Contents

1 Introduction to the Training Establishment **3**

1.1 Company Overview 3

1.2 Company History - Wave Computing 4

1.3 Company History - Paraqum Technologies 5

1.4 Wave-Paraqum partnership, separation and its effect on interns 6

1.5 Organization Structure and Hierarchy 6

1.6 Areas of Interest 7

1.7 Current Situation 8

1.8 Impacts on Sri Lankan Industry 8

1.9 SWOT Analysis 9

1.10 Suggestions to Improve the Company 10

2 Training Experience **11**

2.1 How I got the Opportunity 11

2.2 The Teal Architecture and Wave Flow Graph(WFG) 12

2.2.1 DPU Architecture 12

2.2.2 SoC Perspective 13

2.2.3 Wave design flow 15

2.3 Py2WFG: A better way to write Wave Flow Graph 16

2.3.1 Python Vs. WFG 16

2.3.2 Enter Py2WFG 17

2.4 Designing and Implementing an Efficient End-toEnd Pipeline for Machine Learning in Robotics 18

2.4.1 Need for a standard ML pipeline in DATA61 18

2.4.2 Pipeline: An Overview 19

2.4.3 Collecting and Storing Data 20

2.4.4 Storing as TF Records 21

2.4.5 Training on Supercomputers 22

2.4.6 TensorRT: Deployment on a low power device 24

2.4.7 Problems Faced and Solutions 26
2.5 Hillnet: An Experimental Attempt at Utilizing ML for Hill Climbing 27
2.5.1 Preprocessing IMU and Velocity Data 27
2.5.2 Data Collection 28
2.5.3 Merging Scaler and Image Inputs 28
2.5.4 Regression Approach 29
2.5.5 Classification Approach 30
2.5.6 Problems Faced and Solutions 30
2.6 Life at CSIRO 32
2.6.1 Reading Groups and DATA61 Meetings 32
2.6.2 Presenting the Pipeline at Reading Group to the Scientists 33
2.6.3 DATA61 Live Event 33

3 Conclusion **34**

List of Figures

1.1 Wave Computing logo . . . 3
1.2 Paraqum Technologies logo . . . 3
1.3 Wave Datacenter Server Unit . . . 4
1.4 Wave Consumer Unit . . . 4
1.5 Paraqum Technologies Staff (including Wave team members) [1] . . . 5
1.6 Wave Computing Office location . . . 6
1.7 Wave/Paraqum administration structure . . . 7
1.8 Wave computing Homepage with their target of better AI . . . 7
2.1 Wave DPU with Processing Elements(PEs) . . . 12
2.2 Wave Deep Learning Computer . . . 13
2.3 SoC Perspective of Teal Programmable Accelerator . . . 14
2.4 Data flow structure of Wave Flow Graph designs . . . 16
2.5 Typical Python code snippet . . . 16
2.6 End-to-End Pipeline we built . . . 19
2.7 Structure of a TF Record . . . 21
2.8 Data Input Pipeline with TFRecords for training . . . 22
2.9 TensorRT in a nutshell . . . 24
2.10 Deployment Pipeline: C++ . . . 25
2.11 Deployment Pipeline: Python . . . 25
2.12 Data Collection to Train Hillnet . . . 28
2.13 Merging by Broadcast and Add Elementwise as a Mask . . . 29
2.14 Hillnet Regression Architecture . . . 29
2.15 Hillnet Classification Architecture . . . 30
2.16 Presenting the Pipeline in Robotics Reading Group . . . 32
2.17 DATA61 LIVE Event . . . 33
3.1 Overview of our Time Spent . . . 34

List of Tables

Preface

Described in this report is the 24 week training experience I had in wave computing/Paraqum Technologies dating from 25.06.2018 to 07.12.2018 for the fulfillment of industrial training requirements of my B.Sc. in Electronics and Telecommunication Engineering. The report consists of 3 main sections as follows

Chapter 1: Introduction to Wave computing and Paraqum Technologies

Wave computing is a USA based AI startup and Paraqum Technologies is a Sri Lankan Electronic design startup. Wave is now ranked among the top 25 AI providers of the world [7] and Paraqum Technologies is highly respected in Sri Lanka as a pioneer of the industry and recieves design contracts from all over the world. Our training was done through a design contract offered to Paraqum by Wave computing.

Chapter 2: Training Experience

Even though I was an intern, I was given full exposure to the company workflow. I recieved a full scale project to lead and a support team from all over the world to work with. Everyone in the company were very supportive and friendly. The environment was comfortable to work and the workload was perfectly balanced, demanding yet not overwhelming.

Chapter 3: Conclusion

I can say without a doubt, this is one of the best training experiences one can get. The University of Moratuwa Industrial training division, the Department of Electronic and Telecommunication Engineering and National Apprentice and Industrial Training Authority (NAITA) together did their best to provide us with an exceptional experience that will be very useful for the future of our careers.

Acknowledgment

I take this chance to show my gratitude towards everyone that helped make this training possible and facilitated my time with it. First of all I should thank the industrial training division of University of Moratuwa and the National Apprentice and Industrial Training Authority (NAITA) for allowing us to get an industrial training of this magnitude and quality during the time of our studies and always looking out for the wellbeing of the trainees.

I also would like to thank the administration and staff of Paraqum Technologies, starting with the CEO Dr. Ajith Pasqual, Manager Eng. Hasanka Sandujith, Eng. Kasun Tharaka who coordinated the internships and all other staff members who helped in a great many ways to make the training a staggering success.

Last but definitely not least, The staff of the Wave computing team (later separated as Wave computing Sri Lanka) were always dedicated to make the training worthwhile for us. I sincerely thank the Senior Director of Software Engineering division of wave computing Eng. Henrik Esbenson, who visited and supervised us from time to time, General manager of Wave computing Eng. Nuwan Gajaweera, Technical leaders Eng. Upul Ekanayaka and Eng. Binu Amarathunga, My supervisors Eng. Achintha Ihalage, Eng. Omega Gamage and SDK team lead Eng. Dakila Serasinghe and all other staff members of wave computing for everything they did for us.

Chinthana Wimalasuriya,
Undergraduate,
Department of Electronics and Telecommunication Engineering,
University of Moratuwa.

1 Introduction to the Training Establishment

1.1 Company Overview

Wave Computing is a USA Silicon Valley based company that specializes in dataflow technology [3], an alternative to the tensorflow [2] technology used by tech giants such as Nvidia and Google to accelerate AI and neural net training. Although the company is relatively new to the game, they have had a number of notable achievements, which are explained below in detail. They outsource their design contracts to teams around the globe and one of these contracts was undertaken by Paraqum Technologies. This particular team works with the Software development kit (SDK) and the applications of the Wave dataflow platform.

Fig. 1.1. Wave Computing logo

Paraqum Technologies is one of the very first truly Sri Lankan Electronics industry companies in the country. It undertakes Electronic design contracts from big companies from around the world such as Osprey video and of course, Wave Computing. They also have their own network product line. The company had one of their teams working on a design project of Wave Computing which split up in November 2018 to form the Sri Lankan branch of Wave Computing (pvt) Ltd.

Fig. 1.2. Paraqum Technologies logo

My work was focused on the SDK layer of the Wave dataflow platform. This describes a large toolchain of software utilities that are the primary way that an advanced user would interact with the system. This team is also one of the busiest teams of the whole organization because of the sheer complexity of the SDK. In the context of the Sri Lankan staff, the SDK team is the heart of the whole project.

All details disclosed under this section is intellectual property of Wave Computing. I have included details of the training experience as much as possible while abiding with the requirements of Wave Computing on information disclosure.

1.2 Company History - Wave Computing

Wave computing is primarily a project-turned-to-startup by Dr. Derek Meyer, who is also the current CEO. The idea is to design a processor to process the huge amount of data required for complex operations such as large matrix multiplication applications in parallel. He realized his idea in to a semiconductor manufacturing company, wave semiconductor, which has become a legacy as some urls of the company still reads 'wavesemi'. As the prospect of Artificial intelligence became popular, the company understood that their time and effort is best invested in that field and hence, wave semiconductor reformed into Wave Computing AI.

To meet the demand for Artificial intelligence applications, wave computing has planned a series of devices which can fit the requirement of the users whether it is a large datacenter or a small scale business or mobile/IoT devices.

Fig. 1.3. Wave Datacenter Server Unit

Fig. 1.4. Wave Consumer Unit

MIPS Acquisition

MIPS is one of the old time giants of the silicon design game. To facilitate their ideology of integrating dataflow technology to small devices, Wave Computing recently acquired MIPS [4] for their silicon design expertise. With this step, Wave computing hopes to realize their idea of "Bringing datacenter to the edge" by installing datacenter level high performance hardware in small(edge) devices.

1.3 Company History - Paraqum Technologies

Started in 2013 as a continued final year project of four students from the 2008 batch of University of Moratuwa Department of Electronic and Telecommunication Engineering, Paraqum Technologies was first located inside the university itself. The CEO also being a lecturer in the University, this was the result of the attempt to promote technical entrepreneurs to rise up from the university level.

