# Technology Review and Implementation Plan

# for

# Deep Learning on Embedded Platform

**Version 1.0**

**Prepared by Christopher Johnson, Luay Alshawi, Gabe Morey**

**CS 461 Fall Term**

# June 5, 2017

# 1 Abstract

Working on a large scale project requires multiple pieces to come together in the right way. If the right pieces aren't used, the project will turn out poorly for it. In this document we lay out several different components necessary to build our project, which consists of building a neural network setup which can learn to play the game Galaga. We will discuss different technologies that can be used to satisfy the needs of those components and come to a conclusion on which technologies to use. This document also lays out which team members are responsible for different pieces of the final project.

# 2 Duties

## 2.1 Technology 1-3

### 2.1.1 Luay Alshawi

I will be responsible for setting up the operating system development environment. I will make sure that all the requirements development dependencies such as visual libraries, programming languages, and neural network work on the Linux operating system. Also, I will be responsible for making sure the visual library is working and can capture the game Galaga then feed it to the neural network. Although we as developers we should not be struggling with the programming language but I will be responsible for making sure we are using the right programming language and works on Linux.

## 2.2 Technology 4-6

### 2.2.1 Gabe Morey

I will be in charge of setting up the neural network for learning on the embedded processor. It is my job to make sure the network can be properly trained and utilized to play the game Galaga. With this in mind, i will be reviewing the different deep learning frameworks around the weband deciding on one to use for the project. I will also be in charge of the actual neural network we train and will be covering several established networks. I will also be reviewing several clouds that allow for GPU use without physically having a strong enough GPU for neural network training. So in total I am responsible for making sure we have a proper neural network structure which has ample cloud based processing power to perform the task we have set out for it.

## 2.3 Technology 7-9

### 2.3.1 Christopher Johnson

I am responsible to making sure that our team is working efficiently with the Jetson TX1 platform. Part of this requires that we have good communication with our client. I will be researching various forms of group communication tools that could help make our meetings more efficient. I am also in charge of making sure we have a good way to capture the visual input for our program and that we are getting efficient data visualization when the program is running.

# 3 Technologies

## 3.1 Visual Library

### 3.1.1 Open CV

An OpenCV (Open Source Computer Vision) is a library that can be used with tons of different languages(C, C++, Java, Python, etc.). It provides standard things such as image capture, Faces recognition, Gesture recognition, Motion tracking, Mobile robotics, Object identification and Image manipulation. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code (Team).

### 3.1.2 SimpleCV

SimpleCV is a framework including several libraries and uses Python for scripting. Due to the nature of python, it can either run scripts or use an interactive shell to do computer vision computation and related tasks. With it, you get access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap storage (Sight Machine, Inc).

### 3.1.3 Community Core Vision

CCV (Community Core Vision) library is an open source/cross-platform solution for blob tracking with computer vision. Supports a wide range of camera devices with the ability to stitch and switch in between sensors on the fly. It is built on C++ for real-time performance and stability using common frameworks including OpenCV and open Frameworks. To minimize the code base, it gives up non-essential functionalities aggressively. It is not a library for you to experiment different algorithms. It is a library for you to use in your applications (LiuLiu, 2012).

### 3.1.4 Goals

Computer vision library is used in order to teach a computer machine to learn from video and images in order to understand real-world feel by detecting, classifying and identifying objects and events. In real world application, an example is the BMW parking assistant and NASA used this to land the Mars Rover. It is also used in industries where it is referred to as machine vision, where information is extracted for purposes of supporting a manufacturing process. A good example is a quality control where details of final products

are being automatically inspected in order to find defects. Computer vision aids in machine learning giving computers the ability to learn without being programmed, construction of this algorithms helps a computer learn and make predictions on data, this helps in overcoming strict static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Thus, computer vision library is needed to make sense of images and teach a computer to play the game Galaga.

