Implementing a Spiking Neural Network with floating-point neurons into GeNN to compare the performance to that of a Few Spikes neural network in computer vision.

Thomas Shoesmith [CandNo. 198687]
Supervisor: Thomas Nowotny

Interim Report
Final Year Project
Computer Science with Artificial Intelligence

Department of Informatics
University of Sussex
11/11/2021

Contents

1 Introduction **2**

1.1 Project Introduction . . . 2

1.2 Relevance . . . 2

2 Aims and Objectives **2**

2.1 Implement a network of Few Spikes neurons for image recognition . . . 2

2.1.1 Start with a more basic data set (i.e. MNIST) . . . 2

2.1.2 Finish with a more complex data set (i.e. CIFAR-10 or Omniglot) . . . 3

2.2 Implement a network of floating-point neurons for image recognition . . . 3

2.2.1 Start with a more basic data set (i.e. MNIST) . . . 3

2.2.2 Finish with a more complex data set (i.e.CIFAR-10 or Omniglot) . . . 3

2.3 Assess performance of different spike coding methods . . . 3

2.4 Extension . . . 3

2.4.1 Investigating primacy code as a potential for more efficient neural coding . 3

3 Professional considerations **3**

3.1 BCS Code of Conduct . . . 3

3.1.1 Public Interest . . . 3

3.1.2 Professional Competence and Integrity . . . 4

3.2 Ethical Issues . . . 5

4 Requirements analysis **5**

4.1 Background Research . . . 5

4.1.1 Spiking Neural Networks . . . 5

4.1.2 GPU enhanced Neuronal Network . . . 7

4.1.3 Few Spikes Conversion Neurons . . . 7

4.1.4 Floating-Point Conversion Neurons . . . 8

4.1.5 Primacy Code . . . 9

4.1.6 Spiking Neural Network training databases . . . 9

4.2 Resources Required . . . 9

4.3 Progress so far . . . 10

5 Project Timeline **10**

6 Log **11**

7 Appendix **11**

7.1 Project Proposal Document . . . 11

1 Introduction

1.1 Project Introduction

Despite yielding impressive results from image classification to playing against humans in games such as AlphaGo[1], Artificial Neural Networks (ANN) require a significant amount of energy when compared to the human brain. AlphaGo as an example required approximately 1MW of power to run[2], meanwhile the energy consumed by an average human brain is equivalent to approximately 20W[3] giving AlphaGo, 50,000 times more energy in this competition. This therefore brings us onto the latest challenge in modern machine learning, the focus on the efficiency of algorithms, so that machine learning can be run on more portable devices or scaled up without the requirement of large energy resources.

Spiking Neural Networks (SNN) have demonstrated the potential to offer a more energy efficient alternative, compared to ANN, mainly due to their lower computational requirements and the sparser spike-based communications between neurons[4]. However, training large SNN from scratch remains challenging and converting a high performing ANN into a similarly performing SNN has typically involved encoding ANN activations in the firing rate of the spiking neurons[5]. The resulting SNN from this conversion typically requires a relatively greater number of spikes to generate a similar performance, with the larger numbers of spikes outweighing the energy savings of the SNN.

In a recent paper by Christoph Stöckl and Wolfgang Maass[6], they introduce a new method of ANN to SNN conversion titled: ”Few Spikes (FS) conversion”. FS treats each neuron’s output spikes as a binary code representing the activation of the ANN neuron and, by using the timing of the spikes to hold additional information, fewer spikes are needed and each input can be presented using fewer time steps.

This project will build on the research from this paper to explore further optimisations to these binary codes, potentially allowing the number of spikes and time steps to be further reduced and overall efficiency improved. The SNN performance will be evaluated through the accuracy of image classification on a selection of benchmark data sets including MNIST [7], CIFAR-10[8] and Omniglot[9].

1.2 Relevance

Artificial Neural Networks (ANN) make up a significant portion of the Computer Science with Artificial Intelligence degree, it is the building blocks behind computer vision, machine learning, NLP etc. Spiking Neural Networks are an extension from ANN with an aim to increase the efficiency of neural networks, allowing them to be more accessible on smaller, handheld devices or just more efficient on scaled up neural networks.

2 Aims and Objectives

2.1 Implement a network of Few Spikes neurons for image recognition

The first objective is to implement an already published method for ANN-SNN conversion, a Few Spikes neuron, within GeNN. This will require two neurons to be developed, one neuron will take a floating-point value, sum up the weights and bias and convert this into a spike train. The second neuron will take a spike train from another Few Spikes neuron, sum up the weights and bias and return the updated spike train.

2.1.1 Start with a more basic data set (i.e. MNIST)

A developed Few Spikes neuron will then be implemented into a Spiking Neural Network. The second objective after developing this neuron would therefore be to implement the Few Spikes neuron into a relatively simple network to be trained on a relatively basic data set. The data set provisionally chosen is the MNIST data set due to several properties.