Paraqum Technologies quickly developed into a full sized technology startup and by 2018, the staff had grown to around 40 and every year they took around 10-12 interns under their wings to properly expose them to the electronic industry. Now they have design contracts from renowned industrial leaders and their own networking product line which is also very famous for their unique capabilities.

Fig. 1.5. Paraqum Technologies Staff (including Wave team members) [1]

1.4 Wave-Paraqum partnership, separation and its effect on interns

As described above, Paraqum Technologies had a design contract from wave computing for their toolchain and application needs. This contract had been in place for almost two years and has proved to be one of the most productive teams wave computing has ever employed. They have helped in designing the hardware part of the dataflow processing chips and by the time we started our internships, they were working on RTL level software projects.

However, Wave computing had decided it would be better to acquire the Sri Lankan team for themselves. So, in november 2018, the Wave Computing team was separated from Paraqum Technologies and established as the Sri Lankan branch of Wave Computing (pvt) ltd. with no connection to Paraqum Technologies whatsoever. Before Separation, Wave computing team worked in the Paraqum Technologies office in Kohuwala but after that they moved to a new office in Bambalapitiya.

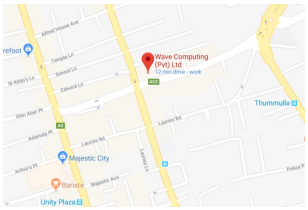

Fig. 1.6. Wave Computing Office location

As the organizations separated, interns of the wave team also moved to work in the new office premises and work was carried out as normal. But the internship contracts were not changed and technically, we were interns of Paraqum Technologies for the whole 24 weeks but worked under the supervision of Wave Computing staff. Therefore this report will be more focused on Wave Computing.

1.5 Organization Structure and Hierarchy

Wave computing team was another division in Paraqum Technologies until the separation but after that they became a fully independent entity. The dotted line in the following diagram represents the administration link that existed before the separation. Now Wave computing Sri Lanka

operates directly under the administration of their head office in Campbell, California.

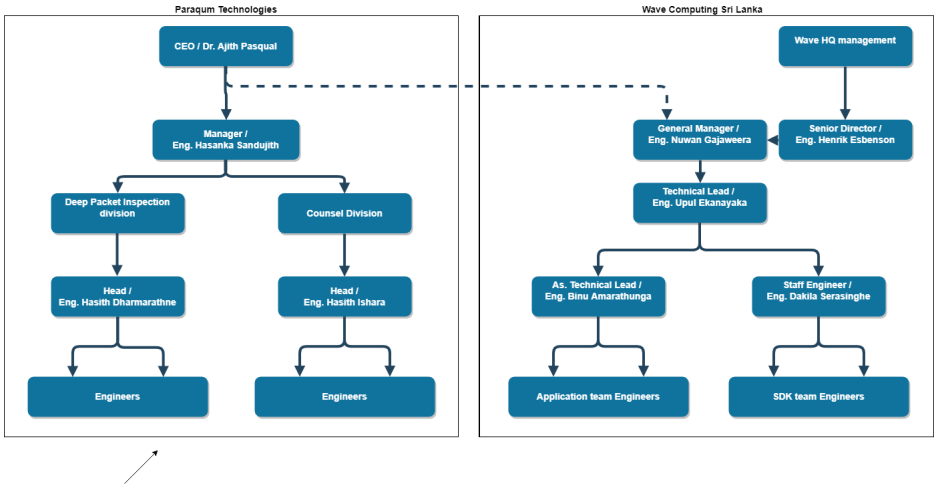

Fig. 1.7. Wave/Paraqum administration structure

1.6 Areas of Interest

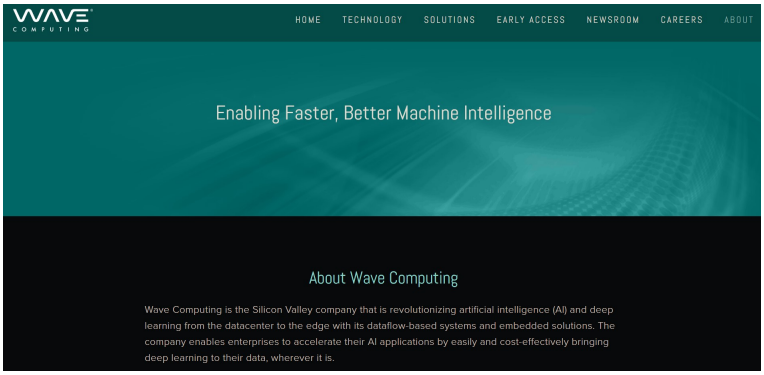

Fig. 1.8. Wave computing Homepage with their target of better AI

The main focus of Wave computing is to Introduce new processing system for heavy computational operations with their proprietary dataflow processing technology. When the system is built and released, it can be used for:

Artificial Intelligence

The main use of the dataflow processing units (DPU) is intended to be Artificial Intelligence systems. Current hardware in our devices are not suited for the massive amount of data that need

to be processed in parallel. The DPU has been optimized for this particular task to perform it at high speed. It is speculated that once wave DPU systems become available in market, it will spark a revolution in the AI industry.

Edge devices

Edge devices are small devices such as mobile phones and IoT sensors. Wave computing already has plans in motion to develop high performance, small size hardware to be installed in these edge devices. This will allow us to have very powerful yet small devices for our day to day usage.

Image processing

Image processing and AI are similar in many ways. The parallel data handling capacity of DPU systems makes it an ideal candidate for image processing applications such as self driving or biometric authentication.

1.7 Current Situation

The initial release of Wave hardware to the general public is scheduled for 2020 and the company is doing well on the roadmap to meet this target. The hardware design is nearly perfect and the software is halfway through. It can be expected that the company will meet the target soon enough and they are expanding rapidly now, with the new office in Sri Lanka hiring more and more people.

1.8 Impacts on Sri Lankan Industry

Having recently completed a very successful series E funding round [8], Wave Computing has a fairly large capital at its disposal and are ready to invest well in the Sri Lanka division. This huge amount of money will directly enter Sri Lankan economy in US dollars, aiding the stabilization of economy. They also offer employment to talented local engineers in bigger and bigger numbers every year. Finally, when the Wave product line enters market and becomes a key element in the AI game, Sri Lankans will have an important role to play in it, carrying us forward in the technical development sector.

1.9 SWOT Analysis

Strengths

- Wave already has MIPS under them, giving them a considerable head start on the silicon game. They got a team of experienced engineers with this acquisition.
- They have teams around the world with different perspectives, allowing creative solutions to problems.
- The company is very well funded, giving them plenty of runway.
- The startup culture is well maintained in the workplaces, making it appealing to employees.

Weaknesses

- Without any kind of product on the market yet, the company solely runs on VC funding
- Global teams sometimes cause delays in communications.

Opportunities

- AI field is expanding day by day, meaning the potential market for the company widens continuously.
- Another funding round can result in more and more funds due to the promising nature of the project.
- Reputation of the company is luring in more talent as it expands.
- Since the hardware is not restricted only to AI applications, there can be more potential uses to the devices.

Threats

- There may be rival companies with better solutions to the AI processing problem than the DPU technology.

- Some of the global teams are from politically and economically unstable countries and they might be lost to national crisis.
- DPU is not fully built yet. It may not give the expected outcomes and performance in real world.

1.10 Suggestions to Improve the Company

- Some sort of outbound activities can be organized to allow the team to bond even better.
- Interns could be provided with some more training sessions.

2 Training Experience

2.1 How I got the Opportunity

A long time before the industrial training selection, I had heard Paraqum Technologies was one of the best places to learn about the electronics industry in Sri Lanka and Its CEO, Dr. Ajith Pasqual had taught a module in the university that sparked an interest in me about the subject of silicon design. Therefore, when Paraqum Technologies was listed as an open CV company, I did not hesitate to submit a CV, which got selected by the company staff who then interviewed me thoroughly in their office and sometime later, informed me that I had been selected to the Wave Computing Division.

At the start of the Internship, I was placed under the supervision of Eng. Achintha Ihalage, an application engineer whose original task was to handle the timing and constraints of the DPU chip. He walked me through the basics of setting up the Wave Computing workspace, company work ethics and other needed technical skills. He also introduced us to the DPU hardware and other proprietary technologies by Wave.

During the second week, Eng. Henrik Esbenson, who is in charge of the Sri Lankan team at Wave HQ visited the office and demonstrated his ideas for projects. One of these was the Python - Wave Flow Graph translator, also known as Py2WFG. I volunteered to take that project and Eng. Henrik allocated me the necessary resources of the company including support teams and software tools to carry on the project.

This project soon became popular among the crowd of wave computing and I developed it according to the requirements and feedback. The amount of work was rather large but I managed to complete the project and hand it over by the time Internship ended. This project will most likely be adopted by full time developers and expanded to probably replace the existing design flow.

2.2 The Teal Architecture and Wave Flow Graph(WFG)

Teal is essentially the best of both programmable logic platforms and Application Specific Integrated Circuits(ASIC) combines together to give out the best performance and power efficiencies. In fact both hardware and software designs can be transferred onto the Teal fabric. Basically, Teal can be broken down into smallest unit of a Processing Element(PE). This is a simple processing unit which can carry out 8 bit operations adhering to Reduced Instruction SET (RISC) architecture.