### 3.1.5 Comparision Table

| Open CV | SimpleCV | Community Core Vision |
|---|---|---|
| - Libraries for most other languages(C,C++ and Python)<br>- Great for large scale programs<br>- OpenCV is best for implementation as it offers a lot more possibilities | - Interactive shell for testing<br>- Great for quick demonstration purposes<br>- Works very well with Pythons | - Uses statically-linked library<br>- Gives up non-essential functionalities aggressively<br>- It is not a library for you to experiment different algorithms |

### 3.1.6 Discussion

OpenCV is open source, free for commercial and research use under the BSD license. It is also cross-platform from, and runs on Windows, Linux, Mac OS X, mobile Android and iOS. With SimpleCV, you get access to several high-powered computer vision libraries such as OpenCV- without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap storage. SimpleCV is free to use, and because it is open source, you can also modify the code if you choose to. It's written in Python, and runs on Mac, Windows, and Ubuntu Linux and is also under the BSD license. CCV is an open source/cross-platform solution for blob with computer vision, it is published under an open-source LGPL license on Github to allow for others to contribute features and forks.

### 3.1.7 Best Option

CCV is a library for small applications. For quick prototyping SimpleCV is far superior, but for actual implementation/usage OpenCV offers a lot more possibilities.

## 3.2 Devolopment Operating System

### 3.2.1 Windows

Windows OS, computer operating system (OS) developed by Microsoft Corporation to run personal computers (PCs). Featuring the first graphical user interface (GUI) for IBM-compatible PCs, the Windows OS soon dominated the PC market (Britannica, 2015). Microsoft Windows operating system was developed by Microsoft to overcome the limitation of its own MS-DOS operating system.

### 3.2.2 MAC OS

The Mac OS or Mac OS is a series of graphical user interface-based operating systems developed by Apple Inc. for their Macintosh line of computer systems (Dong, 2007). Mac Os was designed only to run on Apple Computers. In 1984, Apple introduced the Macintosh PC with the Macintosh Operating System. Apple names its OS as "Mac OS", beginning in 19997 which previously known as "System".

### 3.2.3 Linux

Linux is an open source operating system, or Linux OS, is a freely distributable, cross-platform operating system based on Unix that can be installed on PCs, laptops, netbooks, mobile and tablet devices, video game consoles, servers, supercomputers and more (Beal). Linux is a Unix-like, Kernel-based, fully memory-protected, multitasking operating system.

### 3.2.4 Goals

Operating system will be used as the host environment for the deep learning project. It will be used to run the software of the image visualization as well as the neural network.

### 3.2.5 Criteria

Mac OS is a two layered system: the "attractive GUI" sits atop Unix core, and Unix is best-known for its security features. It's simply impossible to install a destructive Trojan or virus unless the user explicitly allows it root access via typing in the admin password. Mac OS's built-in firewall is set up to work unobtrusively out of the box as well as being highly configurable. Mac OS is incredibly stable. Apple controls production from start to finish, so every part of a Mac is designed and tested to work together.

A Microsoft window is easy to use as compared to other operating systems. Windows gives you an advantage of checking the performance of your PC. No need to install drivers of any peripheral devices. It is a windows operating system meaning every task is performed in its own window. Because of its wide use it has documentations and tutorials available to learn the OS. Windows offers support to the user as it has its own help section, and websites/forums where you can talk to people and gain information about Windows. There are also books available for each version of windows to help you. Windows has an administrative user that can make changes to the computer, but it has other users that can perform small tasks. You need the administrator's password to change anything if on another user.

Linux is a source operating system. Any programmer can modify it due to its needs. It is a free software that anyone can download it from web and install it on its PC. Many PC users can install Linux to its system there is no specific requirements for installing it. Linux has books for help and also online forums. Linux security requires a user to enter an administrator's password in order to make changes or download things, so it is harder for harmful programs to gain access to the computer.

### 3.2.6 Comparision Table

| Operating Systems | Security | Price | Compatibility | Reliability | Package Manager |
|---|---|---|---|---|---|
| Windows OS | There are many types of viruses in Windows OS because of a large amount of market share | Windows is less expensive than Mac OS | Anyone can install any type of game into Windows PC,as it is,compatible to all types of software and games | Windows is very reliable but due to its affinity to viruses it,struck | No official package manager |
| Linux OS | There are lesser amount of viruses in Linux as compared to,Windows because Linux is an open source OS | Linux software is totally free | Since Linux is an open source software you can program the software that you want | Since Linux is not a complete OS you can modify it for that,type of usage that you want | Most Linux distributions provide official package manager,software to install third party software |
| MAC OS | There are no virus in Mac OS because you can install Mac only,on Apple devices | Apple devices are too much expensive | Most software are not compatible with Mac OS so you feel,limited on Mac OS | Mac is very reliable and smooth because it has no viruses that,drag its performance | No official package manager |

### 3.2.7 Discussion

Though some operating systems like Linux may seem the better option because of its flexibility to be used in any kind of way, the choice is usually up to the user. If you're a gamer, then you have no choice, go for Windows. Software Developers might prefer Linux and video/graphics producers will probably tend towards Mac.

