and weight
5:         load_neuron_state(neuron_id)
6:         Neuron_ACC(neuron_id, w)                          ▷ neuron accumulation
7:         output_V(neuron_id)                               ▷ export voltage
8:     end for
9: else if get_winner then                                   ▷ Evaluation stage starts
10:    max_V ← 0
11:    winner ← 0
12:    for neuron_id in range(18) do                         ▷ check all neurons' voltage
13:        V ← fetch_V(neuron_id)
14:        if V > max_V then                                 ▷ maximum voltage appears
15:            max_V ← V
16:            winner ← neuron_id                            ▷ update winner
17:        end if
18:    end for
19:    output_winner(winner)
20: end if
```

machine is in charge of two parts, one is to update the neuron when there is an incoming spike. It can find the related neuron ID by fetching the index from index memory. This will then give the relevant part of the memory to fetch the neuron ID and weight. This neuron ID will then be used to retrieve the voltage saved in the voltage memory and reload the neuron back to the state it was. After accumulation, the voltage will be exported back to the voltage memory.

Another part is evaluation, which is triggered by an external signal, ***get_winner***. Once the evaluation process starts, it will loop through the voltage values saved in the voltage memory and find the biggest voltage and its corresponding neuron. This will then be the winner of this inference.

8.5 Single net composition

With all three layers implemented, a single diagonal net can be simply connected by encapsulating all three modules. This data flow is demonstrated in Figure 8.13.

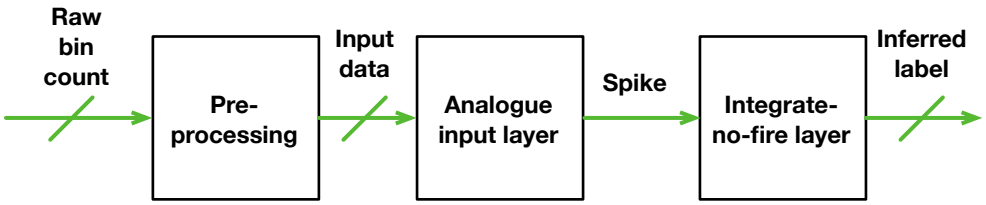


Figure 8.13: Single net construction

8.5.1 State machine 4

Simply connecting all three modules will not function without the proper controller giving control signals. The state machine for this top module design mainly coordinates the module by determining the stages the net should be. This involves the decision-making of when the computation should start, when should stop and how the final decision is exported.

The corresponding FSM chart is given in Figure 8.14 By default, the state machine should be

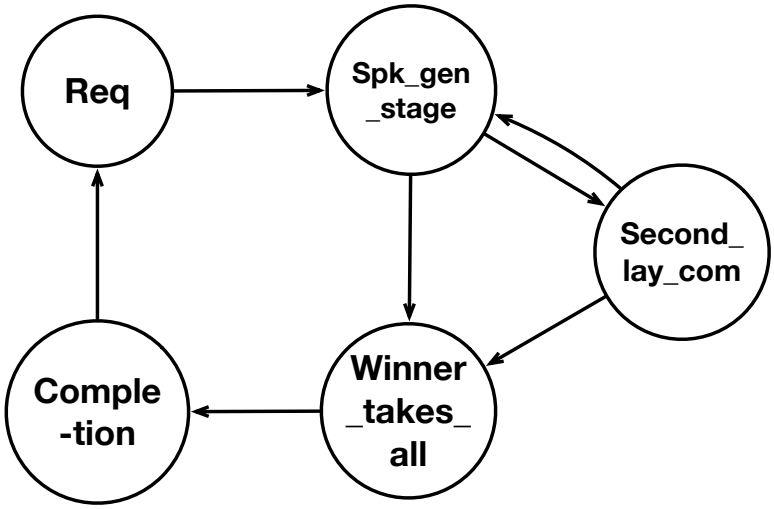


Figure 8.14: Single net FSM chart

at state ***Req***, which flags that the net is ready for another sample. This state will only transition to ***Spk_gen_stage*** when ***trans_start*** signal is present otherwise it will stay at ***Req***. In the meantime, this inter-layer state machine will observe the spike activity in between layers. If there is a spike generated from the hidden layer, the transition from ***Spk_gen_stage*** to ***Second_lay_com*** will be done. Otherwise, it will check if all time steps are done, which is a signal coming from the hidden layer. If ***Steps_done*** is present, the whole net is entering into the evaluation stage. As for the state ***Second_lay_com***, it is designed to make sure the generated spike is processed. Therefore, it can leap back to ***Spk_gen_stage*** when the spike is finished but the time steps are not finished. Or it can directly leap into the evaluation stage

when the spike is finished and all time steps are done. During ***Winner_takes_all*** state, the state machine outputs a ***get_winner*** flag to the output layer and waits until the inference is ready. ***Completion***, just as the name suggests, is a final state to indicate the inference for this sample is complete and outputs the inference result with the valid signal.

8.6 Ensemble net composition

The ensemble of this network should encapsulate each net and extra logic to make the final inference over all the decisions made by the ensemble. What is more, there should be another state machine to control the flow.

8.6.1 Architecture

The overall architecture of the ensemble is depicted in Figure 8.15. The whole process starts

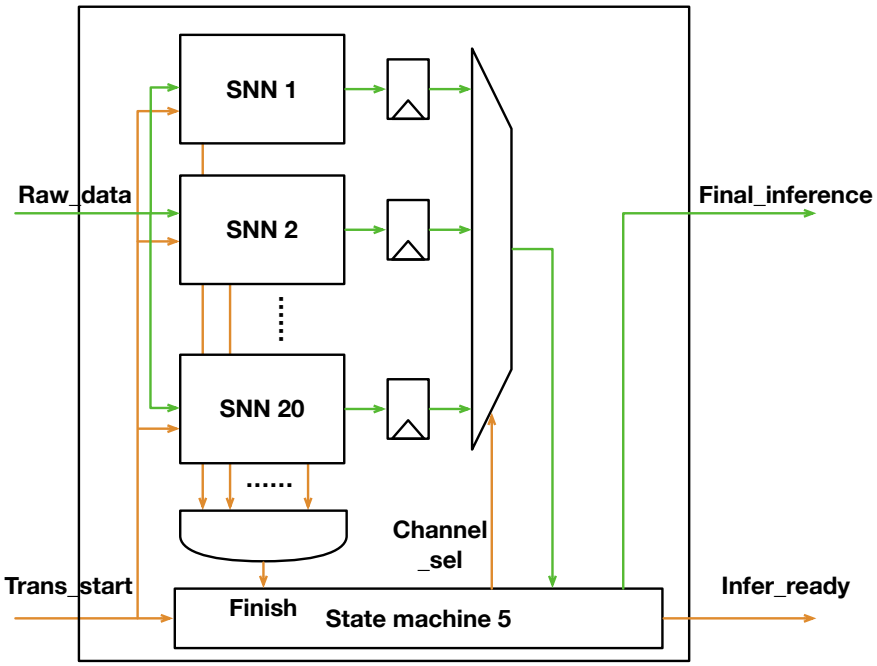


Figure 8.15: Bin-ratio ensemble SNN schematic diagram

with signal ***Trans_start***, which indicates the beginning of the raw data transfer. Each net might have a different processing speed. Therefore, the final judgement should only start when every net is finished. This is marked by the signal ***Finish***, which is the AND signal of all the individual finish signals. When all the nets are finished, the state machine shall do an evaluation over all the inferences from the ensemble. When the final ensemble evaluation is finished, the final inference shall be output together with the valid signal ***Infer_ready***.

8.6.2 State machine 5

The state machine for the ensemble is straightforward and simple. The FSM state machine chart is illustrated in Figure 8.16. It only has 4 different states, each state is self-explanatory.

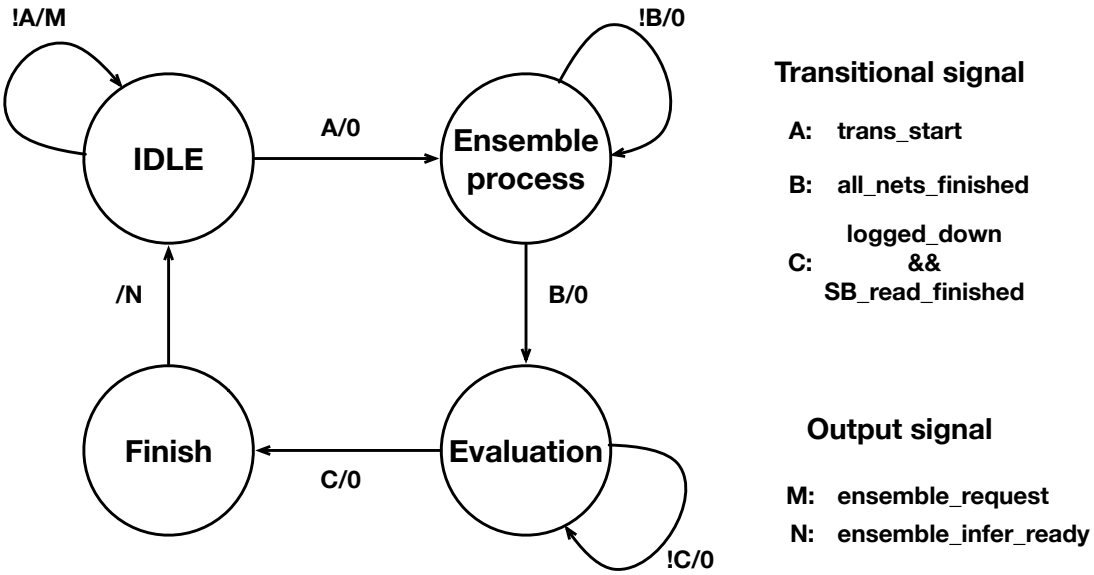


Figure 8.16: Ensemble FSM chart

All the transitional signals are fairly straightforward. There is a scoreboard inside the state machine, which logs down all the counts for different inferences from the ensemble. Once all the counting is finished through the ensemble, ***logged_down*** will be flagged up. This is followed by the maximum count checking for each class, which requires reading from the scoreboard. Once this is finished the flag for ***SB_read_finished*** will be raised. When both these two signals are up, the evaluation stage is finished. The ***Finish*** state signifies the completion of the whole process, during which the ready signal will be up with the output of the final ensemble inference.

8.6.3 Majority voting

A hard voting scheme was implemented because each net's parameters were scaled by different factors during quantisation. Their final output for the same class could not be straightforwardly summed up as it is supposed to be in soft major voting. The majority voting by the end of the ensemble simply collects the inferred label and logs it down to the scoreboard. This is followed by the counting of the poll for each class. The one that is getting the most votes will be the verdict for the whole ensemble.

8.7 External logic for testing

The hardware was implemented on an FPGA, results will be discussed in the next chapter. But this test should include an encapsulation that enables the BESNN to communicate with the external interface like UART, I2C or even PCIe.

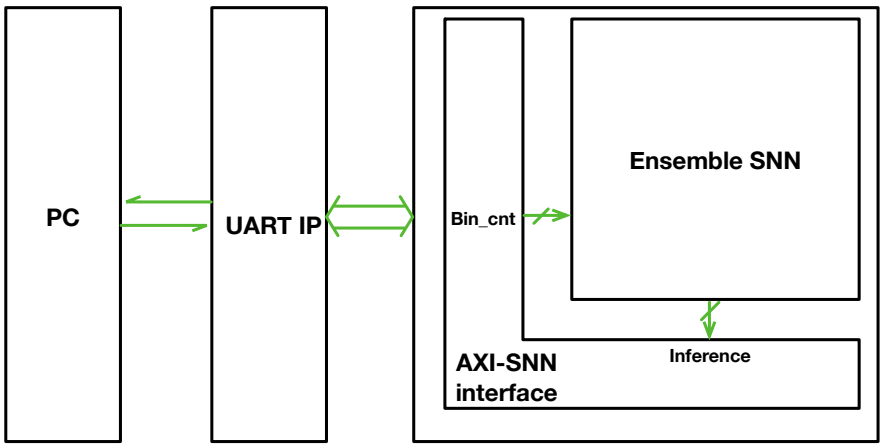


Figure 8.17: Validation system for BESNN

It is preferred to use a standardised interface in a hardware design. On Xilinx FPGA, the expected standard is AXI, Advanced eXtensible Interface. Therefore, an additional module to translate AXI to the SNN was implemented. Details for AXI interface and how it works are out of the scope of this work. The module ***AXI-SNN interface*** will be discussed here.