The first reason is due to its popularity, MNIST is a widely practiced data set with an abundance of neural networks implemented to solving this, therefore there is a good chance of developing a successful Spiking Neural Network with a Few Spikes neuron. The second reason is the data set size, MNIST contains 60,000 training and 10,000 testing images allowing for plenty of training runs. MNIST images are also a relatively small in size, measuring in at 28 by 28 pixels, allowing for a relatively small neural network.

2.1.2 Finish with a more complex data set (i.e. CIFAR-10 or Omniglot)

After successfully implementing a Few Spikes neuron into a neural network and training it on a relatively basic neural network such as MNIST, the next step will be to implement a Few Spikes neuron into a more complex Spiking Neural Network and train this up on a more complex data set such as CIFAR-10 or Omniglot.

By implementing this neuron into a more complicated neural network, it will allow for a greater demonstration of the neuron's performance compared to later developed neurons.

2.2 Implement a network of floating-point neurons for image recognition

Once a successful development of the Few Spikes neuron and implementation into various neural networks for different data sets is achieved, the next objective is to develop the floating-point neuron within GeNN.

Few Spikes neurons used fixed-point arithmetic to store the value passed through, floating-point neurons have the potential advantage with the use of floating-point arithmetic being used, allowing for a greater range of value representation.

2.2.1 Start with a more basic data set (i.e. MNIST)

Similarly to section 2.1.1, once the neuron has been developed, it will first be implemented into a relatively basic Spiking Neural Network where it will be used on image recognition with the same basic data set used in section 2.1.1.

2.2.2 Finish with a more complex data set (i.e.CIFAR-10 or Omniglot)

Similarly to section 2.1.2, once a successful implementation of the floating-point neuron into a relatively basic image recognition has been achieved, the floating-point neuron will need to be implemented into a more complex Spiking Neural Network for training on the same data sets used in section 2.1.2.

2.3 Assess performance of different spike coding methods

Once the four implementations of the two neurons are completed, tests can be conducted comparing the performance between the two neurons in different environments. These tests will compare the accuracy, time taken for training, spike train length (the number of spikes needed to represent a value) and any other metrics which might present themselves as useful.

2.4 Extension

2.4.1 Investigating primacy code as a potential for more efficient neural coding

Another approach into neural coding may come from a more biological influence in the form of primacy code. Odor detection from the olfactory system is achieved regardless of concentration and attention from the source, this ability suggests a unique variant of neural activity, and a potential for a new approach of neural coding. [10]

3 Professional considerations

3.1 BCS Code of Conduct

The relevant sections of the BCS Code of Conduct[11] are listed below, with an explanation as to how this will be dealt with during the project

3.1.1 Public Interest

a. have due regard for public health, privacy, security and wellbeing of others and the environment

The public health and well being would be unaffected by this project due to the focus being on efficiency and performance with no direct interaction between the public and the project. All data used within the project will be sourced from publicly available repositories.

As the project will be looking into the efficiency of alternative coding methods, there is no direct security risks.

b. have due regard for the legitimate rights of Third Parties
This project will primarily be working with GeNN where it will be purely be used for building models within the simulation environment to compare performance of different models. This would not fall within any legitimate rights issues.

c. conduct your professional activities without discrimination on the grounds of sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age or disability, or of any other condition or requirement
All activities will be done at a high level of professional standard. This project will be looking at software performances, so the only relevance this section has would be during interactions with supervisors or other academics.

d. promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society wherever opportunities arise
There will be limited to no opportunities for this section within the project, however, any correspondence will be dealt with equally.

3.1.2 Professional Competence and Integrity

a. only undertake to do work or provide a service that is within your professional competence.
The project proposal and outline has been discussed with the supervisor and the area of research is expected to be within the expected professional competence.

b. NOT claim any level of competence that you do not possess.
This project is a continuation from a University of Sussex JRA, the understanding of the project has been researched and previously practiced. No level of competence has been falsely claimed.

c. develop your professional knowledge, skills and competence on a continuing basis, maintaining awareness of technological developments, procedures, and standards that are relevant to your field.
Throughout this project, research will be required into the fields relevant to the topic, in this case Spiking Neural Networks and GeNN. Correspondence with the project's supervisor and the GeNN GitHub will maintain an up to date understanding on the standards and procedures within this field.

d. ensure that you have the knowledge and understanding of Legislation and that you comply with such Legislation, in carrying out your professional responsibilities.
GeNN is licenced under GNU General Public License, version 2, which allows for changing of the free software and permission to use the code as long as any published works give reference to the original source code. [12]

MNIST is licenced under GNU General Public License, version 3, which allows for changing of the free software and permission to use the code as long as any published works give reference to the original source code. [13]

CIFAR-10 and Omniglot is under a MIT License, which allows for "without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so". [14] [15]

e. respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work.
Throughout the project, feedback and advice will be requested from both the supervisor as well as other academics within the field, their feedback and support will always be looked at and used to improve upon.

f. avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction.
A high professional standard will be maintained throughout this project both within the project and with correspondence. There will be no sensitive information involved that could be mismanaged.

g. reject and will not make any offer of bribery or unethical inducement.
No bribery or unethical inducement will be offered or accepted.