### 3.2.8 Best Option

After all is said and done, Linux eventually comes out as the winner. The reason Linux to be the better option is because it is powerful, runs on multiple hardware platforms. Users like its speed and stability, has no requirements for latest hardware. It is also free and licensed under GPL. Also vendors aredistributors who can package Linux. The ease to install software as it is also useful for software development purposes.

## 3.3 Programming Languages

### 3.3.1 Python

Python which is an interpreted, objected – oriented language which is similar to PERL as it has a high- level built in data structures with a dynamic binding making it attractive for a rapid application development since it is a scripted language for connection of existing components. These creates readable and hence reducing the cost of development and maintenance.

### 3.3.2 C++

C++ has classes which used in the form of an objective where it combines data representation as while as methods employed in the manipulation of the data in a neat package this include arithmetic, bit manipulation, logical operation.

### 3.3.3 Java

As for the Java this is a programming language which helps in creating applications that may run on a single computer Or other server in a network as the portable as it is fast, secure and most reliable and it is easier to learn however one should note that Java software cannot be installed alone hence a plugin software is needed.

### 3.3.4 Goals

The basic knowledge in programming is such a critique in that it has captured the most significant arc of human life. Slowly computer and software are taking the functions which affect our lives and has become an inescapable part of our life. In the modern world software products are used in selection form for the technique which improves the quality of the product of the software development. Today software is accurate, faster and cost effective in our today life in educational, professional as well as personal.

Choosing a programming language depends on the client, the need it solving, this happens in designing whereby the designer indicates guidance to the architecture of the project in detail on the system especially activities involved in designing, development, and enhancement as well as maintenance of the software. In the design phase, a specific system should be produced which gives necessary requirements hence ensuring maximum ease of use and effectiveness in the documentation where retrieval of information is crucial hence minimizing time as while as effort and more so the development cost. Since it involves carrying out a community consensus enhancing a significant number of people in working together in achieving a common goal. Thus, picking a programming language can bring ease of development as well as productivity to the project outcome.

### 3.3.5 Criteria

The criteria's for picking the right programming language depends on many important factors. The fist factor to consider is the documentation because as a developer well-written documentation is the best source for finding resources, explanation and language features.

The second factor is the community,the larger the community the better since help can be found easily on the internet and an example of that is stackoverflow.com. The third factor is the ease of execution which measures the steps for a software to execute and run. The fourth factor is used to measure the requirements for a programming language to run on an operating system. The fifth factor measures the standard libraries and data structures that come with the programming language by default. The last factor measures the codes density which compares the line of codes required to do certain tasks.

### 3.3.6 Comparision Table

| | Documentation | Community | Ease of Execution | OS Environment | Libraries | Code density |
|---|---|---|---|---|---|---|
| Python | Python official website contains well-written documentation as well as tutorials | large community | No need to compile. Need to run the main file then program will execute | Run on Windows, Mac OS, Unix | Built in Data structures and extra packages by default | Less line of code needed |
| C++ | No official website but cplusplus.com provide language references for C++ standard libraries | large community | Needs to compile first and then run the executable file | Run on Windows, Mac OS, Unix | Old standard does not come with data structures. Only standard 99 and standard 14 | More line of code needed |
| Java | Oracle provides documentation and tutorials on their official website | large community | Needs to compile first and then run the executable file | Run on Windows, Mac OS, Unix | Built in Data structures and extra packages by default | More line of code needed |

### 3.3.7 Discussion

All three programming languages have the same level of power based on the comparison table.

### 3.3.8 Best Option

Python is a great option for our project since it needs less step to run compare to the other programming languages. Also, python comes with wide variety of standard libraries and data structures such as arrays, dictionaries, and tuples.