Details are illustrated in Figure 8.18. UART IP from Vivado design suite communicates

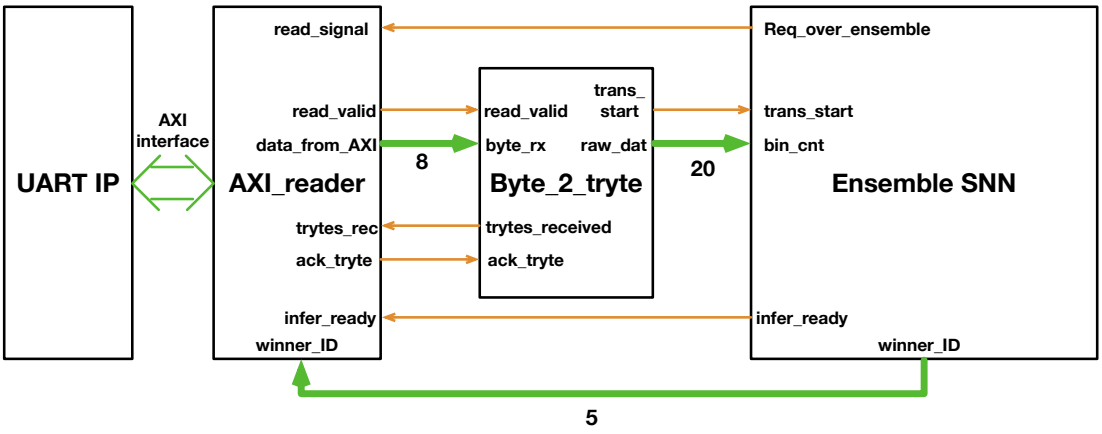


Figure 8.18: External logic needed for AXI interface encapsulation

through AXI interface. Therefore, an AXI reader module is needed to crack the AXI interface to get the data collected. This data is exported through port ***data_from_AXI*** with a valid signal. But this is an 8-bit data, which is not enough to represent the actual data range. What is needed is a 20-bit data, this has led to the need for a data organiser to convert 3 bytes

data into a 20-bit data and send them to the ensemble SNN. The communication between ***AXI_reader*** and ***Byte_2_tryte*** is also regulated with a pair of handshake signal ***trytes_rec*** and ***ack_tryte***. When ensemble SNN finished inference on the sample, it will export the winner ID together with a ready signal back to the module ***AXI_reader*** to be sent back to host through UART.

With the encapsulation of AXI-compatible interface to UART, ensemble SNN is ready to be validated with the help of a PC. One simply just needs to write a Python script to batch test all the samples automatically. These results can be returned back to PC for further analysis (accuracy, receiver operating characteristic curve). This also proves that the proposed ensemble module can be incorporated into a bigger design, e.g. as a peripheral for a system-on-chip (SoC) design.

8.8 Conclusion

This chapter explained the whole ensemble SNN hardware implementation from a top-down hierarchy. Details for each layer and module are presented with figures and specifications. Algorithms have all been converted into equivalent hardware on FPGA. Tests and results analysis will be discussed in the next chapter where the accuracy, power analysis and resource utilisation shall be given.

Chapter 9

Experiments, results and analysis

With both the software and hardware established, tests and validations are needed to verify the performance and other metrics. Here we shall present all the tests that were done and the related results.

9.1 Algorithm evaluation

Since the algorithm proposed in chapter 7 is an SNN-based ensemble network, the proper tool for SNN simulation is needed. Here we used the PyTorch-based simulator SpikingJelly (Fang et al., 2020), a tool developed by Peking University. This has proven to be efficient and flexible. It also supports hardware acceleration on various platforms.

9.1.1 SpikingJelly simulation

Hardware acceleration

Here, the simulation was performed on a Macbook Pro with an M1 chip and 16 GB RAM. Since the simulator was developed based on PyTorch, it can actually employ Apple's GPU API Metal. To use this option, one simply just needs to set the simulation device as Metal Performance Shaders (MPS) before running the simulation.

```
1    device = torch.device("mps" if torch.backends.mps.is_available()
    else "cpu") ## set the backend as "MPS" if available otherwise it
    will be CPU
```

Listing 9.1: Simulation configuration on MPS

One can also use other backends like Nvidia GPU's CUDA.

Pre-charge before simulation

One of the techniques to eliminate the error and speed up the simulation is to pre-charge the membrane potential to half of its threshold. This can be simply applied by making a small change to the simulation process. Normally, the spiking neurons need to be reset before simulation to clear up the voltage residual to start from 0. To pre-charge the neuron, a line of code can be inserted after the reset as follows:

```
1    if not precharge:  ## no pre-charge
2      functional.reset_net(model)
3    else:
4      for m in model.modules(): ## pre-charge
5        if hasattr(m, 'reset'): ## if neurons can be reset
6          m.reset()
7          m.v = 0.5 * m.v_threshold ## set the voltage to half of the
   threshold
```

Listing 9.2: Turn on the pre-charge

In the code above, *precharge* is an option for a simulation function to be checked before the simulation. *model* is a constructed SNN object using SpikingJelly. By looping through the modules in this object, one is checking the constitutional units. This can be a neuron model or a single layer. Generally, only the spiking neuron has the attribute of "reset", the voltage is then pre-charged by the code at line 7.

INF neuron

Unlike commonly used IF neurons, there is no supported class for INF neurons. To simulate the proposed ensemble SNN, a class called "INoFire" was created to suit the needs.

```
1    class INoFire(neuron.BaseNode):
2
3      def neuronal_charge(self, x: torch.Tensor):
4          self.v += x
5          return self.v
6
7      def single_step_forward(self, x: torch.Tensor):
8          self.v_float_to_tensor(x)
9          self.neuronal_charge(x)
10         spike = torch.zeros_like(x)
11         return spike
```

Listing 9.3: Integrate-and-no-fire neuron class

Since SpikingJelly does not directly work on the matrix multiplication but inserts a spiking neuron node after, the class defined here shall simply take in the matrix multiplication product into the accumulation and decide if a spike should be generated. Our INF neuron class inherits the base node class and makes slight changes to the original methods. Two essential methods are defined: ***neuronal_charge*** and ***single_step_forward***. The code itself is self-explanatory, but because we are not generating spikes for this class, the output of the method ***single_step_forward*** is a zero vector that has the shape of x.

Evaluation using INF neuron

The class for an individual SNN net is defined in the order:

- Linear layer (input_size*40)
- IF node
- Linear layer (40*18)
- INF node

After simulation, the voltage of the INF node shall be exported and evaluated to have conclude the inference.

```
class individual_SNN_4_ensemble_no_fire(nn.Module):
    """
    This defines the simple SNN model for the ensemble model, it takes
    the neuron that does not fire as the output layer
    """

    def __init__(self, input_size, threshold: list, use_bias=False):
        super(individual_SNN_4_ensemble_no_fire, self).__init__()
        self.each_individual = nn.Sequential(
            layer.Linear(input_size, 40, bias=use_bias),
            neuron.IFNode(v_threshold=threshold[0], v_reset=None),
            layer.Linear(40, 18, bias=use_bias)
        )
        self.output_lay = INoFire()
```

9.1.2 Results

Floating point BESNN

Following the proposed LSQ conversion scheme, all the weights are saved in the universal 32-bit floating point numbers. The activation bit-width chosen for this work is 2-bit. This should consequently lead to the maximum accuracy at the time step of 4 because all the information needs 2^2 time steps to be passed over. This has proven to be true with the results collected from SpikingJelly simulation. The accuracy against the simulation time step for the trained full precision BESNN is presented in Figure 9.1. It can be seen that the accuracy peaks at the

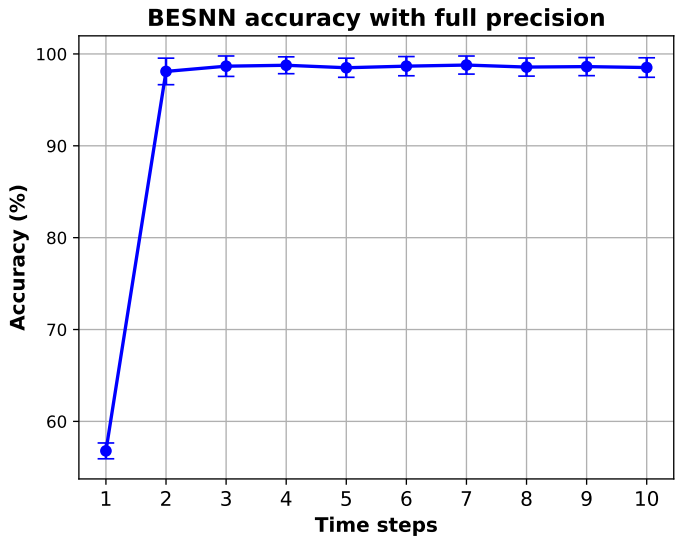


Figure 9.1: Full precision BESNN accuracy against time step

4th time step with 97.48%. Theoretically, the accuracy should stay the same after time step 4. But the accuracy fluctuates around 97% in the real simulation, this is because there are still spikes being generated which might distract the ensemble net to make correct inferences. It can be seen from the figure that the performance of the network overall has been stable with a narrow error range. If there is a restriction set for spiking neurons to have an upper limit of total spikes generated, the accuracy should stay the same.

Quantised BESNN

Eventually, simulation time step 4 is set as the final configuration. The quantisation procedure was explained in chapter 7, but different quantised bits against accuracy is depicted again in Figure 9.2. The experimented quantisation bit-width ranges from 1 to 16, it can be seen that the original full accuracy can be retained with only a 10-bit representation. However, 8-bit is a more commonly used bit-width, which carries a slightly lower accuracy of 97.04%. It can also be seen that the accuracy remains unchanged for any bit-width bigger than 9, where full precision accuracy was reached at 97.48%.

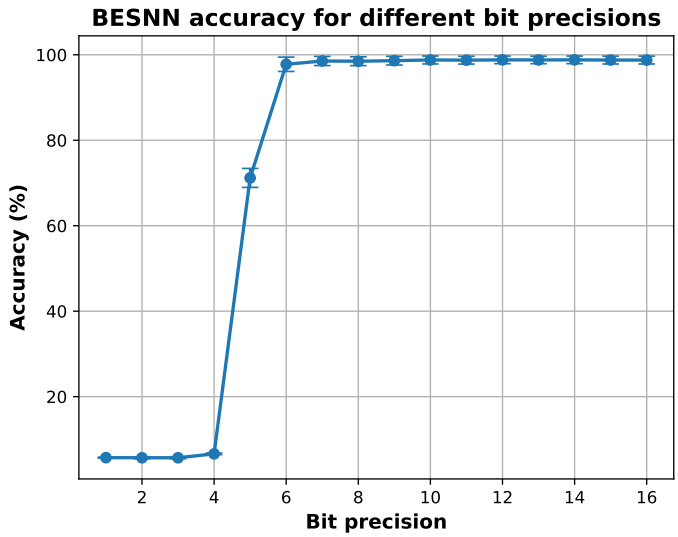


Figure 9.2: Accuracy against bit precision on BESNN with 4 time steps

9.2 FPGA evaluation

9.2.1 FPGA platform

The target FPGA platform chosen for the implementation is Xilinx VCU108 (Shown in Figure 9.3), which is an off-the-shelf high-end FPGA board from Xilinx. It belongs to the Virtex™ UltraScale™ family, which is fabricated in an advanced 20nm technology node. As for the evaluation board itself, it is equipped with a high capacity of logic units and an onboard power measurement chip. To test the performance of the implemented FPGA design, a validation system was constructed with the design unit integrated. Details about the external testing logic have been explained in the last chapter.

This FPGA offers a huge volume of resources and highly extensible ports and interfaces. The core FPGA chip is **XCVJ095-2FFVA2104E**, and all the resources it has is listed in Table 9.1.

Specification			
Logic Cells (K)	DSP Slices	Memory (Mb)	I/O Pins
1,176	768	60.8	832

Table 9.1: System Specifications of VCU108

The universal asynchronous receiver and transmitter (UART) IP in Vivado design suite is compatible with advanced extensible interface (AXI), therefore, the top encapsulation included an interface to communicate through this protocol. The UART IP was configured to have a baud rate of 115,200, and the data generation and receiving on PC has been done with the help of the Python package PySerial (Liechti, 2015).

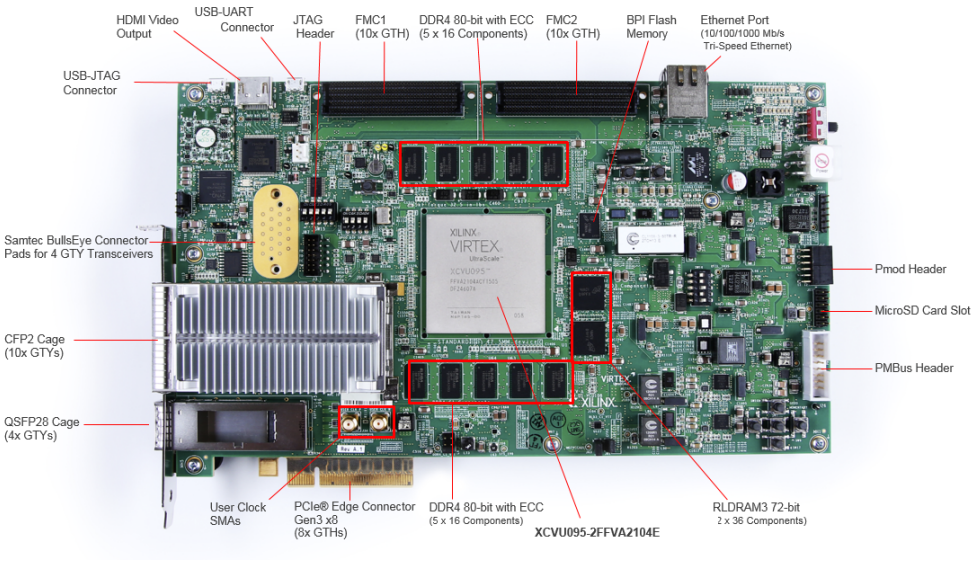


Figure 9.3: Xilinx VCU108 FPGA board (Picture available from (AMD, 2023))

The hardware validation on this board has proven the design is consistent with the SpikingJelly simulation, where the accuracy remains the same at 97.04%. This is only 0.44% lower than the full precision BESNN. All the correct and incorrect labels are also the same as that of the software, even the final membrane potentials for all the neurons have been confirmed the same. The confusion matrix can be found in Figure 9.4.