3.2 Ethical Issues

As this project does not involve the collection of data from human participants, there is no need for an Ethical review.

However, there are a couple of topics of discussion around the ethics of the potential application of this project. Whilst this project does not involve any participants or sensitive information, there is a potential risk of the project's outcomes being abused. This project will be looking into the different methods of Spiking Neural Network coding and the difference in both performance and efficiency. Although this directly has practically no ethical issues, these efficient methods could have the potential to be implemented into a system with malicious intent.

This however, is highly unlikely, primarily because this project when completed would be far from application, but also because similar research within this area is ongoing. The data used within this project is all publicly available and the project will also not be published.

4 Requirements analysis

4.1 Background Research

4.1.1 Spiking Neural Networks

To understand a Spiking Neural Network, we must first be clear as to what an Artificial Neural Network is.

An Artificial Neural Network is a type of neural network which is a bio-inspired method of machine learning influenced by the biological neural networks found in animal brains. These digital minds are made up of an interconnected group of nodes which form connections with weights and biases. These nodes are often made up of different layers which often provide different functionality. The way these nodes are interconnected in an important feature of the neural network; each connection between nodes holds a bias, and each node holds a weight. A bias is a value that is added onto the signal to be passed through to the next layer whereas a weight controls the amount of influence the oncoming signal has on the node.

Artificial Neural Networks are the current, primary method of machine learning due to their high accuracy and compatibility with current computer architecture; they are able to solve tasks to an accuracy and performance similar to, and sometimes even greater than that of their human counterparts, A good example of this is AlphaGo's victory over Lee Sedol in 2016[16].

However, one area where Artificial Neural Networks do not have an advantage is in their resources requirements, despite yielding relatively impressive results, these neural networks require a relatively large computational resources in order to operate. AlphaGo was able to defeat Lee Sedol 4-1 in 2016[16], however, when we compare the estimated energy consumption between the two, we see a different side to this story. It is believed the human brain only uses 20W, a relatively tiny amount when compared to AlphaGo's estimated 1MW of power, giving AlphaGo a 50,000 times power advantage.

Spiking Neural Networks offer a potentially more energy efficient method of machine learning. A Spiking Neural Network takes on an even more biological approach to an artificial mind by imitating spikes between nodes instead of sending floating-point values (figure 1). At the most basic level is a Leaky Integrate and Fire (LIF) neuron (figure 2). This model is a simulation of a simplified biological neuron where there is a membrane voltage, a threshold voltage and a resting voltage. An input is added to the membrane voltage, if the input is a large input signal, then a large voltage is added, if it is a small input signal, then a small voltage is added to the membrane voltage. Over time, this membrane voltage will decay until it reaches the resting voltage. If there is a large enough signal, or enough smaller signals that combine together to cause the membrane voltage to become equal to or greater than the threshold voltage, then the membrane voltage is reset to the resting voltage, a signal is outputted and there will be a refractory period where another signal cannot be sent for a period of time.

Both figures have been influenced from an article by Paredes-Vallés[17].

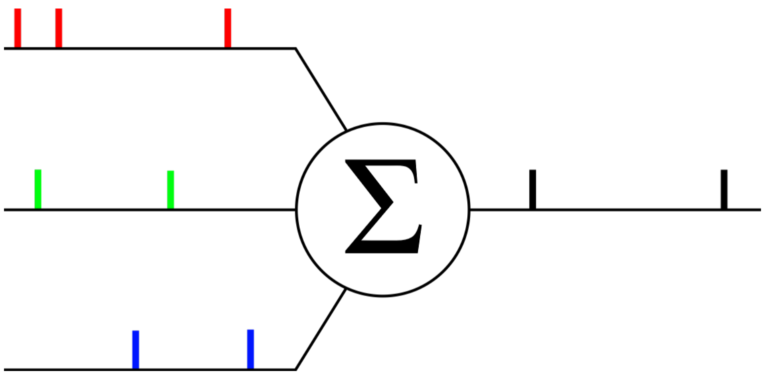


Figure 1: A Spiking Neural Network with inputs from three neurons

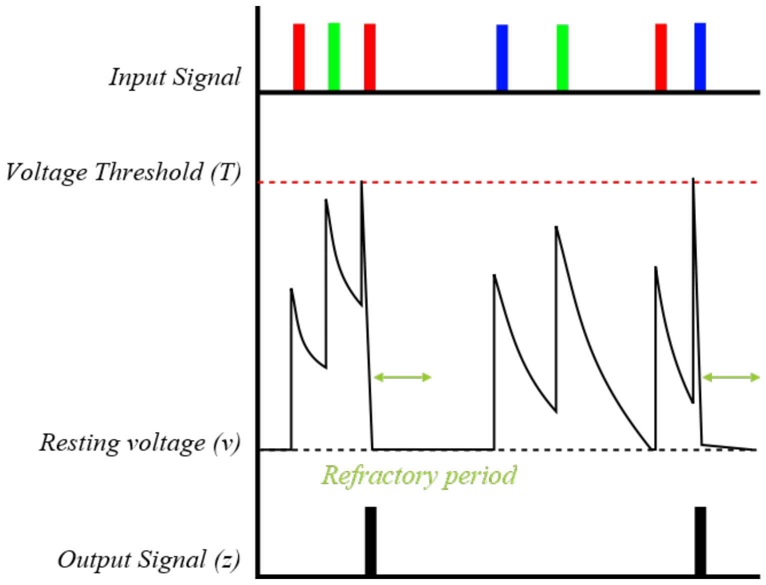


Figure 2: A LIF neuron, taking in inputs from three neurons and outputting a signal once the membrane voltage surpasses the voltage threshold