## 3.4 Deep Learning Framework

### 3.4.1 Theano

Theano is a library for Python which focuses on simplifying and speeding up the calculation of complex mathematical expressions that use multidimensional arrays. It was originally created by LISA lab for use in quick development of machine learning algorithms. It uses its own compiler and functions as an extension to the Python programming language. Its optimizations allow for the translation of commands to GPU and CPU instruction code for increased speed of operation. Theano can perform graph optimizations and compile parts or all of a graph into CPU or GPU instructions [1]. It also has the ability to detect certain unstable expressions and replace them with much more stable algorithms for the program run. It supports tensors and sparse type operations. Theano provides many more functions related to graph evaluation and mathematical optimization. Theano is still in development and is currently at version 0.8 [1]. This technology is primarily useful in deep learning for its ability to interact directly with a GPU and process graph data. This allows for the deep learning algorithms made with it to process data at high speeds, though I have not found any solid metrics of how fast it can process a graph. The speed does depend on the quality of the GPU used and the nature of the graph being processed.

### 3.4.2 Caffe

Developed by the Berkely Vision and Learning Center, Caffe is designed specifically as a framework for deep learning. According to their website, it was made with "expression, speed, and modularity in mind," [2]. Caffe can interface with Python and Matlab, but its primary interface is command line. It uses its own schema to define layers in its deep learning network. Caffe uses what its developers call Blobs, which are essentially multidimensional arrays that wrap the data that will be processed or sent along by Caffe. Blobs are used to "conceal the computational and mental overhead of mixed CPU/GPU operation by synchronizing from the CPU host to the GPU device as needed," [2]. It can allocate memory as needed for its operation. The site claims they believe their program to be convnet implementation possible, though their metric of speed is applied to one specific GPU and probably doesn't apply to regular use [2]. The system supports optimizations for many different data operations required for interfacing with a neural network and training it properly. Caffe also has an optimization spec developed by NVIDIA specifically for running on their GPUs.

### 3.4.3 Torch

Torch is a computing framework that uses LuaJIT as its main code language interface. It has "support for machine learning algorithms that put GPU's first," [3]. Torch supports several models and functions necessary for scientific calculations such as linear algebra routines, numeric optimization, GPU support, n-dimensional arrays, indexing, slicing, and transposing among others [3]. Torch's Lua base provides is with incredible speed as compared to other programming languages, but this comes with the downside of less cross-compatibility. Lua is not a widely used programming language, so support for it is less common than support

for the bases of many other frameworks programmed in langauges like C, C++, and Python [5]. Torch provides tensor support and packages for machine learning and computer vision [4]. The GPU focus of torch makes it ideal for GPU based neural network training.

### 3.4.4 Choice of Framework

This project will be using Caffe. Caffe has widespread support for systems and has been aroundlong enough to have plenty of documentation on its use as well as examples of it in action. Caffe is coded at base in C++, which makes it easy to set up on most systems. The python framework provides support for an easy to use language that everyone in the group understands and can utilize to full potential. The system's integration with major neural nets is well documented, providing us a great base of research to get started. The final lynchpin for this decision is the existing optimizations for it that make it easy to run on NVIDIA GPUs.

## 3.5 Neural Network

### 3.5.1 Information

Regardless of which neural net being discussed, a few things need to be laid out to clarify what we mean by neural network and other terms related. A neural network is essentially a complicated decision graph built in layers by a machine learning algorithm. Each node is referred to as a neuron and has its own learned weights and bias. A convolutional neural network assumes all of its inputs are images, thus reducing the parameters for its use [13]. Because our project relies on visual input from a video game, we can process all incoming data as images. As such, a convulutional model network is ideal for this project. This section will cover the different convolutional models we could use to process our data.

### 3.5.2 GoogLeNet

GoogLeNet is a neural network design which uses Inception architecture. The Inception architecture is designed "to consider how an optimal local sparse structure of a convolutional vision network can be approximated and covered by readily available dense components," [10]. The idea behind this lies in how it deals with the layers that make up a neural network. It attempts to cluster these layers in optimal arrangements for program flow. It also attempts to reduce dimensionality of the network whenever computational requirements would get too high. The chief goal of the architecture is computational efficiency and practicality [10]. GoogLeNet is 22 layers deep. It uses about 100 layers to actual construct this network, though the exact number is reliant upon how many layers are actually counted by the machine learning infrastructure [10]. The document covering GoogLeNet contains a rough estimate that, given the dataset used for its premier competition, the network could be trained to convergence using a high-end GPUs within a week. The biggest limitation of training speed is memory usage [10].