2.2.1 DPU Architecture

A Processing Element will inherit an 8 bit accumulator who in return is connected to its neighboring Processing Elements. It is evident that a combination of a large number of such Processing Elements achieve as much parallelism as witnessed in the computational design history of the world. This revolutionary design with extensively parallel computational ability is a completely ground-breaking approach to the existing technologies used in chips for parallelism. Incidentally, a combination of 16 Processing Elements make up for a Cluster and a combination of 64 Clusters form a Super-Cluster and 16 such Super-Clusters form a single Wave Dataflow Processing Unit. The projected design for this massively complex structure has now reached the taping out process an it will only be a matter of time before the first chip gets manufactured physically. Such a developed Processing Element is projected to have a speed of close to 10GHz.

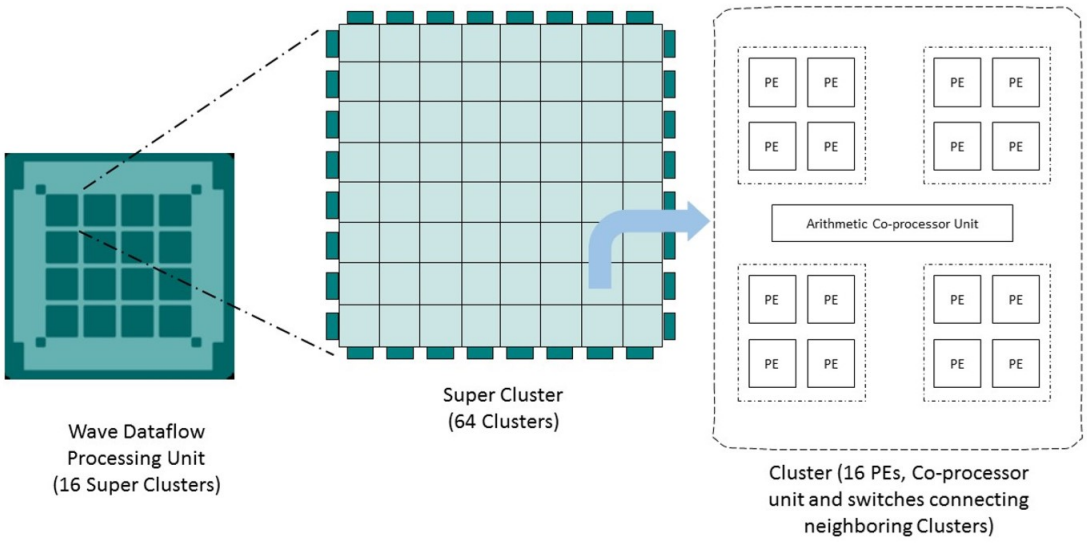

Fig. 2.1. Wave DPU with Processing Elements(PEs)

The Figure 2.1 is a comprehensive illustration of the structure of Teal design. It is noteworthy to realize that each Processing Element acts much similar to a single processing unit and therefore, each Super Cluster can be viewed in the perspective of thousands of processing units ready to function simultaneously. Nevertheless, the true power of Wave Dataflow Processing Unit design is shown in Figure 2.2 where the extent of operation of the final design is depicted. The final design is projected to have a whopping 2 Peta Operations per second amount of power which is ideally suited for the needs of computational power of the future. It is safe to say that Wave Dataflow Processing Unit will become by far the fastest ever processing unit of the world.

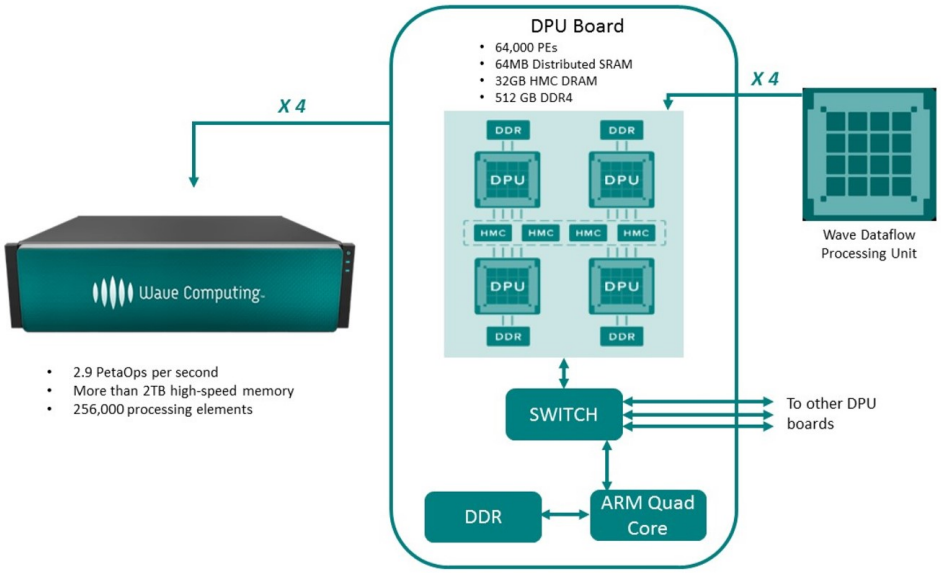

Fig. 2.2. Wave Deep Learning Computer

2.2.2 SoC Perspective

The Teal processing accelerator is essentially integrated into a System-on-a-Chip(SoC). As shown in the Figure 2.3 it is connected to processor SoC using streaming AXI interfaces. It has a direct connection toward high speed IO streams. The DMA controller (refer Figure 2.3) is the key connector between external DRAM internal Block RAMs of Teal. The accelerator itself is demand driven, meaning it will stay at sleep state unless instructed otherwise. The importance of this viewpoint was important to me as some of the design work carried out were finding a way to simulate DMA inputs and outputs. Since accessing data from external RAMs is through DMA 'Channels', some of the work related to improvements in the DMA engine were carried out during

my project executions.

The next interesting design feature of Wave Data-flow processing model are the Wave Flow Graphs. This is a representation scheme for which operations are represented as nodes and values as edges. Nodes and edges connect together in a network of operations to perform a computational task. The idea behind the clock-less operation of the tasks come through the ability for nodes to perform the intended operation whenever valid inputs are provided. A dedicated clock is not needed and thus presence of valid data at the inputs automatically will define trusted operation.

However, a sense of timing is available in the form of the concepts of tics and sub-tics. A tic is equivalent to a circular round of operations executed by a Processing Element buffer which is currently defined as 256 instructions. In fact, it is 256 times the instruction cycle time. Due to the fact that instruction cycle time is not specifically designated but rather dependent on the execution of each operations' processing speed, it is not fair by the Wave DPU design to have a specific tic-rate. Incidentally, there is a typical rate of 10 GHz for a sub-tic cycle. a sub-tic cycle is essentially the rate at which an instruction from a circular buffer(IRAM) is fetched, processed and switched.

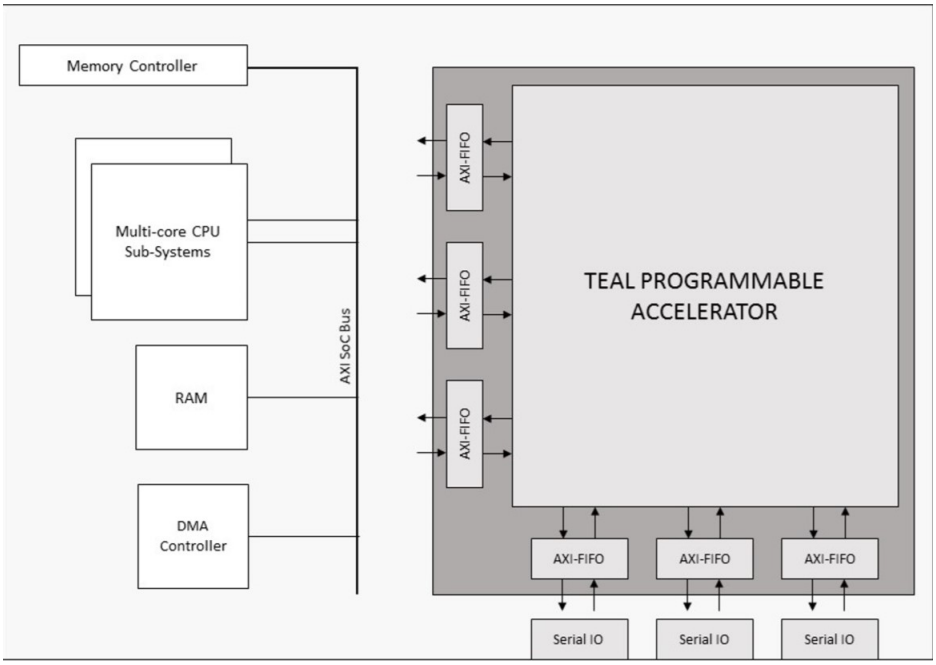

Fig. 2.3. SoC Perspective of Teal Programmable Accelerator

Apart from the contrast between the parallelized implementation of Wave Flow Graph designs and the continuous flow of operations within PEs, Wave Flow Graph is a defined rule to represent any program within the Teal fabric. In XXXXXXX section 1.3.3 (Unit Testing) XXXXXX a direct implementation of testing the Wave Flow Graph operations carried out by me, is explained in detail.