One area of research within Spiking Neural Networks is the method of coding the spikes between neurons, there are 4 main methods currently being used: Rate Coding, Population coding, Sparse coding and temporal coding.

Temporal coding will be the primary focus of this project. Temporal coding uses the timing of each spike to carry additional information which in turn can reduce the number of spikes needed.

4.1.2 GPU enhanced Neuronal Network

One current disadvantage of Spiking Neural Networks is their inefficiency to run on conventional computer hardware, Spiking Neural Networks reach there optimal performance on neuromorphic hardware. However, there are simulators available to run Spiking models on conventional computer architecture.

The University of Sussex have their own Spiking Neural Network simulator titled GPU enhanced Neural Network (GeNN) [18]. GeNN was first proposed in 2016 as a method of using the computational speeds found within graphical processing units (GPUs) to help generate large-scale spiked neural networks with respectable run times.

Since 2016, GPUs can offer general purpose computing thanks to NVIDIA's compute unified device architecture (CUDA), allowing them to be used across a wider range of applications. CUDA allows for C - like code, which can be executed efficiently on NVIDIA's CUDA GPUs [18]. GeNN is able to take advantage of CUDA by compiling some of the code into C, it is able to run significantly faster, allowing for a significant reduction in processing time.

This project will use GeNN to develop neurons that can later be implemented into relatively large Spiking Neural Network models, with the efficiency GeNN is able to produce, model training times hopefully can be of a manageable period.

4.1.3 Few Spikes Conversion Neurons

Directly training a Spiking Neural Network remains a difficult challenge, a current method for producing Spiking Neural Networks is to convert pre-existing Artificial Neural Networks into Spiking Neural Networks by using a variety of conversion method. Few Spikes (FS)[6] offers a conversion method from a floating-point value to a spike train using temporal coding.

A Few Spikes neuron operates in a similar method to the leaky integrate and fire neuron discussed in section 4.1.1 however, the main difference is that with Few Spikes the membrane voltage does not degrade over time and the spikes received / returned are stored in a fixed-point binary representation positioning, rather than frequency.

Within a Few Spikes neuron, there are two global parameters: K and alpha. K is the fixed time frame / spike train length to represent a value. Alpha is the maximum value that can be represented by that neuron / network. Within the neuron, there are several local variables: v, t, T, h and z. v represents the membrane potential, t represents the current time step (this will range from t = 1..K), T represents the threshold value, h is the membrane potential reset and finally, z is the output spike train.

For this project, we will primary be using a ReLU activation function, due to the nature of FS-neurons, the values of T and h will be identical, but for demonstrative purposes, T and h have been defined separately.

The python code for a Few Spikes neuron would be as follows:

```python
# Few Spikes Neuron
# x = input
# K = timesteps K = 1, ... t
# alpha = value representation range (cap)

import numpy as np

def fs(x, K = 10, alpha = 25):
    t = 0
    fx = 0
    v = np.zeros(K)
    z = np.zeros(K)

    T = alpha * 2**(-K) * np.array([float(2 ** (K - i))
                                    for i in range(1, K + 1)]).
  astype(np.float32)
```

```python
    h = alpha * 2**(-K) * np.array([float(2 ** (K - i))
                                    for i in range(1, K + 1)]).astype(np.float32)

    v[0] = x

    while t < K:
        # spike if voltage > threshold, reset if spike.
        if v[t] >= T[t]:
            z[t] = 1
            v[t] = v[t] - h[t]

        # copy over value once reduced.
        if t + 1 < K:
            v[t + 1] =  v[t]

        # sum voltage if spike
        fx += z[t] * h[t] #* 0.5
        t += 1

    return z, fx # outputs tuple of spike train and sum (sum for configuration)
```

Listing 1: Python Code for FS-neuron

Few Spikes offers a significant reduction in the number of spikes needed whilst being able to achieve an accuracy similar to that of Artificial neural network models [6].

4.1.4 Floating-Point Conversion Neurons

Floating-point offers an alternative to Few Spikes conversion by using floating-point arithmetic compared to fixed-point arithmetic.