### 3.5.3 AlexNet

AlexNet is a neural network architecture designed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton at the University of Toronto. It was trained originally with a dataset of "over 15 million labeled high-resolution images belonging to roughly 22,000 categories," [11]. The architecture contains eight learned layers; five of these layers are convolutional and the other three are fully connected [11]. The architecture is designed to be slit across two GPUs, with each GPU running separate layers. The GPU's only communicate between each other on specific layers. The convolutional layers filter the image size for proper processing. The network has 60 million parameters each fully connected layer has 4,096 neurons [11]. The architecture uses two forms of data augmentation to ensure that the images are processed properly despite filtering size down. The aim is to reduce complexity of calculation and increase speed. The first method consists of image translations and horizontal reflections, while the second consists of altering intensities of RGB (Red Green Blue) channels in the images used to train the network [11]. This network achieved a test error rate of 15.3% in the ILSVRC-2012 neural network training competition, placing it among the top 5 entrants.

### 3.5.4 OxfordNet

OxforNet is a neural network architecture designed by Karen Simonyan and Andrew Zisserman at the University of Oxford. One of the key features of this architecture is that all of the filters used in the convolution layers for image processing are quite small. Most other convolutional neural networks work down from slightly larger filters, but OxfordNet remains steadier in its filter size. It makes up for the lost image depth through having 16-19 weight layers in the network [12]. The training methods of this network are similar to those used for AlexNet, as this framework was built to improve upon that architecture. The weights of the node layers at initialization plays a key role in affecting how long it takes the net to learn properly [12]. The structure requires multiple GPUs to practice parallelism in its data processing. The network on its own achieved a 7.3% test error rate, and when two models were combined the error rate was only 6.8%.

### 3.5.5 Single Shot Multi-Box Detector

Single Shot Multi-Box Detector, or SSD for short, is a convolutional neural network designed by a team of people cooperating between the University of North Carolina, Zoox Inc, Google Inc, and the University of Michigan, Ann-Arbor [21]. The key focus of SSD in comparison to other convolutional neural networks is its focus on individual object detection within an image. SSD is based on a feed-forward model that uses well established architecture for image classification on the first few layers, then dives into feature layers which rescale the image and produce bounding boxes on image features. It uses predictive logic in there layers to predict the offsets of the bounding boxes and eventually is able to class objects within the bounding boxes [21]. The training requires extra parameters compared to the training of a usual neural network, as defaults must be set for the bounding boxes and other factors which influece how SSD recognizes image components. Compared to other neural networks, SSD is much faster at identifying image components in real time, reaching frames per second values in the 40's for how many images it can process in real time. Confidence

of detection ranges from around 50% to up to around 76% confidence that the objects are identified correctly, which just shows that SSD's speed does come at a cost when evaluating it in terms of accuracy.

### 3.5.6 Choice of Network

After reviewing the documentation, implementations, and performance of the neural network architectures available, we initially thought to use a GoogLeNet model, but SSD has proven itself to be much more applicable to our project upon further research. SSD is an incredibly fast and powerful algorithm that gives us the tools to identify in game objects while the neural net plays Galaga. The other architectures are only set up with the idea of classifying whole images, while SSD is powerful enough to break down the component parts. This is essential to allowing our neural network to accurately understand the game state and make more informed play decisions. The speed is also an important factor in this decision, as the high processsing speed of a fully trained network is ideal for the high fps environment of a game like Galaga. The speed of detection will allow the game to make decisions faster, which makes its overall skill potential higher. Given the nature of the task, SSD has presented itself as an ideal model for completing the project.

## 3.6 GPU Cloud

### 3.6.1 EC2

EC2 is a cloud computing environment set up by Amazon Web Services. The EC2 cloud supports a wide array of practical features for computing on a large scale. The system supports high performance computing with many more cores than the average CPU. The cloud provides "up to 16 NVIDIA K80 GPUs, with a combined 192 GB of GPU memory, 40 thousand parallel processing cores, 70 teraflops of single precision floating point performance, and over 23 teraflops of double precision floating point performance," [6]. The GPUs are capable of supporting CUDA and OpenCL frameworks. Machine learning is specifically listed as a taskwhich these GPU's are able to support, which makes it a good fit for a deep learning project. The high GPU count also allows for high real time graphics processing, which pairs well with things that require graphical input. The cloud supports many more features such as security, networking, and auto scaling, but the high GPU count is the most impactful to its use on this project. Pricing for on demand use varies per server and memory needs from less than a dollar per hour to around $17/hour.