The overall accuracy comparison across different models are presented in Figure 9.5. In the figure, 5 different implementations are presented. The original bin-ratio BEANN proves to have an accuracy of 96.4%, this has been outperformed by all the models proposed by us. The best-performing model is the full precision BEANN with logarithm and subtraction as the pre-processing replacement of bin-ratio with an accuracy of 97.8%. After pruning, this BEANN still shows strong performance of 97.1% with only 30% parameters left. This is later converted into a full precision SNN with a boosted accuracy after conversion at 97.5%. Finally, our FPGA shows a slight accuracy drop from the full precision at 97.0%.

9.2.2 Resource utilisation

The resource analysis can be found in Table 9.2. Most of the resource was taken by look-up

Table 9.2: Resource utilisation on VCU108

	LUT	LUTRAM	FF	BRAM	DSP
Utilisation	61,822	1,050	9,841	10	0
Utilisation %	11.50	1.37	0.92	0.58	0

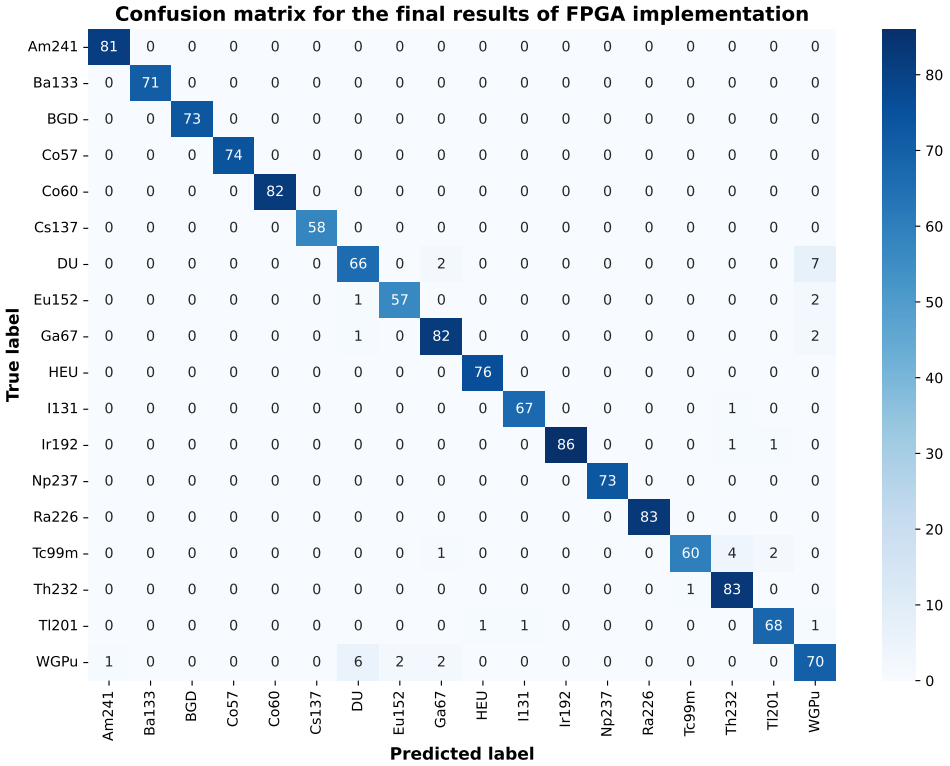


Figure 9.4: Confusion matrix for final FPGA implementation

tables (LUTs) in the analogue input layer, this is because there is big counter logic to switch the memory address frequently.

9.2.3 Power analysis

The post-implementation power analysis tool used a switching activity interchange format (SAIF) file to give a fairly accurate power estimation. The file itself logs down the number of switching activities in a prescribed time. Incorporating this file into our analysis, the total on-chip power is 1.215 W. In this figure, the static power takes up 75% and dynamic power constitutes 25%, i.e. 305 mW. According to Kromek, the original solution was a software-based implementation on an ARM core, which cost nearly 2 W (X. Huang, Jones, Zhang, Xie, et al., 2020).

The actual power efficiency computation shall be based on the total on-chip consumption, which is 1.215 W. On the other hand, our hardware runs at 100 MHz, which would give us the latency for one inference at 0.334 ms according to the simulation. The throughput is thus 2.994×10^3 inferences per second ($inference/s$). Under such an assumption, the energy

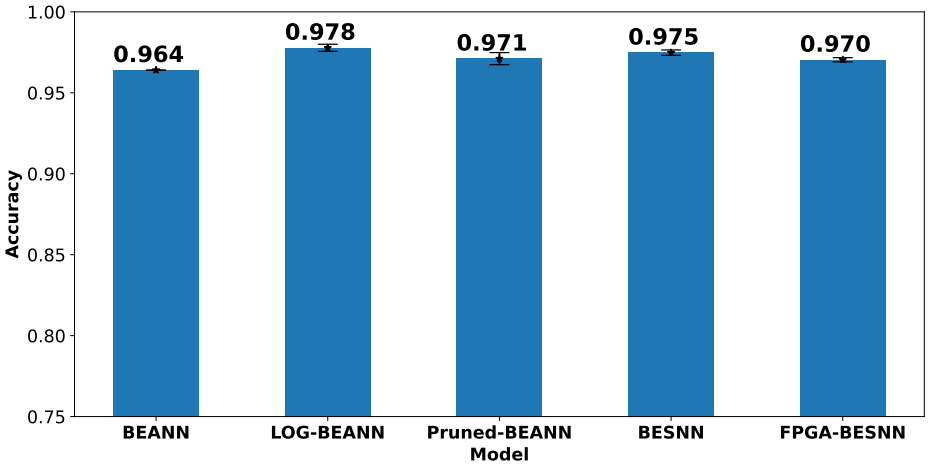


Figure 9.5: Accuracy across different models

efficiency could be calculated:

$$\frac{2.994 \times 10^3}{1215} = 2.464\ inference/(s \cdot mW)$$

To put it differently, each inference burns 406 μJ. On average, the completion of each sample requires 259,720 operations, this means $7.78 \times 10^8\ OP/s$, i.e. 778 MOP/s. Energy efficiency in terms of operations would be 0.64 $MOP/(s \cdot mW)$.

Our design appears to be not the most power-efficient because of the high parallelism. Besides, most of the power was consumed by static leakage and FPGA resources that were not used. The static power is more related to the technology node and the size of the die, given that our design takes only around 12% of the total resource, it may be unfair to use the static power from the estimation to characterise our solution. Presumably, if we use 12% of the total static power as the part consumed by our implementation. This would give us in total of $305 + 910 \times 0.12 = 415\ mW$, which is much lower than 1.215 W but closer to the actual case. Therefore, this design can probably be mapped to a smaller board to reduce power consumption from the static power.

9.2.4 Other boards implementation

The attempts to map to other available FPGA parts have been made. The implementation does not require the actual board to finish, one can simply use the tool to simulate the post-implementation behaviour and signal switching activities. Therefore, one can use the same design to map to other FPGA to estimate the power consumption without actually running on the board.

Here we implemented the same design into different boards of different families, which have different technology node for comparison. The selection of each board within a family is aimed at identifying the most compact board that possesses adequate resources except VCU108. The results are listed in Table 9.3 and 9.4.

Table 9.3: Resource Utilisation for Different Boards

Family	Node (nm)	FPGA part	Resource utilisation			
			LUT	LUTRAM	FF	BRAM
Virtex UltraScale	20	xcvu095-ffva2104-2-e (**VCU108**)	61,822	1,050	9,841	10
			11.50%	1.37%	0.92%	0.58%
Kintex UltraScale	20	xcku025-ffva1156-1-c	61,839	1,050	9,841	10
			42.52%	1.55%	3.38%	2.78%
Kintex-7	28	xc7k160tfbg484-3	60,374	715	8,837	20
			59.54%	2.04%	4.36%	6.15%
Spartan-7	28	xc7s100fgga484-2	60,369	715	8,837	20
			94.33%	4.06%	6.90%	16.67%
Artix-7	28	xc7a100tcsg324-3	60,392	715	8,837	20
			95.26%	3.76%	6.97%	14.81%

Table 9.4: Power Consumption for Different Boards

FPGA part	Power (mW)		
	Dynamic	Static	Total
xcvu095-ffva2104-2-e (**VCU108**)	305	910	1215
xcku025-ffva1156-1-c	298	480	778
xc7k160tfbg484-3	389	113	502
xc7s100fgga484-2	393	92	485
xc7a100tcsg324-3	386	86	472

From the table it can be seen that even the same design can cost differently using different boards. This is related to more than just the resource but also the technology node it is using. When using UltraScale™ series, the resource usage might be slightly higher in terms of look-up-tables (LUT), LUTRAM and FF, but the overall dynamic power taken by these are significant lower thanks to the newer technology node. But because of the big amount of resource on chip, the static power of UltraScale™ series are much higher. These result in a higher total power consumption when combining dynamic and static power dissipation. So far the lowest power consumption is by Artix-7™ board, whose LUT can be used up to 95.26%, this implementation only costs 472 mW, which is almost 1/3 of the original VCU108 implementation. This means that the energy efficiency is now:

$$\frac{2.994 \times 10^3}{472} = 6.343\ inference/(s \cdot mW)$$

And each inference needs 157μJ, energy efficiency for operations is 1.65 $MOP/(s \cdot mW)$.

If one can choose the newest technology node with just enough resource on board, the best power trade-off can be made to have both the lowest dynamic and static power.

9.3 A smaller ensemble?

9.3.1 Exhaustive search

An exhaustive search was carried out to find the best combination of ensembles given a certain ensemble size. This size ranges from 1 to 20, and the accuracy distribution in a given size along sizes is illustrated in Figure 9.6. For sizes 1 and 19, of course, there are only 20

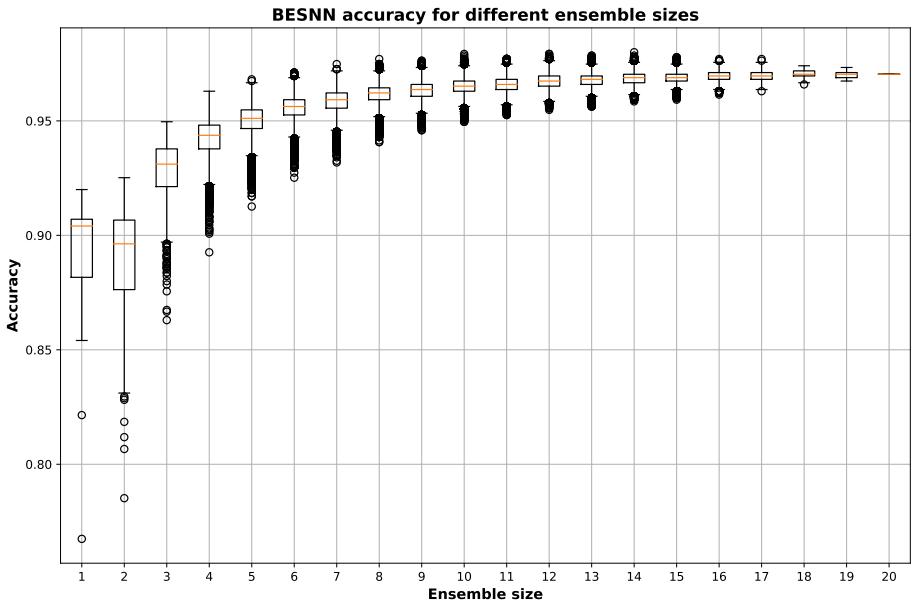


Figure 9.6: Accuracy distribution box plot for different ensemble sizes

combinations. As for the size of 20, there is simply one combination, which should yield an accuracy of 97.04%.

From the box plot figure, it can be observed that the mean accuracy increases along with the ensemble size. However, this increase will slow down after size 7 or 8. This suggests that the accuracy increase brought by the ensemble size is not always significant. It can be expected to bring a decent accuracy boost when the ensemble size increases from 2 to 5 where the lowest accuracy for size 5 is bigger than the mean accuracy of size 2. But when size increases from 5 to 10, the mean accuracy rise is only around 1%, but this is double the ensemble size. Besides, the accuracy outliers in the figure (black circles) show the possibility of how high

and low the accuracy of this ensemble size can bring. The maximum accuracy a combination can bring after size 5 statistically remains unchanged at around 97%. Therefore, with the right combination, an ensemble size of 5 can still produce a maximum accuracy of 96.81%, which is only 0.23% lower than the ensemble size of 20.

9.3.2 What does a smaller ensemble cost?

Test on the best-performing ensemble of 5 was performed to see how much resource it needs and how much power we can save if the smaller design is implemented. Here we would only use the VCU108 board and the smallest possible Artix-7™ board for comparison.

The resource utilisation of these two re-implementation is presented in Table 9.5 The board chosen for Artix-7 has the total LUT of 20,800, LUTRAM of 9,600, FF of 41,600 and BRAM of 50.