Instead of having the values of T and h being fixed across all neuron within a network, they instead can be changed to the range best covers the value needing to be represented, this has both its advantages and disadvantages. An advantage of using this method is that it allows for smaller floating-point values to have improved accuracy, this is due to the increased range through the use of an exponent. The disadvantages however are that part of the spikes are taken up with the data representing the exponent, this means that the overall accuracy is reduced, especially for larger values, as less spikes are holding information about the value directly (the mantissa).

The methods for coding this neuron style are similar, however the exponent needs to be calculated, and the starting position for the T value also need to be calculated.

The python code for a Floating-point neuron would be as follows (note that in this example, the exponent is sent ahead of the mantissa):

```python
# Floating-point Neuron
# x = input
# K = timesteps K = 1, ... t
# elim = length of exponent
# w = weight on neuron
# alpha = value representation range (cap)

import numpy as np

def fp_d2s(x, K = 8, elim = 4, w = 1, alpha = 20):
    if x > alpha:
        x = alpha

    man_start = alpha
    exponent = 0

    while man_start > x and exponent < 2**elim - 1:
        exponent += 1
        man_start = man_start / 2

    T = man_start * 2* 2**(-K) * np.array([float(2 ** (K - i))
        for i in range(1, K + 1)]).astype(np.float32)

    exponent = format(abs(exponent), "b")

    while len(exponent) < elim:
        exponent = "0" + exponent

    # mantessa
    z = []
    for t in T:
        if t <= x:
            x -= t
            z.append("1")
        else:
            z.append("0")

    man = ''.join(z)

    return exponent + man
```

Listing 2: Python Code for Floating-point neuron

One issue that presents itself when trying to code a floating-point neuron is the time step coding requirement. With a Few Spikes neuron, the code is run at each time step and an output is generated with each time step. However, with a floating-point neuron, the production of the exponent and the mantissa are done in one time step when the final spike is received, this issue will need to be resolved before being implemented into a Spiking Neural Networks.

4.1.5 Primacy Code

Humans use five major senses: hearing, sight, taste, touch and smell, it is the latter that primacy coding looks into. The senses from the olfactory system are processed regardless of concentration on the source. This ability for odor detection to be done at this level suggests an early odor-evoked neural activity [19]. It is this neural activity that will be investigated to see if there is a potential for a new method of neural coding that could be implemented.

4.1.6 Spiking Neural Network training databases

There are three main data sets to be used across this project for training the Spiking Neural Networks, they range from relatively small, allowing for a relatively simple neural model to be trained, to relatively complex data sets which will require more complex models.

MNIST [7] (Modified National Institute of Standards and Technology database) is a data set containing 60,000 training and 10,000 testing images of various handwritten digits. The images are size-normalised, centered and have the relatively small size of 28 by 28. The relatively small size of these images makes it ideal for training small Spiking Neural Networks on due to the relatively small input layer needed.

CIFAR-10 [8] is a database consisting of 60,000 32x32 colour images that can be classified into 10 classes with 6,000 images per class. This is a small jump up from MNIST so will allow for a larger training network.

Omniglot[9] is a more complex version of MNIST. MNIST only contains 26 different alphabetical symbols in different styles. Omniglot in comparison contains 340 diffent writing styles, including Abjads, Alphabets, Abugidas, Syllabaries and Semanto-phonetic scripts[20]. Whilst MNIST has 60,000 training data, Omniglot has far less, so training on this data set offers a different challenge.

4.2 Resources Required

This project is primarily programming focused, therefore the hardware resources needed is access to a computer which can run GeNN and Python. Unfortunately, the University computers do not allow GeNN to run, so most of the light computational work will be done on a personal laptop, any larger and heavier computational work can be done with a personal desktop equipped with an GTX 1050TI, a GPU who's performance has been measured as okay by PyGeNN [21]. If more computational processing is required, then there is access to GPU clusters within the GeNN group for the training of larger models.

The software requirements involve the use of GeNN (GPU enhanced Neural Networks) which is a University developed software at Sussex, support for this software is readily available with the developers still being at the University of Sussex.

4.3 Progress so far

Project so far has been background research, basic implementation of neurons and GeNN tutorials.

Background research has involved both research done over the summer with the Sussex Junior Research Associate (JRA) scheme, as well as research over the first 5 weeks of term. The areas of research have started at the beginning with understanding the basics of Spiking Neural Networks, to understanding different methods of neural coding and neuron times to the advantages and disadvantages of Spiking Neural Networks as a whole.

Python implementation of the Few Spikes neuron, Floating-point neuron and a basic comparison of the two has been done over the summer with the JRA. GeNN was first introduced over the JRA, however the understanding needed to implement a Few Spikes or floating-point neuron has never achieved. Understanding of GeNN needs to be learnt for this project to be successful, therefore a reintroduction of GeNN and research into GeNN has already been started to help improve and gain a greater understanding of GeNN.

5 Project Timeline

The aim for this project is to get the development and implementation of the code to be done within the first term for two reasons: the first being to allocate time within the second term to write up the findings, evaluate performance and to explore the extension task of primacy code, the second reason is to allow a buffer in case the implementation and development take longer than expected, therefore, time within the second time can be allocated to allow for development.