### 3.6.2 Nimbix

Nimbix is a cloud computing and storage company. Their cloud supports several different types of NVIDIA GPUs, including the K40 and K80. The cloud supports OpenGL, OpenCL, and CUDA environments [7]. The cloud offers up to 128 GB of RAM and 1 Terabyte of storage. The JARVICE platform on their cloud offers support for big data and heavy computation. It contains an API for use specifically with GPU's. It uses custom Linux systems, referred to as Nimbix Application Environments, to build and run application on the cloud [8]. The environments have full access to the CPU, GPU capacity, and memory of

the underlying hardware. Nimbix offers pay rates for up to 16 cores and up to two NVIDIA Tesla K80 or K40 GPUs. The price of using two K80s hits $5.00/hour. The Nimbix cloud has a lot of articles and documentation surrounded its uses in high power computing and deep learning.

### 3.6.3 Cogeco Peer 1

Cogeco Peer 1's cloud offers customized GPU server solutions with NVIDIA Tesla series GPUs [9]. Cogeco Peer 1 does not offer on-demand server use. To use their cloud, one must rent a physical dedicated server from them. They provide setup and management for the cloud server. They tailor GPU solutions to the tenant's needs. They can set up the server with several operating systems including Windows and Red Hat. Their website promises rapid response to any server issues experienced while using their product. The customizability of the setup offers a concrete way to ensure the cloud has the proper setup for our group needs, but requires much more setup time. The company seems to largely focus on business solutions rather than individual use, so it's not very well suited to our project needs.

### 3.6.4 Choice of Cloud

Given the information we have on cloud computing systems, we have chosen to use Amazon Web Services EC2 due to its convenient use, high availability of powerful GPU processing, and support for frameworks we'll be working with. Furthermore, our client has offered to handle the costs of this cloud computing platform, which means that we will not need to pay out of pocket to use its services. Nimbex is a very tempting second, as it offers very good pricing and a lot of NVIDIA GPU support for Neural Network training. However it isn't quite as powerful as the Amazon cloud. Cogeco Peer 1 is much too focused for long term business solutions to be a reasonable option on this project. It also offers less power than the Amazon cloud.

At the present moment this section mostly remains as a reminder of technology considered for use, as well as remaining in the possiblity that we do end up using a cloud as training amps up for the project. We have not yet had to use any cloud computing for our training, but as we add more images it may come to our best interest to use a cloud to hasten at least the first stage of training the system.

## 3.7 Communication

The role of making sure everything runs smoothly on the platform required of us to implement our solution on can be further devided into supporting goals. These goals are finding the right communication methods to use with our client for group meetings, deciding what camera technology we need and how we want to record our input for the program, and figuring out the best way to visialize the data. For each of these three tasks I have found three approaches that I have research and given thorough consideration. For each approach I have given a paragraph describing some of the costs and benefits as well as my informed

opinion on which approach is the best for the task. This is to ensure that we are making the best possible decisions when working with the Jetson TX1.

### 3.7.1 Skype

One of the most omportant aspects of a group project is communication. This is especially true when we're working with a client who has tasked us with using a certain platform. There will most likely be many times when we'll need to discuss something about the Jetson. There are many very popular and suitable tools available to conduct online meetings. These include Skype, Discord, and Team Speak. Skype was founded in 2003 by Niklas Zennström. Skype has a lot of features that Discord and Team Speak don't have. One of these features includes screen sharing. This feature could be very beneficial during group meetings when a group member has to explain something visually. Our group has used Skype twice in the past and the sound quality was good enough for everyone to understand each other. Discord and Team Speak also do not allow video calls, they only provide groups to voice and text chat. One down side with Skype however is the presence of bugs. Among reviewers, skype is the least reliable. A huge factor in our decision on with communication tool to use is what everyone is comfortable using. Since most people are alread familiar with skype and our group has alread conducted two meetings with it, skype might be the best tool for the job.

### 3.7.2 Discord

Discord, which is the most recent VoIP service released, is known for its very nice user interface. This service also does not require a download. Users can log on and join a Discord server using their browser. Discord hosts its own servers which are very reliable. Joining a call is very simple. Whoever is hosting the server simply sends each group member a link. Other group members simply click the link and join the voice server using either their web browser or downloaded application. It also has a very good chat system. Call quality is not limited to a group members poor internet. Discord is also very light weight and puts much less strain on the cpu than Skype and Team Speak. However, Discord does not have many of the Skype features including video calling and screen sharing. The user interface might offer some advantages but since everyone is already familiar with skype its not a very big advantage. The sound quality was also good enough for our purposes.