2.2.3 Wave design flow

Owing to the complexity of parallel PE operation it is evidently clear that programming instructions into the Teal Architecture is not achievable manually. Therefore, an EDA flow is prevalent so that any hardware design can be brought down through compilation, scheduling, mapping and routing to the Teal fabric level. This flow of compilation from hardware design to Byte-fabric level is done through the Wave introduced design flow. The Figure 2.4 is a representation of a very simple Wave Flow Graph design where you can see the nature of operation of the language. It basically consists of bit wise operations between inputs and outputs. Most operations are related to bit level manipulation of data and it is note worthy to realize that these operations are carried out for 8 bit data chunks. The combination of several bytes of data can be processed with special modules of Wave Flow Graph code where the data is subdivided and processed accordingly. The node edge representation of the design will give you enough ides as to the complexity that can arise with the introduction of several operations and variables into one design. The final structure will be a collection of various nets combined together from input end to output end.

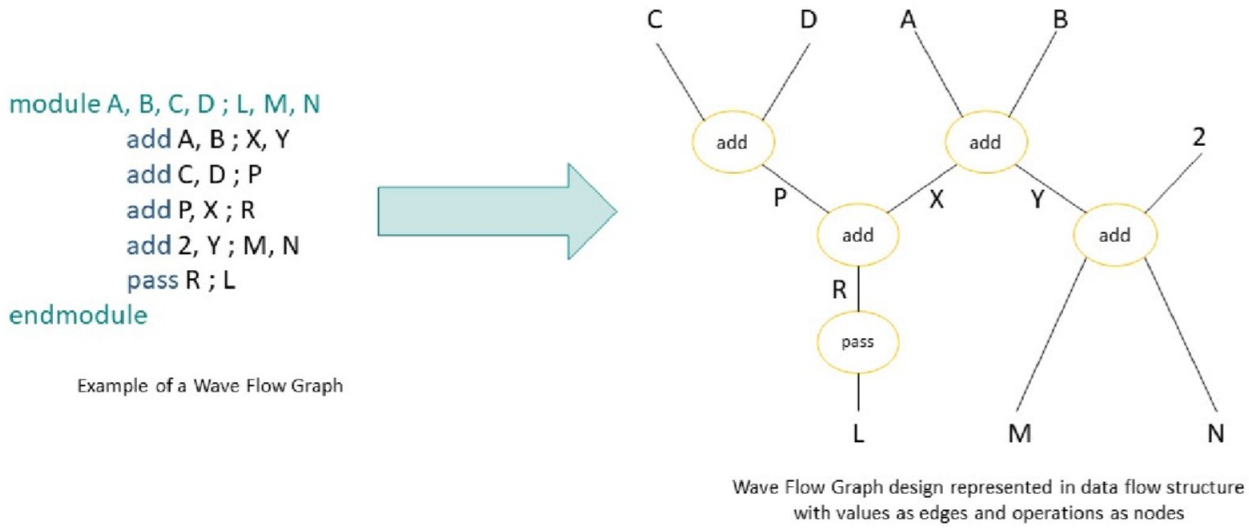

Fig. 2.4. Data flow structure of Wave Flow Graph designs

2.3 Py2WFG: A better way to write Wave Flow Graph

2.3.1 Python Vs. WFG

```python
class job():
    def __init__(self, payment, start, end):
        self.payment_ = int(payment)
        self.start_ = int(start)
        self.end_ = int(end)

def get_end_day(jobs):
    res = 1

    for j in jobs:
        if j.end_ > res:
            res = j.end_

    return res
```

Fig. 2.5. Typical Python code snippet

The Figure 2.5 shows a typical python code example and Figure 2.4 shows a WFG snippet. These two languages were built for two entirely different purposes but the highly adaptable nature of python makes a valid point whether if it can replace the functionality of WFG. But both languages have their pros and cons.

Python

Python is a general purpose scripting language which focuses on user friendliness. It supports object oriented programming and is also backed up by a huge number of highly optimized libraries

for various purposes. But when compared with languages such as C++ and Java, Python is heavy on the memory and slow for large volume computing. It is also not optimized for the particular purpose of programming the Wave DPU.

WFG

WFG was created with one purpose and one purpose only in mind, Programming the Wave DPU. Thus it has primary operators that can precisely match the deep capabilities of the DPU hardware. It can be very efficient too when properly programmed. The downside to this language is that it is very strongly typed and is a user friendliness nightmare. Repetitive operators need to be manually entered by the user and the language has no capability of any kind of looping of the language itself.

2.3.2 Enter Py2WFG

A solution was created by taking the best of both worlds

2.4 Designing and Implementing an Efficient End-toEnd Pipeline for Machine Learning in Robotics

2.4.1 Need for a standard ML pipeline in DATA61

Robotics and Autonomous Systems Group of CSIRO's DATA61 consists of world class engineers who have specialized in field robotics. Over their multiple years of experience in research and engineering, each of them have formulated an efficient workflow for themselves. Workflow is the general procedure where they collect data, process them in specialized software or write programs for custom processing, build software that can run real time on an existing robot to process the environment and make decisions, debugging their products or code and so on.

Today, with the rise of machine learning in several fields, it is a valuable tool for a field roboticist since it is fundamentally suitable to address the common problem in field robotics: processing complicated real world through noisy sensors and making high level decisions. Many of these leading roboticists have realized that and have already started using machine learning in their projects. However, in DATA61, so far, there is no unified framework that acts as a common ground to the different researchers working in different projects. Also, many of them currently use their local computers (powered by GPUs) for machine learning tasks out of habit and convenience, rather than utilizing the supercomputing resources available at CSIRO. And some of them who have figured out the procedure of using a supercomputer follow certain practices which are fine in local computers, but inefficent when done on supercomputers.

When Nicolas Hudson (our supervisor) took position in DATA61, he set out with the task of streamlining the machine learning (and other) procedures in DATA61 into unified frameworks, such that different researchers can work together, share their knowledge on a common ground and follow practices which are efficient and that maximize the utilization of the resources at disposal. One of the main motives behind such a task of streamlining is the DARPA Subterranean Competition (see: section **??**) on which different groups of DATA61 will be working together for the next few years.

2.4.2 Pipeline: An Overview

Therefore, we were requested to create an end to end pipeline for machine learning. This pipeline should be compliant with the current general workflow of most roboticists in RAG and should be efficient and scalable. We were asked to use the latest versions of software and the most general hardware controllers so that our pipeline can be effectively generalized. It is end-to-end in the sense that includes best practices in collecting and storing data, building models and training them with large volumes of data and finally deploying the trained models on power-constrained high level controllers which are to be mounted on robots.

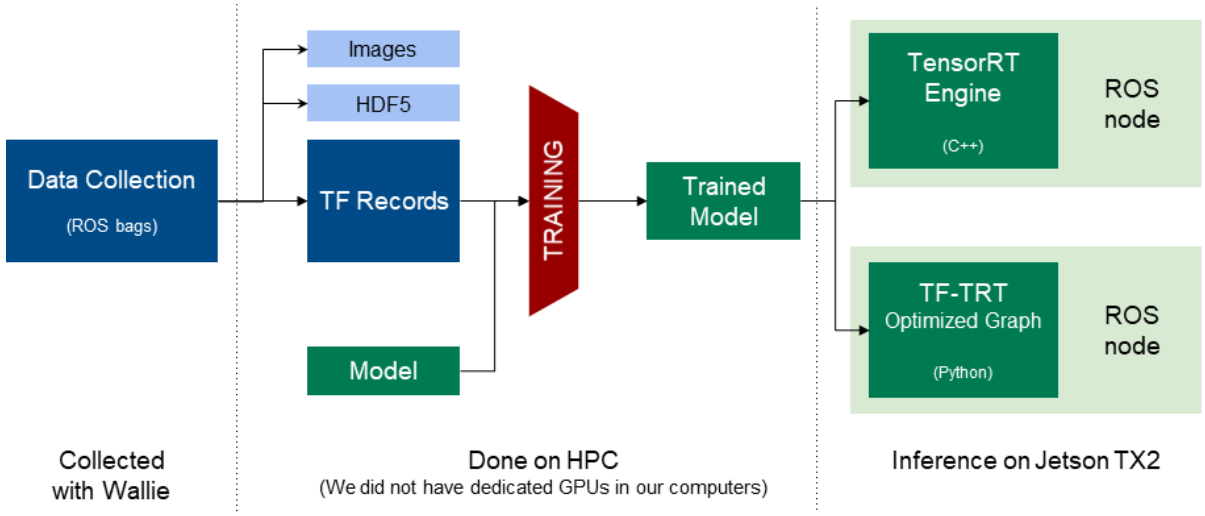

Fig. 2.6. End-to-End Pipeline we built

We were asked to do the two projects: Trailnet and Hillnet using this pipeline while perfecting the pipeline itself as a proof of concept. We were also asked to document our processes extensively in the internal wiki pages of CSIRO along with reusable code. In addition we were requested to create simple code snippets that demonstrate the key points of each part of the pipeline and finally explain and demonstrate our pipeline to the fellow scientists at a Robotics Reading Group meeting.