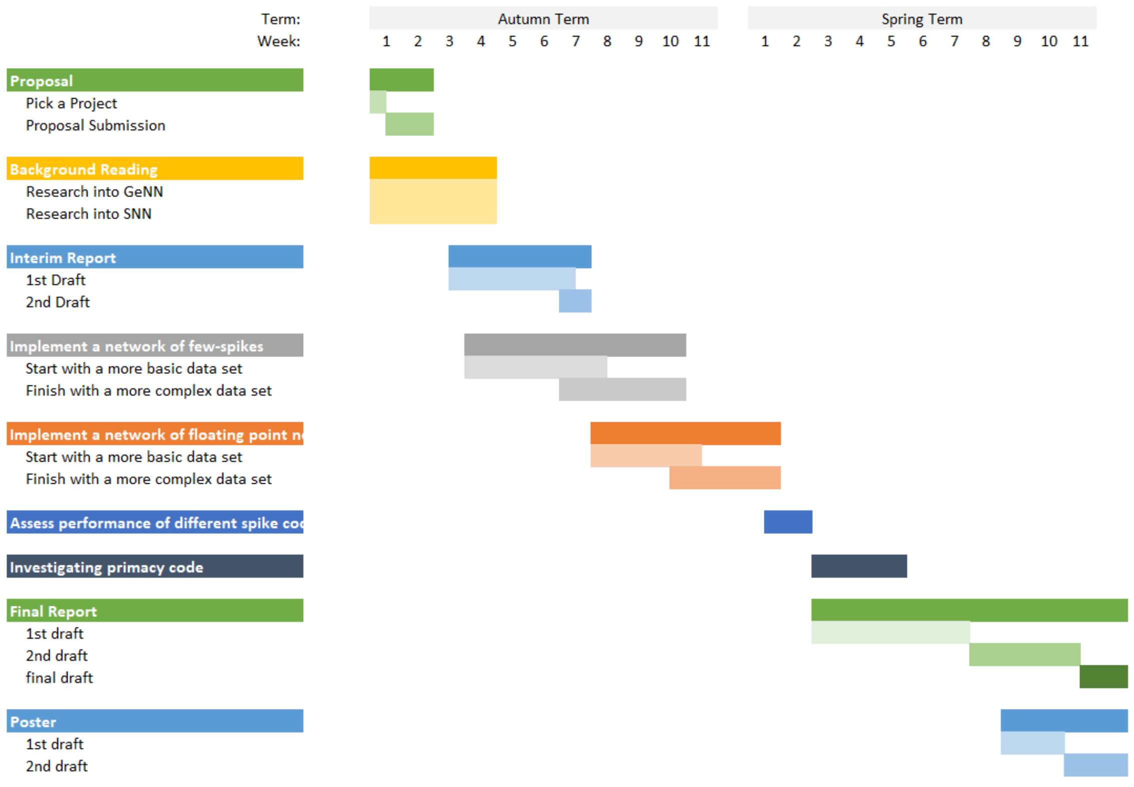


Figure 3: A Gnatt chart to show the projects timeline

6 Log

Date	Summary of Meet
16/09/2021	First meeting about FYP, talked about project outline, primary and secondary objectives and extension task. Will conduct research in the area of these goals and in the area of the project as a whole
29/09/2021	Meeting to discuss current progress, discussed project proposal form and converting the current version (in word) into Latex. Discussion into GeNN and understanding GeNN code
06/10/2021	went over the converted Latex project proposal form, discussion of FS-code
20/10/2021	limited progress of the code side due to ill health, talked about GeNN and code on Github
29/10/2021	Talking about the interim report, walked through GeNN code that Jamie had used on the GitHub, outlined aims to go though the GeNN tutorials
05/11/2021	Talking about the interim report, limited progress of the code side due to ill health

7 Appendix

7.1 Project Proposal Document

Final Year Project
Project Proposal

Thomas Shoesmith [CandNo. 198687]
Supervisor: Thomas Nowotny

October 2021

Implementing a spiking neural network with floating-point neurons into GeNN to compare the performance to that of a few-spikes neural network in computer vision.

Abstract

Modern Artificial Neural Networks (ANN) still require a lot of processing power. For instance, AlphaGo used approximately 1MW of power to run, whilst a human brain only uses 20W. [1]

Unlike units in ANNs which continuously exchange real-valued activations, neurons in spiking neural networks communicate infrequently using binary events called spikes like neurons in biological brains. This reduced communication means that, theoretically, SNNs could offer a much more energy- efficient alternative to ANNs.

Training SNNs from scratch remains tricky but there are several promising approaches for converting already trained ANNs to SNNs. The current limitation with SNNs is that they require a lot of signal spikes in order to achieve the same task performance as ANNs; the aim for this final year project will be to investigate a potentially more efficient means of spike based encoding called floating-point conversion against an existing method and to evaluate their performance using a computer vision trained neural network using the respective encoding method.