### 3.7.3 Team Speak

Team Speak is best known for its sound quality. Team Speak is the second oldest voice chat service, which has been in development since 2004. The newest version is Team Speak 3. Team Speak has the greatest sound quality and reliability out of all three VoIP applications. One of the big problems with Team Speak however is its user interface. For beginner users it can be a lot more difficult to figure out. Many reviewers say that it has the worst user interface design compared to Discord and Skype. This is probably a product of its customizability. There are many ways to improve a users experience with Team Speak including downloading user interface package downloads. If there are only a couple of us in the group already familiar with how to use Team Speak then it probably isn't the best

choice. The better sound quality isn't enough to merit learning how to use Team Speak since its not very necessary for out group meetings. However, if everyone is already accustomed to using Team Speak then our group might be able to benefit from Team Speak's greater flexibility.

## 3.8 Camera

### 3.8.1 Leopard Imaging

For deciding what kind of camera we should use, there are a lot of factors to consider. The camera will need to be able to record in HD with at least 30 frames per second. The program will need to be able to make out shapes for its visual computing. Leopard Imaging had partnered with NVIDIA to build cameras spceifically for the Jetson TX1. This camera meets the miniumum requirements and was built specifically for machine vision. There are many other HD cameras and Camcorders that are much less expensive but meet all the requirements. However, the project might benefit from a camera built for visual computing.

### 3.8.2 NVIDIA Shadowplay

There is also the option of using a screen capture service. NVIDIA Shadowplay is a very convenient and easy to use tool for recording one's screen. Shadowplay can record in HD at 60 frames per second and it doesn't require purchasing extra hardware. It is very reliable software that puts very little strain on the cpu. This also might be a very good tool to use since it's very easy to set up. I myself have had a lot of experience using Shadowplay and our team would have no problems getting started with it.

### 3.8.3 Client Recommended Camera

There are many very good resources we can use for gathering the input for our program. A decent HD camcorder could do the trick and would only cost about $100. The first camera mentioned, which was designed for visual computing would cost about $400. There is also a camera that our client is supplying us. This is likely the best camera our team should use since it is provided freely and meets the needs of our project. The camera is also recommended by our client for use with the Jetson TX1. Our client is shipping our team this camera along with the Jetson.

### 3.8.4 Camera we ended up using

After further research in our prject we realized that a good camera didn't necessarily have to be very high end. The camera we needed simply had to capture frames at a moderate speed and clearly video the galaga gameplay from a short distance. Our project could easily process images at as low as 10 frames per second and still be able to play the game fairly well. Therefor we saw no need to request anything that cost more than $100. We were able to buy an inexpensive camera for around $30 and was still more than we required. It is a three megapixel camera that records at 720p.

## 3.9 Data visualization

### 3.9.1 Live Data

The third task involves figuring out how to visualize the data gathered from the program. There are many ways in which this could be done. We could have live data visualization, that shows the time, number of lives, number of near misses, and other such useful information to help gage how well the program is performing during the current playthrough. We could have live data visualization that is also graphical and includes graphs and figures. We could also have the data only appear at the end after the program has completed so that only the end result is displayed. Live data visualization can offer a few things for data analysis. In the beginning this might be very useful because during the start of the project the program may only survive a few seconds of the game. Later however, when the program is beating more complicated levels, data gathered during the playthough could indicate areas where the program struggles.

### 3.9.2 Live Visual Data

Live data that can be displayed visually could enhance the benefits of seeing data gathered thoughout the playthrough. Graphs that recorded which level the program lost a life at and what was the thing that hit it could easily be displayed visually to enhance comprehension. There could be a lot of benefit to seeing a bar graph go up and down throughout the playthrough as the program loses or gains a life. We could comparatively see how long it takes the program to complete each level. This could be seen in real time and compared to how the program is actaully performing in the game. One problem with live visualization is that it could also flood the screen with too much information, especially when the playthroughs get really long.