Table 9.5: Resource Utilisation for Ensemble of 5

Family	FPGA part	Resource utilisation			
		LUT	LUTRAM	FF	BRAM
Virtex UltraScale	xcvu095-ffva2104-2-e (VCU108)	15,308	265	2,537	2.5
		2.85%	0.35%	0.24%	0.14%
Artix-7	xc7a35tcpg236-3	14,946	182	2,282	5
		71.86%	1.90%	5.49%	10.00%

On the other hand, the power consumption is listed in Table 9.6.

Table 9.6: Power Consumption

Family	FPGA part	Power (mW)		
		Dynamic	Static	Total
Virtex UltraScale	xcvu095-ffva2104-2-e (VCU108)	75	907	982
Artix-7	xc7a35tcpg236-3	98	69	167

This means that the smallest board featuring an ensemble of 5 BESNN only costs 167 mW. With the overall performance of 96.81%, this is a really energy efficient design. The corresponding power efficiency is:

$$\frac{2.994 \times 10^3}{167} = 17.928\ inference/(s \cdot mW)$$

Each inference only costs 56 μJ, the efficiency in terms of operations is 1.16 $MOP/(s \cdot mW)$.

9.4 Conclusion

Overall in this chapter, we discussed the testing methods on both software and hardware in detail. The construction of the nets for testing in SpikingJelly was shown with the essential code presented. Towards the software solution, we demonstrated the accuracy for the trained BESNN with different simulation configurations including time steps and bit precision.

Then on the hardware implementation, an 8-bit precision implementation was chosen and tested. This test was automated using UART port and PySerial Python package. The test result collected from FPGA shows consistency with the software simulation in terms of the BESNN performance.

This hardware implementation is characterised by its resource utilisation and power consumption. Despite that it is not highly energy efficient with a total on-chip power of 1215 mW, this is due to the high static power dissipation by the board itself. There is still a lot of unused resource on the board that is taking up most of the power. This has been validated by exporting the design to a smaller board that can be used to maximise the resource use. By far, the lowest total power consumption is 472 mW for an ensemble of 20 BESNN.

To push this even further, one can trimmed the ensemble to a smaller number of nets and implement that into a smaller board. An example of 5 nets was done where an exhaustive search showed the best combination of 5 nets to be mapped. This retained an accuracy of 96.81%, which is only 0.24% lower than the ensemble of 20. In the meantime, it can be mapped to an Artix-7™ board that only costs 167 mW.

When compared to the existing solution that runs on the handheld RIID device, which has a power consumption of an average of 2 W, our ensemble of 5 nets implementation possesses better than an order of magnitude improvement in power consumption, this is only 0.167 W.

Since this algorithm is based on an ensemble of artificial neural network, it is also possible to be mapped to a micro-controller to deliver the task. Similar process may also be needed including individual network training, trimming to a smaller ensemble and quantisation. TensorFlow lite is a well developed framework suitable for this Tiny ML project (Warden & Situnayake, 2020). Tiny ML suggests tiny machine learning in this context. So far in the market there are multiple choices available in terms of developing board that supports Tiny ML project. For example Arduino Nano 33 BLE sense is a good platform that has already been used for machine learning tasks (Waqar, Gunawan, Morshidi, & Kartiwi, 2021) (V et al., 2022). This might serve as a future direction for this specific ensemble neural network and make the application more accessible to general developers.

Chapter 10

Conclusion

10.1 Summary of findings

This is demanded by a real application case where longer battery life is needed for a handheld device to operate RIID task. By the nature of scintillator, an event-based processing framework was proposed. The general idea of computing by events has the potential to be a promising candidate to save power. One of the representative event-based computing paradigms, spiking neural networks (SNN), demonstrates comparable computing performance to deep learning. This has inspired us to combine SNN, an event-based processing methodology, with RIID for the power optimisation on the proposed task. Here, we present some key findings made during this research under the big project.

In Summary, here lists a table to compare all these different implementations we did. The dataset column refers to the datasets defined in chapter 4.

Project	**Architecture**	**Dataset**	**Accuracy**	**Platform**	**Power**
Simple SNN	3-layer fully connected SNN 100-FC40-FC8	Simple re-binned dataset	87.61%	Xilinx Artix-7 FPGA	72 mW
CSNN	1024-4CONV-AVGPool-FC8	CLLBC synthetic dataset	90.62%	Xilinx Artix-7 FPGA	75 mW
Unsupervised SNN	3-layer recurrent winner-take-all structure	Simple re-binned dataset	71.55%	4-node SpiNNaker board	N/A
BESNN	20 ensemble of simple 3-layer SNN, each SNN: Input-FC40-FC18	Enhanced CLLBC synthetic dataset	97.04%	Xilinx VCU108 FPGA	1.215 W

Table 10.1: Comparison of different SNN projects.

10.1.1 Simple multi-layer SNN implementation

This project I involved in implemented a 3 layer simple SNN that has the architecture of 100-FC40-FC8 (FC means fully connection) on an FPGA for RIID on 8 different isotopes.

My contribution

I was mainly in charge of final FPGA implementation, testing and debugging, where I cooperated with another colleague. This is the first published work made during my PhD.

Accuracy

Overall accuracy under 3 seconds simulation on FPGA is 87.61%, this requires around 1500 input spikes with the input frequency of 500 Hz. This is 1.3% lower than the corresponding ANN algorithm.

Power

With the main clock running at 100 MHz, the total on-board power consumption is 72 mW, where static power contributes 70 mW. This design has a really low power consumption because there is no pre-processing in the design.

10.1.2 Convolutional SNN (CSNN) implementation

This project implemented a convolutional SNN (CSNN) with one covolutional layer (4 kernel with the size of 5 and stride of 1), one pooling layer (kernel size of 16) and a fully connection layer for RIID task for 8 isotopes.

My contribution

In this project, I proposed and implemented the automated testing methodology for our design, where the testing module can be interfaced through PC with an additional module. To do the testing, I drafted the python scripts for testing vectors generation. Besides, I also drafted all the python scripts for testing and visualisation of final results. Finally, I reported all the testing methods used in our published paper.

Accuracy

The overall accuracy is slightly higher at 90.62%, but this is achieved by extending the simulation time to 60 seconds. The worst performing classes are $^{226}\mathrm{Ra}$ and $^{232}\mathrm{Th}$. This is because $^{226}\mathrm{Ra}$ highly resembles $^{133}\mathrm{Ba}$ and therefore got mistaken. As for $^{232}\mathrm{Th}$, it does not show strong radioactivity within 3 seconds.

Power

With the main clock running at 100 MHz, the total on-chip power consumption is 75 mW, in which dynamic power constitutes 5 mW.

10.1.3 Unsupervised STDP implementation on SpiNNaker

This implementation on SpiNNaker employs unsupervised spike-time dependent plasticity (STDP) for SNN learning to achieve the RIID task on 8 isotopes. This shares the same data set as simple multi-layer SNN implementation.

My contribution

I proposed and implemented the whole algorithm on SpiNNaker. This implementation has also been trained and optimised by me. The following testing work and experiments design were operated by me. In addition, I also drafted the final paper and data visualisation.

Accuracy

This approach can only achieve 72% of accuracy for testing, and at most at 80% during training. This is because unsupervised SNN is not good at classifying highly resembling pattern. From the result, it can be observed that it also struggles with the class $^{226}\mathrm{Ra}$ and $^{232}\mathrm{Th}$ for the similar reason as CSNN implementation. As for $^{60}\mathrm{Co}$, it is due to the defect from pre-processing.

Power

There is no hardware implementation for this work for the power consumption measurement. The design was implemented on a 4-node SpiNNaker board with a power supply of 5 V, when idle, the card draws about 400 mA but peaks at around 1 A (*SpiNNaker Technical Documents*, n.d.) .

10.1.4 Bin-ratio ensemble SNN (BESNN) implementation

This implementation redesigns the original bin-ratio ensemble ANN (BEANN) into its spiking version and deploys this algorithm into FPGA for a more challenging RIID task of 18 radioisotopes with added background and gain shift noise.

My contribution

I started with idea conceptualisation and carried on with formal analysis. Acting as the main investigator, I proposed the methodology, drafted the software, implemented the hardware and finally operated validation. This is followed by visualisation of the results and original draft writing. Finally I was also responsible for the revision and editing for the publication.

Accuracy

The ensemble of 20 SNN presents a high accuracy at 97.45% at full precision. When quantised into 8 bit for FPGA implementation, this 20 nets ensemble still demonstrates 97.04%. This is even higher than the original bin-ratio ensemble ANN algorithm. Strong resilience was shown in this algorithm with a trimmed 5-net ensemble still performing at 96.81%.

Power

Different choices of FPGA should be made to host this design for the optimal performance/watt. The targeted FPGA VCU108 with 20 nm technology shows total on-chip power of 1,215 mW where dynamic power only takes 305 mW. Under such condition, each inference should cost 106 μJ. For best performance on FPGA, an Artix-7™ board can be probably chosen for total on-chip power of 472 mW and single inference cost of 157 μJ. For a further power-saving option, a smaller Artix-7™ board can be used to host an ensemble of 5. This will only cost 167 mW in total, with a single inference burning 56 μJ.

10.2 Achievement of Objectives

Various routes of employing SNNs have been attempted because of the existing training techniques for SNNs. We have shown the proof of concept for event-based processing on RIID using ANN-SNN conversion methods (X. Huang, Jones, Zhang, Furber, et al., 2020). This is later solidified by an actual multi-layer SNN implementation on field programmable gate array (FPGA) (X. Huang, Jones, Zhang, Xie, et al., 2020) ,which suggests a great potential with an overall accuracy of 87.61% at 8 different radioisotopes and a power consumption of 72 mW. A further step was taken by extending the simple multi-layer SNN into a convolutional spiking neural network (CSNN) implementation on FPGA (X. Huang et al., 2021). This work demonstrated an improved accuracy towards the task of 90.62% and a power dissipation of 75 mW.

In the mean time, other ways of employing SNNs are explored. A more bio-plausible training approach was applied to SNNs in an unsupervised fashion for RIID (Xie et al., 2022), which is an fully event-driven implementation on SpiNNaker, a SNN emulation platform. To achieve this, detailed parameter search, data pre-processing and other techniques were employed

including neural homeostasis, filtering and minibatch processing. Contrary to the ANN-SNN conversion, this spike-time dependent plasticity (STDP) based SNN learned the data feature in an unsupervised way, which does not require data labelling. The learning process occurs within a fully event-based paradigm. This is desirable because it utilises temporal information a spike carries, and it is highly compatible with SNNs. The classification on 8 different radioisotopes could be achieved with an accuracy of 72% with this unsupervised network.

To improve the accuracy and possibly drop the power cost, a bin-ratio ensemble SNN (BE-SNN) was proposed and implemented on FPGA. From algorithm development to the deployment of this neural network on Virtex Ultrascale VCU108, endeavour was made to optimise the network architecture, pre-processing and computational cost. We initiated the process with an equivalent substitution of 'bin-ratio' using the base-2 logarithm and subtraction, a modification that enhances its suitability for hardware implementation. In the meantime, this pre-processing reduces operations needed and speeds up the training convergence. To simplify the implementation and reduce the number of parameters needed, pruning was applied to reduce the total parameters down to 30%. Once the sparse ANN models were defined, they were transformed into SNNs using 2-bit quantisation aware training namely LSQ where the threshold values were derived from the learned step size. Subsequently, they were further quantised into 8-bit networks for the fixed-point calculation on an FPGA. The hardware design consists of 20 nets, a top-level state machine and logic for majority voting. Inside each net, we have the on-chip pre-processing, analogue input layer and integrate-no-fire layer. The performance was validated using UART IP from Vivado and it presented high accuracy on the classification task for 18 different radioisotopes at 97.04%, which is just 0.44% lower from the full-precision SNN but surprisingly 0.64% higher than the original bin-ratio ensemble ANN. In terms of power consumption, the proposed design costs approximately $406\ \mu J$ per inference with a throughput of 2.994×10^3 inferences per second. This gives us the energy efficiency of $2.464\ inference/(s \cdot mW)$ or $0.64\ MOP/(s \cdot mW)$ in terms of operations.

This implementation may seem not highly power-efficient, this is due to high parallelism of our design and the high static power consumed by the FPGA board given that the resource utilisation overall is only around 12%. To prove that, our design was mapped to a smaller board to avoid resource waste. Another 4 boards from different families were chosen based on the requirement that the board must be the smallest possible board with adequate resources to host the implementation. After comparison, we showed that the smallest possible board we can find provides a total on-chip power with only $472\ mW$, which is made of dynamic power of $386\ mW$ and static power of $86\ mW$. This aligns with the expectation where the board costs higher dynamic power but a much lower static power due to a much smaller FPGA chip with a less advanced technology node. This updates our energy efficiency with $6.343\ inference/(s \cdot mW)$ and energy consumption drop to $157\ \mu J$.

If saving power is the top priority, a trimmed ensemble of SNN can be implemented. We have run an exhaustive search for the accuracy against ensemble size and realised that this algorithm is resilient to sparsification. The best accuracy of ensemble size of 5 was 96.81%, which is only 0.24% lower than the ensemble of 20. However, the potential power saving is significant. With the smallest possible Artix-7™ board, we have shown that the total on-chip power can be compressed to $167\ mW$, this yields an energy efficiency of $2.464\ inference/(s \cdot mW)$. In other words, each inference only burns $56\ \mu J$.