1 Aims

Over the summer I was able to carry out undergraduate research into Spiking Neural Networks (SNN) where I primarily investigated a floating-point method of spike encoding. The purpose of this Final Year Project (FYP) is to build on the knowledge gained over the summer to more closely compare the performance of different spike encoding and to implement a floating-point method into GeNN.

2 Primary Objectives

1. Implement a network of few-spikes [2] neurons for image recognition
   1.1 Start with a more basic data set (i.e. MNIST [3])
   1.2 Finish with a more complex data set (i.e. CIFAR-10 [4] or Omniglot [5])
2. Implement a network of floating point neurons for image recognition
   2.1 Start with a more basic data set (i.e. MNIST)
   2.2 Finish with a more complex data set (i.e. CIFAR-10 or Omniglot)
3. Assess performance of different spike coding methods

3 Extension

1. Investigating primacy code [6] as a potential for more efficient spike coding

4 Relevance

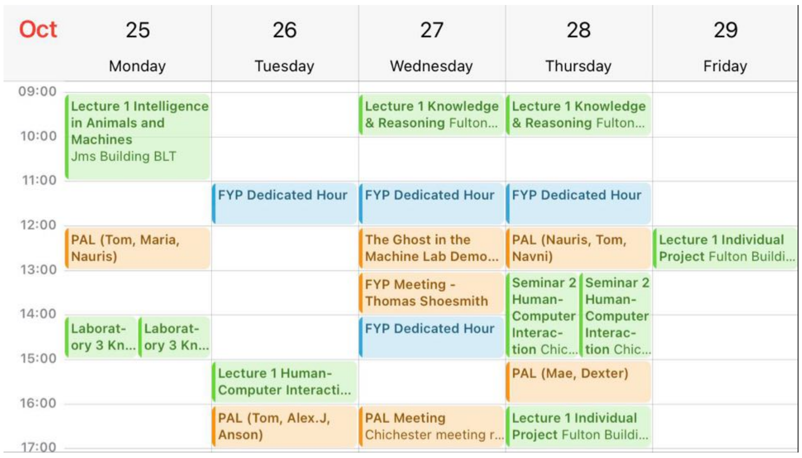
Artificial Neural Networks (ANN) make up a large portion of my Computer Science with Artificial Intelligence degree, it is the building blocks behind computer vision, machine learning, NLP etc. SNN are an extension from ANN with an aim to increase the efficiency of neural networks, allowing them to be more accessible on smaller, handheld devices or just more efficient on scaled up neural networks.

5 Resources Required

This project is primarily programming focused, therefore the only hardware resources needed is access to a computer which can be done either from the Chichester Labs or via a personal computer. The software requirements involve the use of GeNN (GPU enhanced Neural Networks) which is a University developed software at Sussex, I have already been introduced to GeNN through my summer research, however, will need to improve on my understanding of GeNN to implement a SNN into it.

6 Timetable

A usual week I will commit to 1 weekly meeting with my supervisor on a Wednesday at 13:00. I will have 4 allocated hours during University time to work on the project, however, I expect to work on this more into the evenings so expect around 5-6 hours on some weeks.



Oct	25 Monday	26 Tuesday	27 Wednesday	28 Thursday	29 Friday
09:00	Lecture 1 Intelligence in Animals and Machines Jms Building BLT		Lecture 1 Knowledge & Reasoning Fulton...	Lecture 1 Knowledge & Reasoning Fulton...	
10:00					
11:00		FYP Dedicated Hour	FYP Dedicated Hour	FYP Dedicated Hour	
12:00	PAL (Tom, Maria, Nauris)		The Ghost in the Machine Lab Demo...	PAL (Nauris, Tom, Navni)	Lecture 1 Individual Project Fulton Buildi...
13:00			FYP Meeting - Thomas Shoesmith	Seminar 2 Human-Computer Interac-tion Chic... / Seminar 2 Human-Computer Interac-tion Chic...	
14:00	Laborat-ory 3 Kn... / Laborat-ory 3 Kn...		FYP Dedicated Hour		
15:00		Lecture 1 Human-Computer Interacti...		PAL (Mae, Dexter)	
16:00		PAL (Tom, Alex.J, Anson)	PAL Meeting Chichester meeting r...	Lecture 1 Individual Project Fulton Buildi...	
17:00					

Figure 1: An example of an average week with FYP allocated time

7 Bibliography of background reading

Currently I am reading the relevant chapters of two books at the University library; Pulsed Neural Networks by Wolfgang Maass and Christopher M. Bishop [7] and, Spiking Neuron Models by Wulfram Gerstner and Werner Kistler [8].

I will also be practicing and improving my ability to coding within GeNN with tutorials for GeNN found on their website.

For my extension task, I will be reading "Wilson CD, Serrano GO, Koulakov AA, Rinberg D. A primacy code for odor identity".

8 Interim log

Weekly meetings have been set up on a Wednesday at 13:00, these will continue each week unless either supervisor or student are unable to attend.

So far we have had a Zoom call to discuss Final Year Project ideas and outline what primary objectives and extensions should be laid out, and an in person meeting to check that the proposal outlines the aims and objects needed for this project.