2.4.3 Collecting and Storing Data

In field robotics research, ROS (Robot Operating System) is used as the common platform or middleware to connect sensors and actuators of a robot. ROS packages are available for most of the professional high-end sensors and data from such sensors can be recorded in 'ROSbags'. ROSbag (.bag) is a file where all the activities of a ROS session was recorded in time-ordered format. They can be generated by a built in ros package. The user is able to record any session into a rosbag and play them later. ROS and ROSbags are used as the de-facto standard by the roboticists at CSIRO to collect data for analysis.

These ROSbags are unnecessarily huge in size and time-ordered. This poses difficulty in directly using them as a data storage format for machine learning. Due to their size, it is not possible to fit them into memory and due to their structure, it is not possible to perform multiple parallel reads from the same file in disk to accelerate the training process. Being time-ordered is an advantage for training RCNNs, but for training simple CNNs, we need the data to be completely shuffled for faster convergence when training.

Storing as Images

Therefore, the first solution is to generate a dataset of thousands of individual images (JPG, PNG) sorted into folders representing various classes. Such a dataset of images can be thoroughly shuffled easily, by simply shuffling a string-list of their filenames. Loading all the images into the memory at once is impossible even for 40,000 images. Instead, one can write scripts to read and queue images from disk, but it is cumbersome and error-prone.

In addition to that, storing thousands of images in the HPC (High Performance Computing or Supercomputer) is highly discouraged. When it is done, the performance gains of processing the image faster is offset by the overhead delay of reading and processing the image headers of each JPG or PNG image and results in a slowdown. Also, storing a large 'number of files' in network drives puts a strain on the existing file management systems. I had my access revoked multiple times for accidentally storing a large number of files in my personal folders of the supercomputer.

Storing as HDF files

Due to these issues, I spent a few days experimenting with serialized file formats such as HDF5. They are ideal in the way that they are read from disk directly (not from memory) and queuing operations can be performed with little difficulty to boost the reading speed. However, a major problem with HDF files is the inability to render themselves to a shuffle operation. Since the data is serialized, to shuffle an existing HDF5 dataset, the entire dataset has to be read into the memory, which is impossible. Then the only solution is to shuffle before writing a dataset. But given that ROSbags can be read only in a time-ordered manner, this is also not possible.

2.4.4 Storing as TF Records

Then I turned to TF records [6], a dataset format recommended by the TensorFlow framework, which is the standard framework for ML in CSIRO RAG. I found TF records to be ideal for our purpose due to several reasons. Firstly, TFRecord is a serialized file format, that supports reading from disk. Also, the data stored in TFRecords can be bundled, like bundling attributes into an object in Object Oriented Programming. That is, the input image and the corresponding label, IMU data, velocity can all be bundled into a single TFexample, stored and retrieved for training. In addition to that, tensorflow offers multiple operations that can be performed on tfrecords to shuffle them, queue them and read them effectively without reading the entire file. In a nutshell, tfrecords were designed to be used in supercomputers and to solve the problems with other methods.

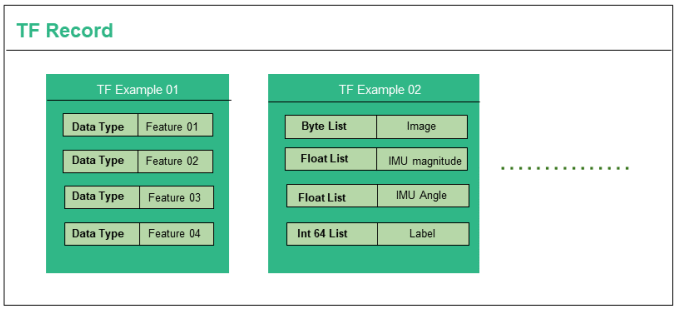

Fig. 2.7. Structure of a TF Record

A TFRecord dataset can be composed of thousands of tfexamples (tfexample is conceptually similar to an object in OOP, in reality is a protobuf defined in tensorflow). Each TFexample is a bundle of multiple features (a feature is conceptually similar to an attribute in OOP) that describe the TFexample. A feature can be of only one of three datatypes as allowed by tensorflow: ByteList, FloatList and Int64List. For efficient shuffling, one needs to split the dataset into multiple tfrecords. That is, as the raw file (such as ROSbag) is traversed in time-order, first 500 tfexamples should be written into one record in time-ordered manner, second 500 into next and so on. This is also because, having millions of examples inside one tfrecord slows down the read operation and the recommended optimum size of the tfrecord is around 100 MB.

2.4.5 Training on Supercomputers

The set of tfrecords written that way can be read in the following manner. First, the list of tfrecord files in a dataset is shuffled. Then we start reading from all the tfrecords in parallel. Here, we can split the dataset into training and validation randomly. This read operation can be queued easily (prefetching). This pipeline can then be effectively shuffled (by shuffling files inside a prefetch buffer from each file and then interleaving them across the read heads from multiple files), repeated and decoded using a function we specify (to convert the tf datatypes to the datatypes we use, perform operations such as BGR to RGB conversion in images, changing dimensionality...etc), then batched and released as the input for training a model.

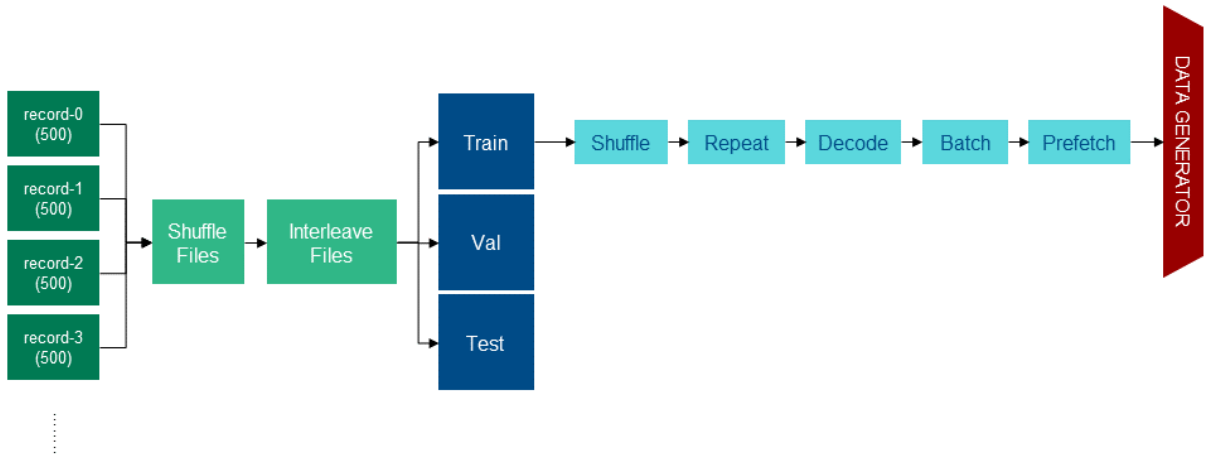

Fig. 2.8. Data Input Pipeline with TFRecords for training

A key insight here is that each of the above mentioned operations: shuffle, interleave, decode, batch, prefetch, train-val split are all implemented as tensorflow ops (operations). That is, tensorflow follows a dataflow programming paradigm (like verilog HDL). We specify the operations that should be performed on data and the connections between those operations (a tensorflow graph) through our python code, but these operations are not performed as control flows through those respective lines in the code. Instead an abstract graph is formed as control flows through those statement. Finally, when we start a tensorflow session and run it (sess.run()), the graph gets executed and data starts flowing through it in parallel (much like verilog HDL). The above mentioned operations are also performed this way. We specify the transformations that should be done on the dataset in an abstract manner. Tensorflow forms a graph and performs those transformations in an efficient parallalized way that maximizes the use of resources at disposal.

TFRecords also yield well into the training of models in tensorflow and tensorflow-keras. Both APIs support generators, a type of python functions that mimic an iterator (like a list) to provide their input, output data pair for training. Such a generator can be created to read a set of tfrecords using the above ops and yield one pair at a time as output. This blends seamlessly into the training procedure.

Unfortunately tfrecords are not very intuitive to use. Not many roboticists are not familiar with the unique datatypes and the dataflow programming paradigm of tensorflow. Advocating the use of tfrecords as the common dataset storage format and showing others how they can be included in their pre-established custom workflows was a key part in the development of the pipeline. I helped many of my coworkers by creating tools to visualize, write and read tfrecords. I also documented these procedures extensively for the future reference in CSIRO's wiki pages.

2.4.6 TensorRT: Deployment on a low power device

Machine learning frameworks such as TensorFlow and Caffe are built to be optimized for training a neural network model. They use high precision datatypes and typically utilize a large amount of memory, GPU and processing resources even to perform inference using a trained model. In addition, neural networks are typically built and trained by data scientists without a software engineering background who prefer high level languages like python and generally are not focused on writing fast, efficient code to be run on resource-limited devices. This poses a key problem in machine learning: these models are generally not portable to devices that have limited memory, processing power and use little power.