By comparison to the target, which is $2\ W$, our design has proven to be a strong alternative solution in terms of both performance and power saving. The ultimate implementation demonstrates a significant improvement in power consumption of more than tenfold.

10.3 Contribution to knowledge

Overall, this project is the first of its kind to use event-based processing algorithm for radioisotope identification task. Its breakthrough lies in the establishment of the whole application flow employing SNN using hardware. This has demonstrated the possibility and performance of SNN in this specific application domain.

As for the unsupervised STDP implementation, it is the first unsupervised STDP based SNN algorithm for RIID by our knowledge. This presents the capability of local self learning rule for a more complex task other than hand written digits recognition.

As for the bin-ratio ensemble project, it pushes the accuracy boundaries with an excellent performance. In the mean time, it is also the first ensemble SNN application on RIID by our knowledge. What's more, an advanced ANN-SNN conversion method was complemented with pruning to reduce computational overhead. This is by far the first attempt to apply iterative pruning in SNN conversion. Last but not least, this research work provided a detailed application flow for sparse SNN construction and characterised the hardware implementation with its power and energy cost.

10.4 Limitations

10.4.1 Unsupervised STDP implementation

As for unsupervised STDP implementation, it has the following drawbacks:

- Accuracy is discernibly lower than ANN conversion
- Training costs much more resources and longer even with less training overhead
- The network architecture is not suitable for hardware implementation

Firstly, when compared to other implementations utilising the same dataset, this approach achieves only 72% accuracy, despite incorporating pre-processing steps prior to the unsupervised network. Conversely, the testing accuracy reported in (X. Huang, Jones, Zhang, Xie, et al., 2020) approaches nearly 90%. Secondly, despite employing accelerated training techniques, the training duration for this unsupervised network still exceeds 10 hours when utilising the 4-node SpiNNaker board. Lastly, deploying this implementation may necessitate the use of neuromorphic hardware as the most viable option. This is attributed to the challenging nature of implementing the model on FPGA or other hardware platforms due to its utilisation of conductance-based neurons and a highly recurrent architectural model.

10.4.2 Bin-ratio ensemble SNN (BESNN)

The following parts can be possibly improved in the future:

- Fully end-to-end event-based flow development is needed, i.e. bring logarithm pre-processing into spiking paradigm too.
- Intelligent algorithm that can decide how many nets should be needed before batch training.

The mentioned BESNN takes the analogue value as input, this is due to the fact that logarithm could not be performed by spiking neurons at the moment. Even though the current approach retains more accuracy, moving towards an end-to-end event-based processing system could further improve power efficiency and latency.

Regarding the algorithm, 20 nets may seem excessive for the task, this methodology cannot have an intelligent decision making about how many exactly nets are needed for ensemble learning before training individually. This algorithm cannot decide which is the best combination before batch training. Addressing this would potentially improve the training overhead and speed up the overall deployment.

10.5 Personal Reflection

Overall during this PhD programme, I have grown significantly. No matter if it is research habit or mentality, I have become a more independent researcher with the projects I have either been part of or personally led.

10.5.1 Research habits

I have picked up some good research habits and also identified some problems that I used to have. Always leave a detailed research log of what has been done and what the corresponding results and analysis are. Human beings tend to be forgetful, researchers are not exceptions. Sometimes a big project can last more than 6 months, keeping a log of any kind (it can be a dated lab book, a digital markdown file, or other tools to keep the notes) will always remind the researcher how the progress has been made. Besides, this can assist one to systematically organise the research work into a piece of comprehensive paper for publication.

Keep a copy of the work for backup. As a PhD of engineering student, I have been mainly working with simulations and programming. Backing up code is always a good practice to have version control and to provide precautionary measures for emergencies. GitHub has been a highly valued website for this practice.

Another point needs to be emphasised is the importance of making extensive comments along side the code. This practice not only improves the comprehensibility but also facilitates potential future maintenance and mitigates the risk of ambiguities or errors during the development. The person who is more likely to look back on the code of you is always yourself.

Good research tools should be proficiently used to pave the way for research. For example, LaTeX, as a professional academic writing language, should be adeptly mastered. Zotero is another recommended tool for systematically organising references and bibliography. This would be substantially handy when writing a long thesis with more than 100 references.

I used to not to keep up with the new academic papers frequently. This should be improved in the future with the help of Zotero. For a research field like artificial intelligence and neuromorphic engineering, keeping up to date with state of the art is a necessity. This is because the developing speed these years have been increasingly fast.

10.5.2 Mentality

Doing research is like running a marathon, it cannot be achieved quickly with a single sprint. Perseverance and execution are the keys to success. Only when one keeps making progress that a substantial work with continuity can be developed. "Rome is not built in one day", PhD is not awarded for just one month of hard work.

Also one needs to be wary of the external impact brought by other colleagues. One's own research work should always be the priority and should not be distracted or overpowered by other's demands and requests.

But in the meantime, a professional researcher should keep an open mind to other's opinion, which can be drastically different sometimes. It does not mean one should be agreeable, but keep a critical mind towards the different idea. What can we learn from them? This is a more important question to think rather than who is correct.

Group project is likely to be more fruitful than individual project. This is a good way to quickly have some research outcomes and raise one's research reputation.

10.6 Future Research Directions

My upcoming project involves digital design and a chip tape out for an imaging system, this will primarily be my research interest. But other future research directions should also include neuromorphic engineering with both hardware design and algorithm development. I would still be delighted to see progress being made for new training methods for SNN and possible combination of asynchronous logic and event-based processing.

On the other hand, in terms of this specific algorithm and implementation, it would also be curious to explore the possibility of using micro-controller for machine learning tasks and compare the power efficiency between the micro-controller and FPGA. This can also be taken further for an application specific integrated circuits (ASIC) design for further power saving.

10.7 Conclusion Statement

In this four years PhD project, we have established a successful flow for an event-based system development, which can be applied to radioisotope identification. This is gradually achieved by delivering proof-of-concept projects, and carrying out solid algorithm-hardware co-design. The whole system is validated by an FPGA prototype that demonstrates the strong capability of developed algorithm. This demonstrates the performance superiority over the pre-existing software solution that runs on a handheld device.

Other than that, the proposed hardware solution shows promising results in terms of power saving. This has been confirmed to fit within a compact power envelope, potentially extending the battery life of the RIID device.

Appendix A

List of publications

Here is the list of publications made during PhD programme.

A.1 Conference papers

There are 3 conference publications:

1. Huang, X., Jones, E., Zhang, S., Xie, S., Furber, S., Goulermas, Y., . . . Hamilton, A. (2020). Spiking neural network based low-power radioisotope identification using fpga. In *2020 27th ieee international conference on electronics, circuits and systems (icecs)* (p. 1-4). doi: 10.1109/ICECS49266.2020.9294873
2. Huang, X., Jones, E., Zhang, S., Xie, S., Furber, S., Goulermas, Y., . . . Hamilton, A. (2021). An fpga implementation of convolutional spiking neural networks for radioisotope identification. In *2021 ieee international symposium on circuits and systems (iscas)* (p. 1-5). doi: 10.1109/ISCAS51556.2021.9401412
3. Xie, S., Jones, E., Marsden, E., Baistow, I., Furber, S., Mitra, S., & Hamilton, A. (2022). Unsupervised stdp-based radioisotope identification using spiking neural networks implemented on spinnaker. In *2022 8th international conference on event-based control, communication, and signal processing (ebccsp)* (p. 1-8). doi: 10.1109/EBCCSP56922.2022.9845586

A.2 Journal papers

There is 1 journal publication:

- Xie, S., Jones, E., Zhang, S., Marsden, E., Baistow, I., Furber, S., . . . Hamilton, A.(2024, August). FPGA-based fast bin-ratio spiking ensemble network for radioisotope identification. *Neural Networks, 176*, 106332. Retrieved 2024-05-27, from `https://www.sciencedirect.com/science/article/pii/S0893608024002569` doi: 10.1016/j.neunet.2024.106332

Appendix B

Rough estimation of the cost for proposed implementation

B.1 VCU108 evaluation kit

The mentioned board in chapter 9 is a kind donation from Xilinx™ through their Xilinx University Programme (XUP) for education and research purpose. However, if one wishes to purchase one for their own implementation. The price listed for this board is $ 7,770.00, this includes all the necessary cables and external modules and boards for self testing.

B.2 Artix board

There are also other boards mentioned in the power analysis section for chapter 9. Artix-7™ is one of them. Technically speaking, Artix-7 does not represent a specific board but a family that follows the same architecture design with different specifications. There are two Artix-7™ FPGA parts mentioned in this section. And they are:

- xc7a100tcsg324-3
- xc7a35tcpg236-3

The first FPGA part xc7a100tcsg324-3 has been featured in an FPGA product called Arty A7-100T. This can be found on Digilent, Inc. website. As listed, this FPGA board shall cost $ 299.00. Even though this board features a slightly different part which is xc7a100tcsg324-1, this should not impact the total resources available on board. The last number simply indicates the performing speed.

While the second FPGA part can also be found in a similar product from Digilent, Inc., it is called Basys 3 Artix-7 FPGA Trainer Board. This is an entry level FPGA board that only costs $ 165.00. Again, this product is simply another board that features a slightly slower parts.

B.3 Spartan board

The other good alternative might be part that is within Spartan-7™ family, which is xc7s100 fgga484-2. This is a slightly rarer part to be found, the closest board I could find that features a similar part is Spartan 7 SP701 FPGA Evaluation Kit from Xilinx. The board is included in this evaluation kit that costs $ 836.00. This features a part xc7s100fgga676-2, which should have the same amount of resources as xc7s100fgga484-2. The only difference is the package and pin count.

Appendix C

GitHub availability

There have been some simulation results shown in previous chapters that the reader can reproduce on their own. Here we provide a GitHub repository that contains the code needed for these simple simulations.

This repository can be found here. These codes are organised by the order of appearance by chapter. They are mainly drafted in Python and can be run with necessary packages needed. Details can be found in the README.md file attached within the repository.

Bibliography

Aarnio, P., Routti, J., & Sandberg, J. (1988). Microsampo—personal computer based advanced gamma spectrum analysis system. *Journal of Radioanalytical and Nuclear Chemistry*, *124*(2), 457–465.

Aarnio, P. A., Routti, J. T., & Sandberg, J. V. (n.d.). MicroSAMPO — Personal computer based advanced gamma spectrum analysis system. *Journal of Radioanalytical and Nuclear Chemistry Articles*, *124*(2), 457–465. Retrieved 2021-07-26, from `http://link.springer.com/10.1007/BF02041336` doi: 10.1007/BF02041336

Abdollahi, M., & Liu, S.-C. (2011). Speaker-independent isolated digit recognition using an aer silicon cochlea. In *2011 ieee biomedical circuits and systems conference (biocas)* (p. 269-272). doi: 10.1109/BioCAS.2011.6107779

Abubakar, I. R., & Olatunji, S. O. (2020). Computational intelligence-based model for diarrhea prediction using demographic and health survey data. *Soft Computing*, *24*(7), 5357–5366.

Abusland, A., Lande, T., & Hovin, M. (1996). A vlsi communication architecture for stochastically pulse-encoded analog signals. In *1996 ieee international symposium on circuits and systems. circuits and systems connecting the world. iscas 96* (Vol. 3, p. 401-404 vol.3). doi: 10.1109/ISCAS.1996.541618

Aeschlimann, F., Allier, E., Fesquet, L., & Renaudin, M. (2004). Asynchronous fir filters: towards a new digital processing chain. In *10th international symposium on asynchronous circuits and systems, 2004. proceedings.* (p. 198-206). doi: 10.1109/ASYNC.2004.1299303

Ahn, E. Y., Lee, J. H., Mullen, T., & Yen, J. (2011). Dynamic vision sensor camera based bare hand gesture recognition. In *2011 ieee symposium on computational intelligence for multimedia, signal and vision processing* (p. 52-59). doi: 10.1109/CIMSIVP.2011.5949251

Akopyan, F., Manohar, R., & Apsel, A. (2006). A level-crossing flash asynchronous analog-to-digital converter. In *12th ieee international symposium on asynchronous circuits and systems (async'06)* (p. 11 pp.-22). doi: 10.1109/ASYNC.2006.5

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., ... Modha, D. S. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *34*(10), 1537-1557. doi: 10.1109/TCAD.2015.2474396

Alamaniotis, M., Heifetz, A., Raptis, A. C., & Tsoukalas, L. H. (2013). Fuzzy-logic radioisotope identifier for gamma spectroscopy in source search. *IEEE Transactions on Nuclear Science*, *60*(4), 3014-3024. doi: 10.1109/TNS.2013.2265307

Alamaniotis, M., Lee, S., & Jevremovic, T. (2015). Intelligent analysis of low-count scintillation spectra using support vector regression and fuzzy logic. *Nuclear Technology*, *191*(1), 41–57.