References

[1] Jacqueline Ling. Power of a human brain, 2001.

[2] Christoph Stöckl and Wolfgang Maass. Classifying images with few spikes per neuron. *CoRR*, abs/2002.00860, 2020.

[3] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[4] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[5] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, December 2015.

[6] Christopher D. Wilson, Gabriela O. Serrano, Alexei A. Koulakov, and Dmitry Rinberg. A primacy code for odor identity. *Nature Communications*, 8(1):1477, Dec 2017.

[7] Wolfgang Maass and Christopher M. Bishop. *Pulsed Neural Networks*. MIT Press, Cambridge, MA, USA, 1999.

[8] Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

References

[1] D. Silver, A. Huang, C. J. Maddison, *et al.*, “Mastering the game of go with deep neural networks and tree search”, *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: `10.1038/nature16961`.

[2] J. Mattheij, *Jacques mattheij*, `https://jacquesmattheij.com/another-way-of-looking-at-lee-sedol-vs-alphago/`. (visited on 09/11/2021).

[3] J. Ling, *Power of a human brain*, `https://hypertextbook.com/facts/2001/JacquelineLing.shtml`, 2001. (visited on 09/11/2021).

[4] L. Khacef, N. Abderrahmane, and B. Miramond, “Confronting machine-learning with neuroscience for neuromorphic architectures design”, *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018. DOI: `10.1109/ijcnn.2018.8489241`.

[5] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going deeper in spiking neural networks: Vgg and residual architectures”, *Frontiers in Neuroscience*, vol. 13, 2019. DOI: `10.3389/fnins.2019.00095`.

[6] C. Stöckl and W. Maass, “Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes”, *Nature Machine Intelligence*, vol. 3, no. 3, pp. 230–238, 2021. DOI: `10.1038/s42256-021-00311-4`.

[7] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web]”, *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012. DOI: `10.1109/MSP.2012.2211477`.

[8] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-10 (canadian institute for advanced research)”, [Online]. Available: `http://www.cs.toronto.edu/~kriz/cifar.html` (visited on 09/11/2021).

[9] B. Lake, R. Salakhutdinov, and J. Tenenbaum, “Human-level concept learning through probabilistic program induction”, English (US), *Science*, vol. 350, no. 6266, pp. 1332–1338, Dec. 2015, ISSN: 0036-8075. DOI: `10.1126/science.aab3050`.

[10] C. Wilson, G. Serrano, A. Koulakov, and D. Rinberg, “A primacy code for odor identity”, *Nature Communications*, vol. 8, Nov. 2017. DOI: `10.1038/s41467-017-01432-4`.

[11] *Bcs code of conduct.* [Online]. Available: `https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/` (visited on 09/11/2021).

[12] Genn-Team, *Genn/license.txt at master · genn-team/genn*, Mar. 2017. [Online]. Available: `https://github.com/genn-team/genn/blob/master/LICENSE.txt` (visited on 09/11/2021).

[13] Sharmaroshan, *Mnist-dataset/license at master · sharmaroshan/mnist-dataset.* [Online]. Available: `https://github.com/sharmaroshan/MNIST-Dataset/blob/master/LICENSE` (visited on 09/11/2021).

[14] Wichtounet, *Cifar-10/license at master · wichtounet/cifar-10.* [Online]. Available: `https://github.com/wichtounet/cifar-10/blob/master/LICENSE` (visited on 09/11/2021).

[15] Brendenlake, *Omniglot/license at master · brendenlake/omniglot.* [Online]. Available: `https://github.com/brendenlake/omniglot/blob/master/LICENSE` (visited on 09/11/2021).

[16] *Alphago: The story so far.* [Online]. Available: `https://deepmind.com/research/case-studies/alphago-the-story-so-far` (visited on 09/11/2021).

[17] F. Paredes-Valles, K. Y. W. Scheper, and G. C. H. E. D. Croon, “Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 2051–2064, 2020. DOI: `10.1109/tpami.2019.2903179`.

[18] E. Yavuz, J. Turner, and T. Nowotny, “Genn: A code generation framework for accelerated brain simulations”, *Scientific Reports*, vol. 6, no. 1, 2016. DOI: `10.1038/srep18854`.

[19] C. D. Wilson, G. O. Serrano, A. A. Koulakov, and D. Rinberg, “A primacy code for odor identity”, *Nature Communications*, vol. 8, no. 1, p. 1477, Dec. 2017, ISSN: 2041-1723. DOI: `10.1038/s41467-017-01432-4`.

[20] *Omniglot about*, `https://omniglot.com/about.htm#what`. (visited on 09/11/2021).

[21] J. C. Knight, A. Komissarov, and T. Nowotny, “Pygenn: A python library for gpu-enhanced neural networks”, *Frontiers in Neuroinformatics*, vol. 15, p. 10, 2021, ISSN: 1662-5196. DOI: `10.3389/fninf.2021.659005`. [Online]. Available: `https://www.frontiersin.org/article/10.3389/fninf.2021.659005`.