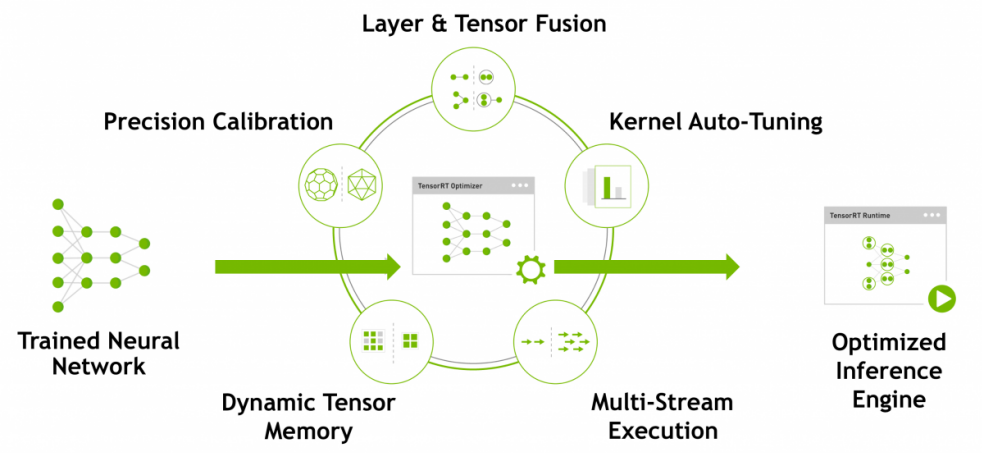

Fig. 2.9. TensorRT in a nutshell

To address this problem, NVIDIA released TensorRT as their own library. TensorRT is NVIDIA's programmable inference accelerator, that can optimize any supported model for inference on devices with limited resources. When a model is optimized, first the full-precision float32 datatype is converted to half-precision float16. Then, tensorRT traverses the tensorflow graph and finds subgraphs that can be optimized. That is, a convolution operation followed by activation and max-pooling operations is combined into a single tensorRT operation that can be executed in very few clock cycles. These tensorRT operations are then assigned to various parts of the target device: the CPU, GPU and special DLA (Deep Learning Accelerator) chips and queued for parallel operation.

Therefore, by converting a tensorflow graph (optimized for training) into tensorRT graph (optimized for inference), one can obtain a model that can have very low latency during inference, used very little memory without a significant loss of accuracy. This model can then be deployed for inference on NVIDIA's embedded hardware such as Jetson TX2.

TensorRT is released both as a C++ API and a Python API. However, Jetson TX2 only supports the C++ API of TensorRT. The following figure shows the procedure I followed to convert a tensorflow graph into a tensorRT graph for inference. After conversion, I created C++ ROS nodes for Trailnet and Hillnet to accept messages from sensors, process them with the model and output velocity commands for the motor controller. The C++ API is extremely fast: it could process a 320 x 180 x 3 image through a resnet with 20 convolution layers within 5 ms, allowing a processing rate upto 200 frames per second.

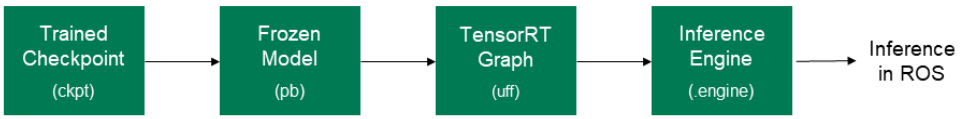

Fig. 2.10. Deployment Pipeline: C++

However, not all models can be converted into tensorRT graphs for inference. That is, only a small subset of tensorflow's operations are supported by tensorRT for inference. For example, batch normalization, a common operation used in many networks, was not supported in tensorRT 3.0. Therefore, in order for our pipeline to be generalized, I also looked into a way to convert optimize any tensorflow graph for inference. Google and NVIDIA joined hands to solve this issue and have provided tensorflow-tensorRT (TF-TRT) as a part of tensorflow framework since 2017. TF-TRT can analyse any tensorflow graph (any neural network) and choose the subgraphs (operations) that can be accelerated by tensorRT. The unsupported operations are then perform through tensorflow in an unoptimized manner. Result is a model that is slower than pure tensorRT and faster than pure tensorflow during inference. Following is the workflow for this.

Fig. 2.11. Deployment Pipeline: Python

I created a python ROS node also to accept messages from sensors, process them with the model and output velocity commands for the motor controller. The same network had a latency of 20 ms in the python node. That is about 50 frames per second. The detailed procedure and my scripts used for these conversions were documented thoroughly in CSIRO's wiki pages.

2.4.7 Problems Faced and Solutions

When I began building the pipeline, the major problem I faced was that none of my coworkers were familiar with tfrecords. I had questions about the best practices in shuffling and the order of operations, which adequate answers were not provided in the documentation. Therefore, I started to experiment with a simple dataset of integers from 1 to 50,000. I brainstormed my ideas and tried different configurations of writing, reading and shuffling and figured out the best way of doing it efficiently.

Then, there was a problem when we found ROS was not installed on the supercomputer as a package and we are not allowed to install new packages there. This meant that we cannot play the ROSbags to extract them in the supercomputer. We tried to extract the ROSbags into tfrecords and then pushing the tfrecords into the supercomputer, but it was a cumbersome task and the storage on our local machines were not sufficient for that. Our coworker Micheal gave me a docker container image he has built for a similar task. Docker container is a virtual environment which one could build with necessary packages and run on a computer as an abstract layer. I spent several days building and rebuilding a docker based on that, to include rosbag, opencv and cvbridge python packages and used it to convert ROSbags to tfrecords in the supercomputer, without running ROS.

When working with the C++ interface of tensorRT, I worked overnight at office to build the first draft of the ROS node: the pipeline to process the image and perform inference and faced with a TensorRT error: "dimensions don't match for the inputs of add layer". I spent a week debugging this issue: raised an issue in NVIDIA's developer forum, rechecking dimensions of my network, building a residual block with pure tensorflow ops and the problem persisted. I decided it was a bug. After a week they confirmed it indeed was a bug and asked me to update our software to the latest version released few weeks prior to that. We did, and hence solved that issue.

2.5 Hillnet: An Experimental Attempt at Utilizing ML for Hill Climbing

After successfully demonstrating indoor-Trailnet built from scratch, trained and deployed with my pipeline, our supervisor Nick described about his idea of experimenting with a algorithm to climb hills while avoiding obstacles using computer vision. We were asked to come up with a system where two different types of inputs are merged: scaler inputs representing the direction of the slope and the RGB image input from a camera to output a velocity command to control the motor controller. Trailnet, which itself was based on resnet-18 was chosen to be modified to build such a neural network.

2.5.1 Preprocessing IMU and Velocity Data

The LORD Microstrain IMU sends its data through serial to its ROS package, which publishes the IMU readings in two types: as a quaternion and set of cartesian x,y,z vector components of perceived acceleration. I decided to use the vector components to calculate the magnitude and direction of the gravity vector projected on the horizontal plane of the robot. Direction $\theta \in [-\pi, \pi]$ was measured with respect to the heading direction and then converted to $[0,1]$ range using the sigmoid function to match the range of the other normalized inputs. I chose to output the angular velocity as a float $\in [0,1]$ using a sigmoid output node (in classification approach) and scale it to the necessary angular Velocity.

$x, y, z =$ Magnitude of the cartesian components of perceived acceleration. x: forward, z: vertical

$r, \theta =$ polar components of the acceleration vector projected on the horizontal plane of robot

$r_{in}, theta_{in} =$ processed values given as input to the network

$$r = \sqrt{x^2 + y^2}$$

$$r_{in} = \frac{r}{g} \qquad \text{normalized by the maximum: g}$$

$$\theta = \arctan(\tfrac{y}{x}) \qquad \theta \in [-\pi, \pi]$$

$$\theta_{in} = \text{sigmoid}(\theta) \qquad \theta_{in} \in [0,1]$$

2.5.2 Data Collection

As per the instructions of our supervisor, Uvindu collected data in the slopes around CSIRO campus during the last few weeks of the internship. He placed the robot on the bottom of the hill and drove the robot following the steepest ascent. When he faced an obstacle, he went around the obstacle and continued to follow the steepest ascent. Image streams from the camera, IMU data and the odometry reading from the motor encoders were recorded in ROSbags.

Fig. 2.12. Data Collection to Train Hillnet

2.5.3 Merging Scaler and Image Inputs

For this task, it was necessary to decide how the two types of inputs: scaler and image are combined in the neural network. Our supervisor proposed a method, where the IMU values are concatenated to the flattened output of the convolution layers, followed by few fully connected layers. The inspiration for this idea comes from the insight that deeper layers of a deep neural network identify higher level features and therefore the flattened average pooling output of the CNN should be containing information about by how much should the robot turn to avoid the obstacle. Therefore, concatenating the IMU inputs, which also tell by how much the robot should turn to follow the hill and following it with few fully connected layers might result in the network learning an OR operation between two inputs.