Allier, E., Sicard, G., Fesquet, L., & Renaudin, M. (2003). A new class of asynchronous a/d converters based on time quantization. In *Ninth international symposium on asynchronous circuits and systems, 2003. proceedings.* (p. 196-205). doi: 10.1109/ASYNC.2003.1199179

AMD. (2023). *AMD Virtex UltraScale FPGA VCU108 Evaluation Kit.* Retrieved 2024-02-22, from `https://www.xilinx.com/products/boards-and-kits/ek-u1-vcu108-g.html`

American national standard performance criteria for handheld instruments for the detection and identification of radionuclides, ansi n42.34-2015 (revision of ansi n42.34-2006). (IEEE Standard, 2016). doi: 10.1109/IEEESTD.2016.7551091

Amirshahi, A., & Hashemi, M. (2019). Ecg classification algorithm based on stdp and r-stdp neural networks for real-time monitoring on ultra low-power personal wearable devices. *IEEE transactions on biomedical circuits and systems*, *13*(6), 1483–1493.

Anonymous. (2022). *What is a 'dirty bomb' and why is russia saying ukraine could use one?* Retrieved from `https://www.bbc.co.uk/news/world-63373637` (Last accessed on 25 Oct 2023)

Barua, S., Miyatani, Y., & Veeraraghavan, A. (2016). Direct face detection and video reconstruction from event cameras. In *2016 ieee winter conference on applications of computer vision (wacv)* (p. 1-9). doi: 10.1109/WACV.2016.7477561

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T., Rasmussen, D., ... Eliasmith, C. (2014). Nengo: a python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, *7*. Retrieved from `https://www.frontiersin.org/articles/10.3389/fninf.2013.00048` doi: 10.3389/fninf.2013.00048

Bilton, K. J., Joshi, T. H., Bandstra, M. S., Curtis, J. C., Quiter, B. J., Cooper, R. J., & Vetter, K. (2019). Non-negative matrix factorization of gamma-ray spectra for background modeling, detection, and source identification. *IEEE Transactions on Nuclear Science*, *66*(5), 827-837. doi: 10.1109/TNS.2019.2907267

Boahen, K. (2000). Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *47*(5), 416-434. doi: 10.1109/82.842110

Boardman, D., Reinhard, M., & Flynn, A. (2012). Principal component analysis of gamma-ray spectra for radiation portal monitors. *IEEE Transactions on Nuclear Science*, *59*(1), 154-160. doi: 10.1109/TNS.2011.2179313

Bohte, S. M., Kok, J. N., & La Poutré, H. (2002, October). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1), 17–37. Retrieved 2024-02-01, from https://www.sciencedirect.com/science/article/pii/S0925231201006580 doi: 10.1016/S0925-2312(01)00658-0

Broyles, C., Thomas, D., & Haynes, S. (1953). The measurement and interpretation of the k auger intensities of sn113, cs137, and au198 [Article]. *Physical Review*, *89*(4), 715 – 724. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-36149008329&doi=10.1103%2fPhysRev.89.715&partnerID=40&md5=8c5c2635cf15d40c6fd5d3e5d9b488a0 (Cited by: 33) doi: 10.1103/PhysRev.89.715

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., & Huang, T. (2023). *Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks.*

Carlisle, D. (2007). Dhiren barot: Was he an al qaeda mastermind or merely a hapless plotter? *Studies in Conflict & Terrorism*, *30*(220901), 1057–1071.

Ceolini, E., Anumula, J., Braun, S., & Liu, S.-C. (2019). Event-driven pipeline for low-latency low-compute keyword spotting and speaker verification system. In *Icassp 2019 - 2019 ieee international conference on acoustics, speech and signal processing (icassp)* (p. 7953-7957). doi: 10.1109/ICASSP.2019.8683669

Chen, L., & Wei, Y.-X. (2009). Nuclide identification algorithm based on k–l transform and neural networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *598*(2), 450-453. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168900208014435 doi: https://doi.org/10.1016/j.nima.2008.09.035

Cieślak, M. J., Gamage, K. A., & Glover, R. (2019). Critical review of scintillating crystals for neutron detection. *Crystals*, *9*(9), 480.

Ciscato, D., & Martiani, L. (1967). On increasing sampling efficiency by adaptive sampling. *IEEE Transactions on Automatic Control*, *12*(3), 318-318. doi: 10.1109/TAC.1967.1098605

Cohen, G., Afshar, S., Morreale, B., Bessell, T., Wabnitz, A., Rutten, M., & van Schaik, A. (2019, Jun 01). Event-based sensing for space situational awareness. *The Journal of the Astronautical Sciences*, *66*(2), 125-141. Retrieved from https://doi.org/10.1007/s40295-018-00140-5 doi: 10.1007/s40295-018-00140-5

Cosofret, B. R., Shokhirev, K., Mulhall, P., Payne, D., & Harris, B. (2014). Utilization of advanced clutter suppression algorithms for improved standoff detection and identification of radionuclide threats. In *Chemical, biological, radiological, nuclear, and explosives (cbrne) sensing xv* (Vol. 9073, pp. 253–265).

Cosofret, B. R., Shokhirev, K. N., Mulhall, P. A., Payne, D., Harris, B., Arsenault, E., & Moro, R. (2013). Utilization of advanced clutter suppression algorithms for improved spectroscopic portal capability against radionuclide threats. In *2013 ieee international conference on technologies for homeland security (hst)* (p. 618-622). doi: 10.1109/THS.2013.6699075

Daniel, G., Ceraudo, F., Limousin, O., Maier, D., & Meuris, A. (2020). Automatic and real-time identification of radionuclides in gamma-ray spectra: A new method based on convolutional neural network trained with synthetic data set. *IEEE Transactions on Nuclear Science*, *67*(4), 644-653. doi: 10.1109/TNS.2020.2969703

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., ... Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, *38*(1), 82-99. doi: 10.1109/MM.2018.112130359

Davison, A., Brüderle, D., Eppler, J., Kremkow, J., Muller, E., Pecevski, D., ... Yger, P. (2009). Pynn: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, *2*. Retrieved from https://www.frontiersin.org/articles/10.3389/neuro.11.011.2008 doi: 10.3389/neuro.11.011.2008

Davoodi, M. R., Meskin, N., & Khorasani, K. (2015). Event-triggered fault detection, isolation and control design of linear systems. In *2015 international conference on event-based control, communication, and signal processing (ebccsp)* (p. 1-6). doi: 10.1109/EBCCSP.2015.7300650

Delbruck, T., et al. (2008). Frame-free dynamic digital vision. In *Proceedings of intl. symp. on secure-life electronics, advanced electronics for quality life and society* (Vol. 1, pp. 21–26).

Diehl, P. U., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, *9*, 99.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., & Pfeiffer, M. (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 international joint conference on neural networks (ijcnn)* (p. 1-8). doi: 10.1109/IJCNN.2015.7280696

Dorf, R., Farren, M., & Phillips, C. (1962). Adaptive sampling frequency for sampled-data control systems. *IRE Transactions on Automatic Control*, *7*(1), 38-47. doi: 10.1109/TAC.1962.1105415

Du, T., Zhao, Z., Zhu, Q., & Tian, L. (2021). Locating a γ-ray source using cuboid scintillators and the knn algorithm. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *993*, 165069. Retrieved from https://www.sciencedirect.com/science/article/pii/S016890022100053X doi: https://doi.org/10.1016/j.nima.2021.165069

Eliasmith, C., & Anderson, C. H. (2004). *Neural engineering: computational, representation, and dynamics in neurobiological systems* (1. MIT Press paperback ed ed.). Cambridge, Mass.: MIT Press.

Encyclopaedia Britannica. (2023). *Fukushima accident.* Retrieved from https://www.britannica.com/event/Fukushima-accident (Last accessed on 23 Oct 2023)

Eppler, J., Helias, M., Muller, E., Diesmann, M., & Gewaltig, M.-O. (2009). Pynest: a convenient interface to the nest simulator. *Frontiers in Neuroinformatics*, *2*. Retrieved from https://www.frontiersin.org/articles/10.3389/neuro.11.012.2008 doi: 10.3389/neuro.11.012.2008

Eshraghian, J. K., Ward, M., Neftci, E. O., Wang, X., Lenz, G., Dwivedi, G., ... Lu, W. D. (2023). Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, *111*(9), 1016-1054. doi: 10.1109/JPROC.2023.3308088

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., & Modha, D. S. (2020). *Learned step size quantization.*

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., ... Modha, D. S. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, *113*(41), 11441-11446. Retrieved from https://www.pnas.org/doi/abs/10.1073/pnas.1604850113 doi: 10.1073/pnas.1604850113

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., ... Socher, R. (2021). Deep learning-enabled medical computer vision. *NPJ digital medicine*, *4*(1), 1–9.

Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., ... others (2020). *Spikingjelly.* https://github.com/fangwei123456/spikingjelly. (Accessed: 2023-09-26)

Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., ... Tian, Y. (2023, October). SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, *9*(40), eadi1480. Retrieved 2024-02-07, from https://www.science.org/doi/full/10.1126/sciadv.adi1480 (Publisher: American Association for the Advancement of Science) doi: 10.1126/sciadv.adi1480

Figliolia, T., Murray, T., & Andreou, A. (2015). Acoustic micro-doppler signal processing with foveated electronic cochlea. *Electronics Letters*, *51*(2), 132-134. Retrieved from https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/el.2014.3711 doi: https://doi.org/10.1049/el.2014.3711

Finger, H., & Liu, S.-C. (2011). Estimating the location of a sound source with a spike-timing localization algorithm. In *2011 ieee international symposium of circuits and systems (iscas)* (p. 2461-2464). doi: 10.1109/ISCAS.2011.5938102

Foster, J., & Wang, T.-K. (1991). Speech coding using time code modulation. In *Ieee proceedings of the southeastcon '91* (p. 861-863 vol.2). doi: 10.1109/SECON.1991.147882

Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The spinnaker project. *Proceedings of the IEEE*, *102*(5), 652-665. doi: 10.1109/JPROC.2014.2304638

Gallego, G., Forster, C., Mueggler, E., & Scaramuzza, D. (2015). *Event-based camera pose tracking using a generative event model.*

Gewaltig, M.-O., & Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, *2*(4), 1430.

Ghosh-Dastidar, S., & Adeli, H. (2009). Spiking neural networks. *International Journal of Neural Systems*, *19*(04), 295-308. Retrieved from https://doi.org/10.1142/S0129065709002002 (PMID: 19731402) doi: 10.1142/S0129065709002002

Gomez-Fernandez, M., Wong, W.-K., Tokuhiro, A., Welter, K., Alhawsawi, A. M., Yang, H., & Higley, K. (2021). Isotope identification using deep learning: An explanation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *988*, 164925. Retrieved from https://www.sciencedirect.com/science/article/pii/S016890022031322X doi: https://doi.org/10.1016/j.nima.2020.164925

Gómez-Rodríguez, F., Miró-Amarante, L., Rivas, M., Jimenez, G., & Diaz-del Rio, F. (2011). Neuromorphic real-time objects tracking using address event representation and silicon retina. In J. Cabestany, I. Rojas, & G. Joya (Eds.), *Advances in computational intelligence* (pp. 133–140). Berlin, Heidelberg: Springer Berlin Heidelberg.

Goodman, D., & Brette, R. (2008). Brian: a simulator for spiking neural networks in python. *Frontiers in Neuroinformatics*, *2*. Retrieved from https://www.frontiersin.org/articles/10.3389/neuro.11.005.2008 doi: 10.3389/neuro.11.005.2008

Gutierrez-Galan, D., Dominguez-Morales, J. P., Perez-Peña, F., Jimenez-Fernandez, A., & Linares-Barranco, A. (2020, March). Neuropod: A real-time neuromorphic spiking CPG applied to robotics. *Neurocomputing*, *381*, 10–19. Retrieved 2024-01-31, from https://www.sciencedirect.com/science/article/pii/S0925231219315644 doi: 10.1016/j.neucom.2019.11.007

Göttsche, M., Schirm, J., & Glaser, A. (n.d.). Low-resolution gamma-ray spectrometry for an information barrier based on a multi-criteria template-matching approach. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *840*, 139–144. Retrieved 2021-07-26, from https://www.sciencedirect.com/science/article/pii/S0168900216310403 doi: 10.1016/j.nima.2016.10.013

Göttsche, M., Schirm, J., & Glaser, A. (2016). Low-resolution gamma-ray spectrometry for an information barrier based on a multi-criteria template-matching approach. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *840*, 139-144. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168900216310403 doi: https://doi.org/10.1016/j.nima.2016.10.013

Hague, E. J., Kamuda, M., Ford, W. P., Moore, E. T., & Turk, J. (2019). A comparison of adaptive and template matching techniques for radio-isotope identification. In *Algorithms, technologies, and applications for multispectral and hyperspectral imagery xxv* (Vol. 10986, pp. 62–73).

Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., & Kozma, R. (2018). Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics*, *12*. Retrieved from https://www.frontiersin.org/articles/10.3389/fninf.2018.00089 doi: 10.3389/fninf.2018.00089

Hebb, d. o. the organization of behavior: A neuropsychological theory. new york: John wiley and sons, inc., 1949. 335 p. $4.00. (1950). *Science Education*, *34*(5), 336-337. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/sce.37303405110 doi: https://doi.org/10.1002/sce.37303405110

Heemels, W., Gorter, R., van Zijl, A., van den Bosch, P., Weiland, S., Hendrix, W., & Vonder, M. (1999). Asynchronous measurement and control: a case study on motor synchronization. *Control Engineering Practice*, *7*(12), 1467-1482. Retrieved from https://www.sciencedirect.com/science/article/pii/S0967066199001136 doi: https://doi.org/10.1016/S0967-0661(99)00113-6

Hilger Crystals. (2023). *How do scintillators work?* Retrieved from https://www.hilger-crystals.co.uk/2023/04/13/how-do-scintillators-work/

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, *117*(4), 500.