### 3.9.3 Visual Summary of Data

One benefit to having a smaller amount of data appear at the end giving a summary of the playthrough is the simplicity. An end of game summary could provide more of the information we need and leave out a lot of unnecessary items. A lot of information would be floating by on the screen as the program played the game. A concrete summary might be a lot more efficient if it could compile all the data in a meaningful way. Visual timelines could be presented as well to get a quick idea where the program struggled. A lot of the live information can also be gathered by just watching the playthrough. We'll be able to see when the program lost a life or suddenly could not compute how to avoid objects. We'll also be able to see what objects the program might have trouble seeing, which would be a hard thing to discover with just the data. I think this idea is the best approach. There are a lot of creative ways that all the data gathered throughout the playthrough of the game and displayed in a summary at the end. Having a live stream of data doesn't seem very useful. We might get more efficient feedback if it was all summarized at the end.

# 4 Bibliography

[1] "Theano at a glance — Theano 0.8.2 documentation," in deeplearning.net, 2008. [Online]. Available: http://deeplearning.net/software/theano/introduction.html.
[2] "Deep learning framework," in caffe.berkeleyvision.org. [Online]. Available: http://caffe.berkeleyvision.org/.
[3] "Scientific computing for LuaJIT," in torch.ch. [Online]. Available: http://torch.ch/.
[4] torch, "Torch/torch7," in GitHub, GitHub, 2016. [Online]. Available: https://github.com/torch/torch7/wiki/
[5] zer0n, "Zer0n/deepframeworks," in GitHub, GitHub, 2016. [Online]. Available: https://github.com/zer0n/de
[6] "EC2 product details - Amazon web services," in Amazon Web Services, Amazon Web Services, 2016. [Online]. Available: https://aws.amazon.com/ec2/details/.
[7] Nimbix, "NVIDIA cloud computing - GPU cloud," in Nimbix, Nimbix. [Online]. Available: https://www.nimbix.net/cloud-computing nvidia/.
[8] "Working in the Nimbix application environment (NAE)," in Nimbix, Nimbix, 2015. [Online]. Available: https://nimbix.zendesk.com/hc/en-us/articles/201740897-Working-in-the-Nimbix-Application-Environment-NAE-.
[9] C. P. 1, "Managed GPU servers — affordable high performance computing," in Cogeco Peer 1, Cogeco Peer 1, 2016. [Online]. Available: https://www.cogecopeer1.com/en/services/hosting/servers/ma
gpu/. Accessed: Nov. 16, 2016.
[10] C. Szegedy et al., "Going Deeper with Convolutions," in cs.unc.edu. [Online]. Available: https://www.cs.unc.edu/ wliu/papers/GoogLeNet.pdf.
[11] A. Krizhevsky, I. Sutskeever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in papers.nips.cc, University of Toronto. [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.
[12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Neural Networks for Large-Scale Image Recgonition," in arxiv.org, University of Oxford, 2015. [Online]. Available: https://arxiv.org/pdf/1409.1556v6.pdf.
[13] A. Karpathy, "CS231n Convolutional neural networks for visual recognition," in GitHub. [Online]. Available: http://cs231n.github.io/convolutional-networks/.
[14] C. Gulcehre, "Deep Learning Software Links," in deeplearning.net, 2016. [Online]. Available: http://deeplearning.net/software_links/.
[15] NVIDIA, "GPU cloud computing solutions from NVIDIA," in nvidia.com, 2016. [Online]. Available: http://www.nvidia.com/object/gpu-cloud-computing-services.html.
[16] Beal, V. (n.d.). Linux OS (Operating System). Retrieved 11 13, 2016, from Webopedia: http://www.webopedia.com/TERM/L/linux_os.html
[17] Britannica, h. E. (2015, November 16). Windows-OS. Retrieved 11 13, 2015, from Encyclopedia Britannica: https://www.britannica.com/technology/Windows-OS
[18] Dong, J. (2007). Macintosh Operating System. In J. Dong, Network Dictionary (p. 298). Javvin Technologies Inc.
[19] Sanner, M. F. (1999). Python: a programming language for software integration and development. J Mol Graph Model, 17(1), 57 61.

[20] Selic, B. (2003). The pragmatics of model-driven development. IEEE software, 20(5), 19.
[21] W. Liu et al., "SSD: Single Shot MultiBox Detector," arxiv.org, 29-Dec-2016. [Online]. Available: https://arxiv.org/pdf/1512.02325.pdf.