However, a research paper on merging these kind of inputs [9] proposed an alternative method, where the scaler inputs are broadcasted into a matrix with the right size, processed by few fully connected layers and added elementwise to the output of an intermediate layer in the CNN. The insight behind this is the fact that the output of the fully connected layers might act like a mask

on the image, effectively clouding and directing the decision process of the CNN. After some consideration and experimentation, we decided to follow our supervisor's method since it was more suitable for our task.

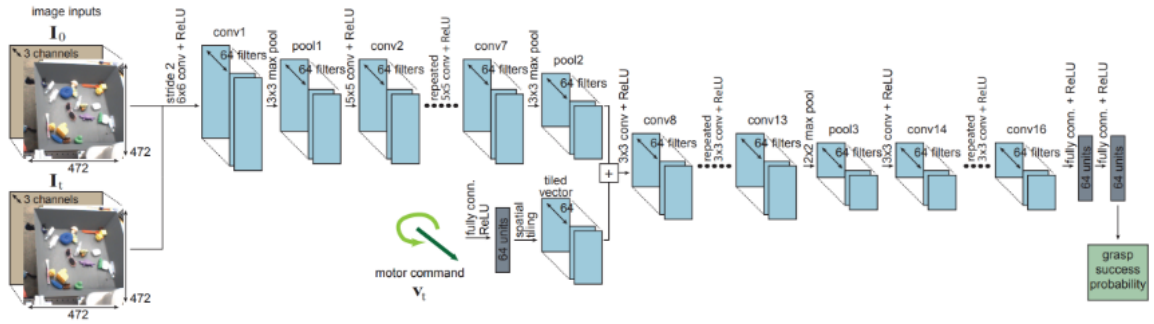

Fig. 2.13. Merging by Broadcast and Add Elementwise as a Mask

2.5.4 Regression Approach

Next we brainstormed on how to post-process the output of the network to steer the robot. During the discussion, our supervisor first suggested to use the regression approach. That is, to build a network with a single output node that has no activation function, so the output value can be directly used as the angular velocity command to steer the robot. This is quite straightforward to build, train and test. The neural network can be trained with IMU and Image data and output as the angular velocity from odometry reading when driven by remote control during data collection.

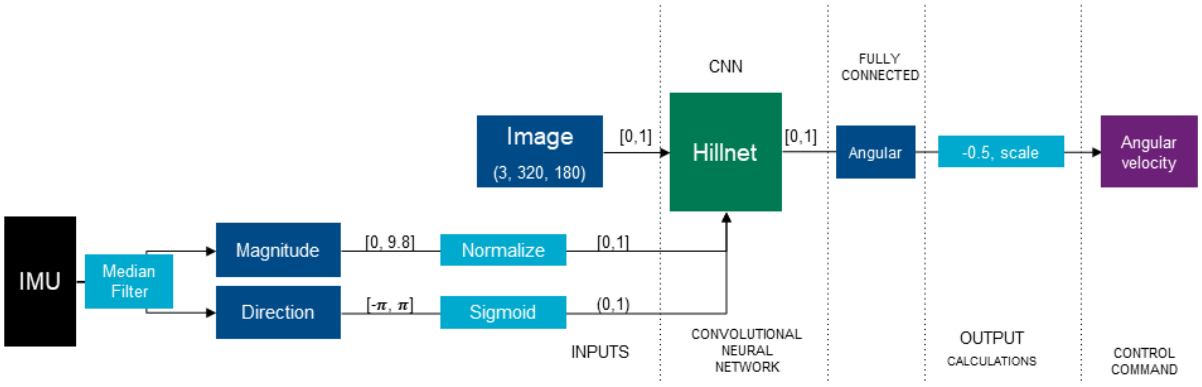

Fig. 2.14. Hillnet Regression Architecture

2.5.5 Classification Approach

During that discussion, I suggested trying a classification approach similar to that of trailnet. First advantage was the ability to fine tune the output by adjusting the constants. With regression, either it works or not. But with classification, we could fine tune it to work as we wish. Then, the effect of human error introduced in collecting data can be made insignificant by quantizing the angular velocity into classes: left, center and right.

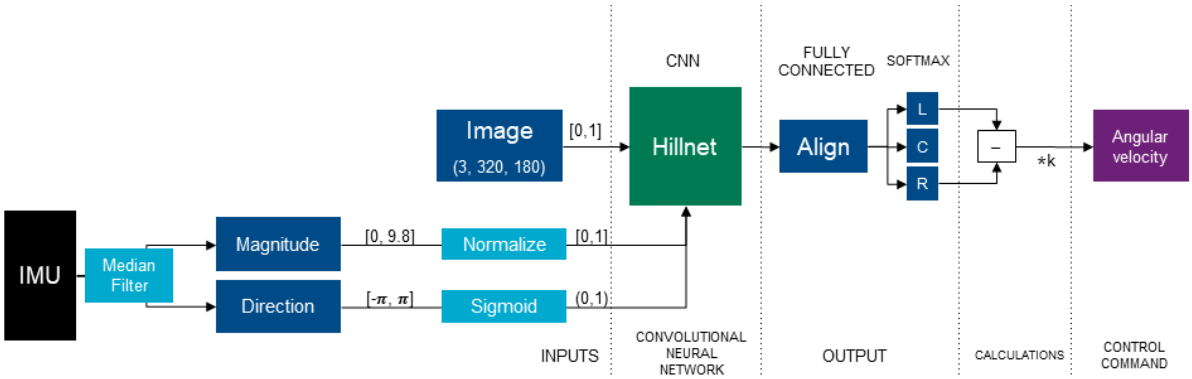

Fig. 2.15. Hillnet Classification Architecture

Next, I proposed a scheme of augmenting data to effectively triple the amount of collected training data. That is, first we record image streams from all three cameras. Then, in the input pipeline, I made changes to add +30°and -30°to the IMU angle and use it as datapoints along with the image streams from left and right cameras respectively. This would train the network to NOT turn away from the steepest slope, unless an obstacle is encountered, I argued. Our supervisor accepted the idea and asked me to work on both classification and regression in parallel to compare their merits.

2.5.6 Problems Faced and Solutions

First problem we faced was the human errors in data collection. Since the hills had a small slope, Uvindu had difficulty in visually identifying the highest-slope direction and steering the robot towards it. When we analysed the collected data using the visualization techniques, we found there was a steady state error by up to 5°-10°. We tried collecting more data with a couple of days remaining to end the internship.

Another anomaly I observed from the visualization was the fact the IMU input had a high variance (noise). This was due to facts the slope was gentle (high percentage error) and the spring loaded suspension system of Wallie was causing it to wobble, introducing low frequency vibrations. This was critical, since our neural network does not remember nor correlate with the past inputs, but considers only the current inputs to give instantaneous outputs. Such a noisy input will prevent the training process from converging to a global maxima. Hence I suggested mounting the IMU at an angle to reduce the percentage error. After experimenting with mean and median filters of various lengths, I implemented a median filter of length 50 to smoothen the noise.

On the last few days of the internship, while debugging the network, I accidentally noticed that our collected dataset had an unhealthy disparity. That is, only 0.15% data accounted for avoiding obstacles, while the rest 99.85% accounted for climbing following the steepest hill. This is a classic problem in data science where the neural network simply learns to suggest "go forward" and be right 99.85% of the time! I had discussed a similar potential problem with the supervisor at the beginning of the project, where I raised concerns that "we are not showing the robot which input-output combinations are wrong. we are showing only what is right". Our supervisor assured that "the robot will learn what is wrong, when you turn the robot to face the hill after avoiding an obstacle". However, the percentage of that kind of data was dwarfed by the "straight climbing" data in the dataset. I discussed this with Micheal and our supervisor and started implementing my idea of artificially boosting the frequency of "avoiding obstacle" data in the input pipeline.

However, we were running out of time by the end of the internship to try all possible ideas and experiment with all the possibilities. I stayed for multiple days overnight at office to try and finish as much as possible, but we couldn't try everything within that short time. Spending most of our time on building and debugging the robot platform could be one of the reasons for our time being limited for exploring new ideas with Hillnet. However, we realized that this is quite common in experimental field robotics, where scientists get to spend most of the time struggling with the hardware issues.

2.6 Life at CSIRO

CSIRO, being a world class research institute, thrives to create a stress-free work environment that encourages people to socialize and to boost creativity. There is a workplace culture in DATA61 to bring cakes (or any equivalent sweets) for all the coworkers if one gets married, arrive at CSIRO, leave CSIRO, has a birthday and so on. I shared Sri Lankan sweets (sent by my mother in mail) for my birthday and cakes with everyone for arriving at and leaving CSIRO.

2.6.1 Reading Groups and DATA61 Meetings

Every friday, a small meeting called "Robotics Reading Group" is hosted, where one scientist explains his current project to everyone who attends the meeting. This way, we get to know the latest technology that is being developed in different parts of DATA61 and new projects that are being started. This meeting also allows the scientist to be questioned, so that he can derive insights from the audience and refine his procedures in the future. Also, once in a fortnight our supervisor Nick holds a "ML Robotics" meeting for all engineers working on machine learning. There we discuss our current ideas and issues to help each other. In addition to these, there are monthly meetings with the entire CSIRO (branches from all over Australia join via video conferencing), where new developments are discussed. One such meeting had the lead scientist from NASA's Insight Mars Lander mission as the chief guest explaining the challenges faced in their mission and taking our questions on the matter.