Huang, J., Vogginger, B., Gerhards, P., Kreutz, F., Kelber, F., Scholz, D., ... Mayr, C. G. (2022). Real-time radar gesture classification with spiking neural network on spinnaker 2 prototype. In *2022 ieee 4th international conference on artificial intelligence circuits and systems (aicas)* (p. 362-365). doi: 10.1109/AICAS54282.2022.9869987

Huang, X., Jones, E., Zhang, S., Furber, S., Goulermas, Y., Marsden, E., ... Hamilton, A. (2020). Event-based signal processing for radioisotope identification. In *2020 6th international conference on event-based control, communication, and signal processing (ebccsp)* (p. 1-4). doi: 10.1109/EBCCSP51266.2020.9291349

Huang, X., Jones, E., Zhang, S., Xie, S., Furber, S., Goulermas, Y., ... Hamilton, A. (2020). Spiking neural network based low-power radioisotope identification using fpga. In *2020 27th ieee international conference on electronics, circuits and systems (icecs)* (p. 1-4). doi: 10.1109/ICECS49266.2020.9294873

Huang, X., Jones, E., Zhang, S., Xie, S., Furber, S., Goulermas, Y., ... Hamilton, A. (2021). An fpga implementation of convolutional spiking neural networks for radioisotope identification. In *2021 ieee international symposium on circuits and systems (iscas)* (p. 1-5). doi: 10.1109/ISCAS51556.2021.9401412

Hwang, S., Chang, J., Oh, M.-H., Min, K. K., Jang, T., Park, K., ... Park, B.-G. (2021). Low-latency spiking neural networks using pre-charged membrane potential and delayed evaluation. *Frontiers in Neuroscience*, *15*. Retrieved from https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2021.629000 doi: 10.3389/fnins.2021.629000

Imam, N., & Manohar, R. (2011). Address-event communication using token-ring mutual exclusion. In *2011 17th ieee international symposium on asynchronous circuits and systems* (p. 99-108). doi: 10.1109/ASYNC.2011.20

International Atomic Energy Agency. (2016). *Frequently asked chernobyl questions | iaea.* Retrieved from https://www.iaea.org/newscenter/focus/chernobyl/faqs (Last accessed on 23 Oct 2023)

International Atomic Energy Agency. (2023). *Iaea releases annual data on illicit trafficking of nuclear and other radioactive material.* Retrieved from https://www.iaea.org/newscenter/pressreleases/iaea-releases-annual-data-on-illicit-trafficking-of-nuclear-and-other-radioactive-material (Last accessed on 31 Oct 2023)

Iyer, L. R., & Basu, A. (2017). Unsupervised learning of event-based image recordings using spike-timing-dependent plasticity. In *2017 international joint conference on neural networks (ijcnn)* (pp. 1840–1846).

Izhikevich, E. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569-1572. doi: 10.1109/TNN.2003.820440

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... Kalenichenko, D. (2017). *Quantization and training of neural networks for efficient integer-arithmetic-only inference.*

James Martin Center for Nonproliferation Studies for Nuclear Threat Initiativeve. (2023). *Overview of the cns global incidents and trafficking database.* (data retrieved from CNS, `https://www.nti.org/analysis/articles/overview-of-the-cns-global-incidents-and-trafficking-database/#report`)

Jeon, B., Kim, J., Lee, E., Moon, M., & Cho, G. (2021). Pseudo-gamma spectroscopy based on plastic scintillation detectors using multitask learning. *Sensors*, *21*(3), 684.

Kamuda, M., Stinnett, J., & Sullivan, C. J. (2017). Automated isotope identification algorithm using artificial neural networks. *IEEE Transactions on Nuclear Science*, *64*(7), 1858-1864. doi: 10.1109/TNS.2017.2693152

Kamuda, M., Zhao, J., & Huff, K. (2020). A comparison of machine learning methods for automated gamma-ray spectroscopy. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *954*, 161385. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0168900218313779` (Symposium on Radiation Measurements and Applications XVII) doi: https://doi.org/10.1016/j.nima.2018.10.063

Katare, D., & El-Sharkawy, M. (2019). Embedded system enabled vehicle collision detection: An ann classifier. In *2019 ieee 9th annual computing and communication workshop and conference (ccwc)* (p. 0284-0289). doi: 10.1109/CCWC.2019.8666562

Keller, P., Kangas, L., Troyer, G., Hashem, S., & Kouzes, R. (1995). Nuclear spectral analysis via artificial neural networks for waste handling. *IEEE Transactions on Nuclear Science*, *42*(4), 709-715. doi: 10.1109/23.467888

Keller, P., & Kouzes, R. (1994). Gamma spectral analysis via neural networks. In *Proceedings of 1994 ieee nuclear science symposium - nss'94* (Vol. 1, p. 341-345 vol.1). doi: 10.1109/NSSMIC.1994.474365

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., & Masquelier, T. (2018, March). STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, *99*, 56–67. Retrieved 2024-02-05, from `https://www.sciencedirect.com/science/article/pii/S0893608017302903` doi: 10.1016/j.neunet.2017.12.005

Kim, S., Park, S., Na, B., & Yoon, S. (2020, Apr.). Spiking-yolo: Spiking neural network for energy-efficient object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(07), 11270-11277. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/6787 doi: 10.1609/aaai.v34i07.6787

Knoll, G. F. (n.d.). *Radiation detection and measurement* (4th ed. ed.). Wiley.

Kogler, J., Sulzbachner, C., Humenberger, M., & Eibensteiner, F. (2011). Address-event based stereo vision with bio-inspired silicon retina imagers. *Advances in theory and applications of stereo vision*, 165–188.

Krock, L., & Deusser, R. (2003). *Dirty bomb | chronology of events.* Retrieved from https://www.pbs.org/wgbh/nova/dirtybomb/chrono.html (Last accessed on 25 Oct 2023)

Kromek Group PLC. (2023). *Kromek | security & defence products.* Retrieved from https://www.kromek.com/detection/security-and-defence/security-and-defence-products/ (Last accessed on 7 Dec 2023)

Kulkarni, S. R., & Rajendran, B. (2018). Spiking neural networks for handwritten digit recognition—supervised learning and network optimization. *Neural Networks*, *103*, 118–127.

Lazzaro, J., Wawrzynek, J., Mahowald, M., Sivilotti, M., & Gillespie, D. (1993). Silicon auditory processors as computer peripherals. *IEEE Transactions on Neural Networks*, *4*(3), 523-528. doi: 10.1109/72.217193

Lee, C., Srinivasan, G., Panda, P., & Roy, K. (2019). Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity. *IEEE Transactions on Cognitive and Developmental Systems*, *11*(3), 384-394. doi: 10.1109/TCDS.2018.2833071

Leo, W. R. (2012). *Techniques for nuclear and particle physics experiments: a how-to approach*. Springer Science & Business Media.

Li, C., Ma, L., & Furber, S. (2022). Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, *16*. Retrieved from https://www.frontiersin.org/articles/10.3389/fnins.2022.918793 doi: 10.3389/fnins.2022.918793

Li, Y., Shepard, K., & Tsividis, Y. (2005). Continuous-time digital signal processors. In *11th ieee international symposium on asynchronous circuits and systems* (p. 138-143). doi: 10.1109/ASYNC.2005.15

Liang, D., Gong, P., Tang, X., Wang, P., Gao, L., Wang, Z., & Zhang, R. (2019). Rapid nuclide identification algorithm based on convolutional neural network. *Annals of Nuclear Energy*, *133*, 483-490. Retrieved from https://www.sciencedirect.com/science/article/pii/S0306454919303044 doi: https://doi.org/10.1016/j.anucene.2019.05.051

Liechti, C. (2015). *pyserial: Python Serial Port Extension.* Retrieved 2024-02-23, from https://github.com/pyserial/pyserial

Litzenberger, M., Posch, C., Bauer, D., Belbachir, A., Schon, P., Kohn, B., & Garn, H. (2006). Embedded vision system for real-time object tracking using an asynchronous transient vision sensor. In *2006 ieee 12th digital signal processing workshop & 4th ieee signal processing education workshop* (p. 173-178). doi: 10.1109/DSPWS.2006.265448

Liu, S.-C., van Schaik, A., Minch, B. A., & Delbruck, T. (2010). Event-based 64-channel binaural silicon cochlea with q enhancement mechanisms. In *2010 ieee international symposium on circuits and systems (iscas)* (p. 2027-2030). doi: 10.1109/ISCAS.2010.5537164

Luo, Y., Xu, M., Yuan, C., Cao, X., Xu, Y., Wang, T., & Feng, Q. (2020). Siamsnn: Spike-based siamese network for energy-efficient and real-time object tracking. *CoRR*, *abs/2003.07584*. Retrieved from https://arxiv.org/abs/2003.07584

Lyon, R., & Mead, C. (1988). An analog electronic cochlea. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *36*(7), 1119-1134. doi: 10.1109/29.1639

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks*, *10*(9), 1659–1671.

Mark, J., & Todd, T. (1981). A nonuniform sampling approach to data compression. *IEEE Transactions on Communications*, *29*(1), 24-32. doi: 10.1109/TCOM.1981.1094872

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, *275*(5297), 213–215.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997, January). Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs. *Science*, *275*(5297), 213–215. Retrieved 2024-02-05, from https://www.science.org/doi/10.1126/science.275.5297.213 (Publisher: American Association for the Advancement of Science) doi: 10.1126/science.275.5297.213

Masquelier, T., Guyonneau, R., & Thorpe, S. J. (2009, 05). Competitive STDP-Based Spike Pattern Learning. *Neural Computation*, *21*(5), 1259-1276. Retrieved from https://doi.org/10.1162/neco.2008.06-08-804 doi: 10.1162/neco.2008.06-08-804

Mayr, C., Hoeppner, S., & Furber, S. (2019). *Spinnaker 2: A 10 million core processor system for brain simulation and machine learning.*

Medhat, M. (2012). Artificial intelligence methods applied for quantitative analysis of natural radioactive sources. *Annals of Nuclear Energy*, *45*, 73-79. Retrieved from https://www.sciencedirect.com/science/article/pii/S030645491200059X doi: https://doi.org/10.1016/j.anucene.2012.02.013

Meslem, N., & Prieur, C. (2015). Event-based stabilizing controller using a state observer. In *2015 international conference on event-based control, communication, and signal processing (ebccsp)* (p. 1-8). doi: 10.1109/EBCCSP.2015.7300653

Ministry of Health, Labour and Welfare. (2019). *Responses taken by the ministry of health, labour and welfare of japan on radiation protection at works relating to the accident at tepco's fukushima daiichi nuclear power plant.* Retrieved from https://www.mhlw.go.jp/english/topics/2011eq/workers/ri/gr/gr_190131.pdf (Last accessed on 23 Oct 2023)

Miró-Amarante, L., Gómez-Rodríguez, F., Jiménez-Fernández, A., & Jiménez-Moreno, G. (2017). A spiking neural network for real-time spanish vowel phonemes recognition. *Neurocomputing*, *226*, 249-261. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231216314771 doi: https://doi.org/10.1016/j.neucom.2016.12.005

Miskowicz, M. (2007). Asymptotic effectiveness of the event-based sampling according to the integral criterion. *Sensors*, *7*(1), 16–37.

Miskowicz, M. (2018). *Event-based control and signal processing*. CRC press.

Mitchell, D., & Mattingly, J. (2009, 12). *Gamma detector response and analysis software (gadras) v. 16.0, version 01.* Retrieved from https://www.osti.gov/biblio/1231259

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S. J., & Masquelier, T. (2019). Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*, *94*, 87-95. Retrieved from https://www.sciencedirect.com/science/article/pii/S0031320319301906 doi: https://doi.org/10.1016/j.patcog.2019.05.015

Mozafari, M., Kheradpisheh, S. R., Masquelier, T., Nowzari-Dalini, A., & Ganjtabesh, M. (2018). First-spike-based visual categorization using reward-modulated stdp. *IEEE transactions on neural networks and learning systems*, *29*(12), 6178–6190.

Nessler, B., Pfeiffer, M., Buesing, L., & Maass, W. (2013, April). Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity. *PLoS Computational Biology*, *9*(4), e1003037. Retrieved 2024-02-05, from https://dx.plos.org/10.1371/journal.pcbi.1003037 doi: 10.1371/journal.pcbi.1003037

Ninh, G. N., Phongphaeth, P., Nares, C., & Hao, Q. N. (2016, 01). Radioisotope identification method for poorly resolved gamma-ray spectrum of nuclear security concern. *AIP Conference Proceedings*, *1704*(1), 050005. Retrieved from https://doi.org/10.1063/1.4940101 doi: 10.1063/1.4940101

Näger, C., Storck, J., & Deco, G. (2002). Speech recognition with spiking neurons and dynamic synapses: a model motivated by the human auditory pathway. *Neurocomputing*, *44-46*, 937-942. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231202004940 (Computational Neuroscience Trends in Research 2002) doi: https://doi.org/10.1016/S0925-2312(02)00494-0

Occupational Safety and Health Administration. (2023). *Ionizing radiation.* Retrieved from https://www.osha.gov/ionizing-radiation/background (Last accessed on 1 Nov 2023)

Ozkan, I. A., Koklu, M., & Sert, I. U. (2018). Diagnosis of urinary tract infection based on artificial intelligence methods. *Computer methods and programs in biomedicine*, *166*, 51–59.