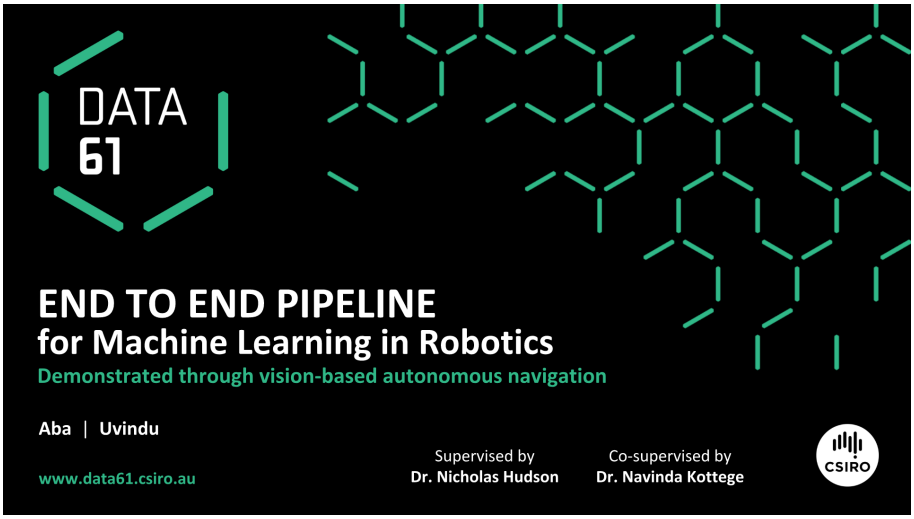

Fig. 2.16. Presenting the Pipeline in Robotics Reading Group

2.6.2 Presenting the Pipeline at Reading Group to the Scientists

Uvindu and I presented [5] the pipeline to other scientists during one of the last Reading Group meetings of the year. It was well received, thanks to the support of our supervisor who encouraged others to use our pipeline in their workflow. Many asked questions and were convinced of the merits of such a unified framework. I had a few scientists reaching out to me on the following days asking me to develop visualization tools to complement the pipeline.

2.6.3 DATA61 Live Event

DATA61 Live is an event held annually to showcase the science and technology innovations of DATA61 from all over Australia. In 2018, it was held in Brisbane, in our city. The theme was: 'Adapting to Disruption'. We signed up as volunteers and apart from volunteering, we had a chance to attend many lectures, talks and forums. It was a great experience.

Fig. 2.17. DATA61 LIVE Event

3 Conclusion

I worked as a research intern at DATA61 for 24 weeks. During this time, I worked three intertwined projects under the supervision of Nicolas Hudson and Dr. Navinda Kottege.

The first project was developing indoor-Trailnet: a classification based approach to the autonomous navigation problem. For this, first I built a robot with Uvindu Perera for data collection and testing. We designed the power distribution system of the robot and assembled a high level and low level controllers. We calculated the torque requirements for the motors, current, voltage and power limitations for the power supply components and motor controllers during this process. We learnt a lot through debugging the errors we came across and extensively troubleshooting whenever a component or circuit board is damaged to provide a report on the event. I also debugged the driver software for the motor controller and modified parts of it to fix certain issues since it was not being maintained anymore.

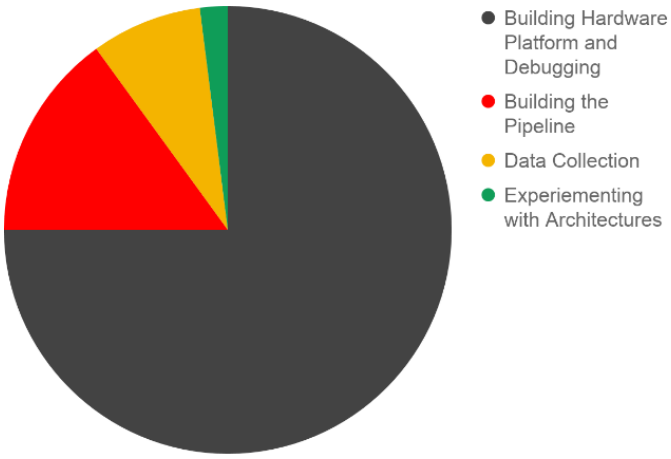

Fig. 3.1. Overview of our Time Spent

I learnt tensorflow and tensorflow-keras for this project and built a 20 layer residual network from scratch and trained it on the supercomputer cluster. Then I learnt tensorRT and deployed the trained network on Jetson TX2: a powerful embedded device that acted as the high level controller for the robot. I also learnt ROS (Robot Operating System) extensively, and created multiple ROS nodes and packages for testing, collecting data from sensors and for controlling the robot.

For the next project of building an end to end pipeline for machine learning in robotics, I experimented and figured out the best practices when training models using large datasets on the supercomputer. I documented these experiments and the results, and proposed an efficient method to set up a writing and reading pipeline. I presented [5] my pipeline in the Robotics Reading Group meeting. Many scientists were interested and some followed up via email asking questions and requesting to build tools for visualizing datasets.

The final project was experimental, where I explored different configurations of neural network architectures to build a robot that can climb hills while avoiding obstacles. Through literature survey and in-depth discussions with my supervisor, I learnt a lot about the structure of Deep Convolutional Neural Networks and possible methods of combining scaler data with the images to train a network. I tried several approaches here: solving the problem as a classification and a regression and using different ways to combine the inputs into the network. However, I was unable to get good results within the time we had there, due to the problems in data collection and the lack of time as we had to spend most of the time building and debugging the robot platform.

Also, I learnt about cross modal learning transfer and neural network distillation process through literature survey as a part of the project that was initially given, before we were changed into a different project a week later. Before and during the internship program, I expressed my interest in working in a project where I can develop algorithms and tackle abstract problems that could lead to new research and a publication. Unfortunately such a project was not available for interns at the time and my supervisors were satisfied with the work I was doing with hardware and deployment of machine learning. However, I learnt a lot through these projects and it had been a great experience. I also got familiarized with the software tools widely used in academia, high end sensors and controllers, and I daily worked with the Bracewell cluster, which is one of the world's largest supercomputer clusters.

In addition to that, I learnt the etiquettes and responsibilities of working as an employee in a company. Helping others and asking for help, attending meetings and following up via official emails, documenting all the tasks and weekly progress in the wiki pages of CSIRO helped me learn a lot about these responsibilities. The work culture in DATA61 is exceptionally inclusive,

where we got to work with people of multiple nationalities and share our culture. The students are allowed to work on their own pace and I was allowed to work overnight on multiple days and work on weekends as well. I also got to attend events such as DATA61 LIVE, where I could attend to many talks and discussion forums and observe the development of cutting edge technology of Australia through the exhibits.

From my experience, I would suggest DATA61 to assess the skills of the interns and assign them to projects relevant to those skills, to utilize their full potential for a project. Also, it would help if they can give an overview of the project at the beginning of the internship and set incremental goals to be completed at given deadlines. I found it disorienting when the project given to me before the internship was changed as I reached there and changed again a week later to settle on an experimental project of my supervisor that subsequently evolved into the three above projects (that I explained in Chapter 2) through the period of six months.

I would also like to suggest NAITA to computerize the supervision process, where interns can submit the intern diary and monthly reports online. This would help because in organizations like CSIRO, the students are expected to maintain an online diary and the interns can save time by writing by hand the same thing they have typed into the online diary.

Therefore, I can conclude that my overall experience in DATA61 was great. I had the opportunity to learn a lot and make contacts. I am deeply thankful to the Industrial Training Division of our university and NAITA for this internship and I am thankful for DATA61 and my supervisors for providing me with such an exceptional opportunity and a training experience.

References

[1] About Paraqum Technologies. https://paraqum.com/about.

[2] About TensorFlow. https://www.tensorflow.org/?hl=hi.

[3] About Wave Computing. https://wavecomp.ai/company.

[4] Mips bought by wave computing. https://www.electronicsweekly.com/news/business/mips-bought-wave-computing-2018-06/.

[5] Presentation: ML Pipeline - DATA61 RRG. https://docs.google.com/presentation/d/1Z7OW8ILDy-wdoWo1Hw84-F0VBjzLAXzpfrpAYbkIddE.

[6] Using TFRecords and tf.Example. https://www.tensorflow.org/tutorials/load_data/tf-records.

[7] Wave Computing named a top 25 AI solution provider. https://globenewswire.com/news-release/2017/07/20/1054678/0/en/Wave-Computing-Named-a-Top-25-AI-Solution-Provider-for-2017.html.

[8] Wave computing raises 86m usd in oversubscribed series e round. https://wavecomp.ai/wave-computing-raises-86m-in-oversubscribed-series-e-round.

[9] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *arXiv:1603.02199 [cs]*, Mar. 2016. arXiv: 1603.02199.