Painkras, E., Plana, L. A., Garside, J., Temple, S., Galluppi, F., Patterson, C., ... Furber, S. B. (2013). Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, *48*(8), 1943-1953. doi: 10.1109/JSSC.2013.2259038

Pfeiffer, M., & Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, *12*. Retrieved from https://www.frontiersin.org/article/10.3389/fnins.2018.00774 doi: 10.3389/fnins.2018.00774

Pfister, J.-P., & Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, *26*(38), 9673–9682. Retrieved from https://www.jneurosci.org/content/26/38/9673 doi: 10.1523/JNEUROSCI.1425-06.2006

Photek Limited. (2021). *Photomultipliers.* Retrieved from https://www.photek.com/photomultipliers/

Plesser, H. E., Eppler, J. M., Morrison, A., Diesmann, M., & Gewaltig, M.-O. (2007). Efficient Parallel Simulation of Large-Scale Neuronal Networks on Clusters of Multiprocessor Computers. In A.-M. Kermarrec, L. Bougé, & T. Priol (Eds.), *Euro-Par 2007 Parallel Processing* (pp. 672–681). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-540-74466-5_71

Proctor, A. (1988). *Ricki. interactive gamma spectral analysis* (Tech. Rep.). EG and G Idaho, Inc., Idaho Falls, ID (United States).

Radiation Monitoring Devices Inc. (2023). *Gamma-neutron scintillator properties.* Retrieved from https://www.rmdinc.com/assets/CLLBC-Gamma-Neutron-Scintillator-Properties.pdf (Last accessed on 8 Dec 2023)

Reuther, A., Michaleas, P., Jones, M., Gadepally, V., Samsi, S., & Kepner, J. (2019). Survey and benchmarking of machine learning accelerators. In *2019 ieee high performance extreme computing conference (hpec)* (p. 1-9). doi: 10.1109/HPEC.2019.8916327

Rhodes, O., Bogdan, P. A., Brenninkmeijer, C., Davidson, S., Fellows, D., Gait, A., ... Furber, S. B. (2018). spynnaker: A software package for running pynn simulations on spinnaker. *Frontiers in Neuroscience*, *12*. Retrieved from https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2018.00816 doi: 10.3389/fnins.2018.00816

Rostami, A., Vogginger, B., Yan, Y., & Mayr, C. G. (2022). E-prop on spinnaker 2: Exploring online learning in spiking rnns on neuromorphic hardware. *Frontiers in Neuroscience*, *16*. Retrieved from https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2022.1018006 doi: 10.3389/fnins.2022.1018006

Routti, J. T. (1969). SAMPO, A Fortran IV Program for Computer Analysis of Gamma Spectrafrom Ge(Li) Detectors, and for Other Spectra with Peaks. (UCRL-19452). Retrieved 2021-07-26, from https://www.osti.gov/biblio/878125 doi: 10.2172/878125

Rueckauer, B., Lungu, I.-A., Hu, Y., & Pfeiffer, M. (2016). *Theory and tools for the conversion of analog to spiking convolutional neural networks.*

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., & Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, *11*. Retrieved from https://www.frontiersin.org/articles/10.3389/fnins.2017.00682 doi: 10.3389/fnins.2017.00682

Saunders, D. J., Siegelmann, H. T., Kozma, R., et al. (2018). Stdp learning of image patches with convolutional spiking neural networks. In *2018 international joint conference on neural networks (ijcnn)* (pp. 1–7).

Saunders, D. J., Sigrist, C., Chaney, K., Kozma, R., & Siegelmann, H. T. (2020). Minibatch processing for speed-up and scalability of spiking neural network simulation. In *2020 international joint conference on neural networks (ijcnn)* (p. 1-8). doi: 10.1109/IJCNN48605.2020.9207452

Sayiner, N., Sorensen, H., & Viswanathan, T. (1996). A level-crossing sampling scheme for a/d conversion. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *43*(4), 335-339. doi: 10.1109/82.488288

Sboev, A., Serenko, A., Rybka, R., & Vlasov, D. (2020). Solving a classification task by spiking neural network with stdp based on rate and temporal input encoding. *Mathematical Methods in the Applied Sciences*, *43*(13), 7802–7814.

Schraml, S., Belbachir, A., & Brändle, N. (2010). A real-time pedestrian classification method for event-based dynamic stereo vision. In *2010 ieee computer society conference on computer vision and pattern recognition - workshops* (p. 93-99). doi: 10.1109/CVPRW.2010.5543775

Selim, Y., Lasheen, Y., Hassan, M., & Zakla, T. (2013, 01). Removal of alcyon ii, cgr, mev ⁶⁰co teletherapy head and evaluation of exposure dose. *Journal of Environmental Protection*, *04*, 1435-1440. doi: 10.4236/jep.2013.412164

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2019). Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, *13*. Retrieved from https://www.frontiersin.org/articles/10.3389/fnins.2019.00095 doi: 10.3389/fnins.2019.00095

Shokhirev, K. N., Cosofret, B. R., King, M., Harris, B., Zhang, C., & Masi, D. (2012). Enhanced detection and identification of radiological threats in cluttered environments. In *2012 ieee conference on technologies for homeland security (hst)* (p. 502-506). doi: 10.1109/THS.2012.6459899

Shrestha, S. B., & Orchard, G. (2018). SLAYER: spike layer error reassignment in time. *CoRR*, *abs/1810.08646*. Retrieved from http://arxiv.org/abs/1810.08646

Shreyas, V., Bharadwaj, S. N., Srinidhi, S., Ankith, K. U., & Rajendra, A. B. (2020). Self-driving cars: An overview of various autonomous driving systems. In M. L. Kolhe, S. Tiwari, M. C. Trivedi, & K. K. Mishra (Eds.), *Advances in data and information sciences* (pp. 361–371). Singapore: Springer Singapore.

Siniscalchi, S. M., Svendsen, T., & Lee, C.-H. (2014). An artificial neural network approach to automatic speech processing. *Neurocomputing*, *140*, 326-338. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231214004007 doi: https://doi.org/10.1016/j.neucom.2014.03.005

SpiNNaker Technical Documents. (n.d.). Retrieved 2024-02-27, from http://spinnakermanchester.github.io/docs/

Stimberg, M., Brette, R., & Goodman, D. F. (2019, aug). Brian 2, an intuitive and efficient neural simulator. *eLife*, *8*, e47314. Retrieved from https://doi.org/10.7554/eLife.47314 doi: 10.7554/eLife.47314

Strubell, E., Ganesh, A., & McCallum, A. (2019). *Energy and policy considerations for deep learning in nlp.*

Suh, Y. (2007, 04). Send-on-delta sensor data transmissionwith a linear predictor. *Sensors*, *7*. doi: 10.3390/s7040437

Sullivan, C., & Stinnett, J. (2015). Validation of a bayesian-based isotope identification algorithm. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *784*, 298-305. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168900214014272 (Symposium on Radiation Measurements and Applications 2014 (SORMA XV)) doi: https://doi.org/10.1016/j.nima.2014.11.113

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks*, *111*, 47-63. Retrieved from https://www.sciencedirect.com/science/article/pii/S0893608018303332 doi: https://doi.org/10.1016/j.neunet.2018.12.002

Thiele, J. C., Bichler, O., & Dupret, A. (2018). Event-based, timescale invariant unsupervised online deep learning with stdp. *Frontiers in Computational Neuroscience*, *12*. Retrieved from https://www.frontiersin.org/articles/10.3389/fncom.2018.00046 doi: 10.3389/fncom.2018.00046

Tsividis, Y. (2010). Event-driven data acquisition and continuous-time digital signal processing. In *Ieee custom integrated circuits conference 2010* (p. 1-8). doi: 10.1109/CICC.2010.5617618

Tsoulfanidis, N., & Landsberger, S. (2021). *Measurement and detection of radiation*. CRC press.

United States Nuclear Regulatory Conmission. (2020). *What is a geiger counter?* Retrieved from https://www.nrc.gov/reading-rm/basic-ref/students/science-101/what-is-a-geiger-counter.html (Last accessed on 1 Nov 2023)

V, V., C, R. A., Prasanna, R., Kakarla, P. C., PJ, V. S., & Mohan, N. (2022). *Implementation of tiny machine learning models on arduino 33 ble for gesture and speech recognition.*

van Schaik, A. (2010). Adaptive sound localization with a silicon cochlea pair. *Frontiers in Neuroscience*, *4*. Retrieved from https://www.frontiersin.org/articles/10.3389/fnins.2010.00196 doi: 10.3389/fnins.2010.00196

van Schaik, A., Chan, V., & Jin, C. (2009). Sound localisation with a silicon cochlea pair. In *2009 ieee international conference on acoustics, speech and signal processing* (p. 2197-2200). doi: 10.1109/ICASSP.2009.4960054

Vigneron, A., & Martinet, J. (2020). A critical survey of stdp in spiking neural networks for pattern recognition. In *2020 international joint conference on neural networks (ijcnn)* (p. 1-9). doi: 10.1109/IJCNN48605.2020.9207239

Waqar, D. M., Gunawan, T. S., Morshidi, M. A., & Kartiwi, M. (2021, August). Design of a Speech Anger Recognition System on Arduino Nano 33 BLE Sense. In *2021 IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)* (pp. 64–69). Bandung, Indonesia: IEEE. Retrieved 2024-02-29, from https://ieeexplore.ieee.org/document/9526323/ doi: 10.1109/ICSIMA50015.2021.9526323

Warden, P., & Situnayake, D. (2020). *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly. Retrieved from https://books.google.com/books?id=sB3mxQEACAAJ

Weikersdorfer, D., Hoffmann, R., & Conradt, J. (2013). Simultaneous localization and mapping for event-based vision systems. In M. Chen, B. Leibe, & B. Neumann (Eds.), *Computer vision systems* (pp. 133–142). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wong, T. (2023). *Fukushima: What are the concerns over waste water release?* Retrieved from https://www.bbc.co.uk/news/world-asia-66106162 (Last accessed on 23 Oct 2023)

World Nuclear Association. (2022). *Chernobyl accident 1986.* Retrieved from https://world-nuclear.org/information-library/safety-and-security/safety-of-plants/chernobyl-accident.aspx (Last accessed on 23 Oct 2023)

World Nuclear Association. (2023). *Fukushima daiichi accident.* Retrieved from https://world-nuclear.org/information-library/safety-and-security/safety-of-plants/fukushima-daiichi-accident.aspx (Last accessed on 23 Oct 2023)

Wu, Y., Deng, L., Li, G., Zhu, J., & Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, *12*. Retrieved from https://www.frontiersin.org/articles/10.3389/fnins.2018.00331 doi: 10.3389/fnins.2018.00331

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., & Shi, L. (2019). Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 1311–1318).

Xie, S., Jones, E., Marsden, E., Baistow, I., Furber, S., Mitra, S., & Hamilton, A. (2022). Unsupervised stdp-based radioisotope identification using spiking neural networks implemented on spinnaker. In *2022 8th international conference on event-based control, communication, and signal processing (ebccsp)* (p. 1-8). doi: 10.1109/EBCCSP56922.2022.9845586

Yan, J., & Glaser, A. (2015, 09). Nuclear warhead verification: A review of attribute and template systems. *Science & Global Security*, *23*, 157-170. doi: 10.1080/08929882.2015.1087221

Yoshida, E., Shizuma, K., Endo, S., & Oka, T. (2002). Application of neural networks for the analysis of gamma-ray spectra measured with a ge spectrometer. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *484*(1), 557-563. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168900201019623 doi: https://doi.org/10.1016/S0168-9002(01)01962-3

Zhang, D., Pee, L., & Cui, L. (2021). Artificial intelligence in e-commerce fulfillment: A case study of resource orchestration at alibaba's smart warehouse. *International Journal of Information Management*, *57*, 102304.

Zhang, S., Marsden, E., & Goulermas, J. Y. (2022). Isotope identification using artificial neural network ensembles and bin-ratios. *IEEE Transactions on Nuclear Science*, *69*(6), 1194-1202. doi: 10.1109/TNS.2022.3176586

Zhou, S., Chen, Y., & Ye, Q. (2019). Deep scnn-based real-time object detection for self-driving vehicles using lidar temporal data. *CoRR*, *abs/1912.07906*. Retrieved from http://arxiv.org/abs/1912.07906

Åarzén, K.-E. (1999). A simple event-based pid controller. *IFAC Proceedings Volumes*, *32*(2), 8687-8692. Retrieved from https://www.sciencedirect.com/science/article/pii/S1474667017574820 (14th IFAC World Congress 1999, Beijing, Chia, 5-9 July) doi: https://doi.org/10.1016/S1474-6670(17)57482-0