Design Document

Deep Learning on Embedded Platform

Version 1.0

Prepared by Christopher Johnson, Luay Alshawi, Gabe Morey

CS 461 Fall 2016

November 29, 2016

This document goes over the design decision made for our Deep Learning on Embedded Platform project. In it we will explain our approach, provide detailed information about why we designed it a certain way, and explain our expected outcome.

Contents

1 Overview **3**
- 1.1 Scope . . . 3
- 1.2 Purpose . . . 3

2 Definitions **4**

3 Design **5**
- 3.1 Introduction . . . 5
- 3.2 Development viewpoint . . . 5
  - 3.2.1 Set up image dataset . . . 5
- 3.3 Neural network setup . . . 5
- 3.4 Data viewpoint . . . 5
  - 3.4.1 Image processing . . . 5
  - 3.4.2 Neural training methods . . . 5
  - 3.4.3 Data visualization . . . 5
- 3.5 Hardware viewpoint . . . 5
  - 3.5.1 Hardware setup . . . 6

4 Conclusion **8**

1 Overview

1.1 Scope

This document outlines the design of a system for teaching a deep learning neural network how to play the game Galaga. The sections of this document will outline the system details, setup, and all things necessary to understand how and why the project is designed. Hardware required, API's used, and setup will be detailed as part of the project design.

This project is to be used to produce a training course for NVIDIA's Deep Learning Institute. After the project is complete it will be handed off to NVIDIA for the purposes of accomplishing this goal.

1.2 Purpose

The main aim of this document is to give others a thorough enough understanding of the project and its design. The reader of this document should walk away able to understand how the system works and why it was designed this way. The document must be thorough enough for NVIDIA to recreate the project and package the design into a training course on deep learning.

2 Definitions

3.1 Neural Network: A multilayered graph structure containing nodes, referred to as neurons, and weights that affect how likely the node is to be chosen when confronted with a decision. The weights of nodes are decided through training the network, and a traversal of the graph acts as a large chain of decisions as the neural network is confronted with situations.

3.2 Deep Learning: Using a layered neural network constructed by GPU processing to teach a computer how to accomplish a specific task.

3.3 Convolutional Neural Network: A neural network specifically designed to take images as input. It uses convolutional layers to process the image into a more manageable chunk of data and extracts the information relevant to the network.

3.4 Caffe: A code framework that allows for complex mathematical operations that are required for training a neural network.

3. API: Application Programming Interface

3. Jetson TX1: Quad core embedded system designed for power efficiency and deep learning projects

3. OS: Operating system

3 Design

3.1 Introduction

3.2 Development viewpoint

3.2.1 Set up image dataset

3.3 Neural network setup

3.4 Data viewpoint

3.4.1 Image processing

3.4.2 Neural training methods

3.4.3 Data visualization

We also needed to figure out how we were going to display the results of each round. We decided it would be best to have a summary graphical display of the data at the end of the playthrough. Once the program runs out of lives a screen will appear graphing statistics about the machines performance. These statistics will include things such as how long the machine lasted, how many levels it completed, which enemies or objects took away the most lives, and so on. Having detailed and easily readable data is key to knowing how to train the machine as we progress through the project.

3.5 Hardware viewpoint

While working with the Jetson TX1 there are several things that we needed to decide. These included our methods to communicate with our client, and the hardware used with the Jetson. For communication purposes, our team decided to stick with using skype. We will conduct regular meetings with our client and continue using skype, which everyone is familiar with. Skype will allow us to video call as well, in case any visuals need to be explained.

Our project will actually need to be able to control the game. Inputs from the neural net need to be received by the game so that it can be played. To do this we decided not to use the standard keyboard. Using a keyboard requires a lot of additional coding to map the Jetson to the controls. Instead we decided to find a game controller that we will hardwire to the Jetson TX1.

3.5.1 Hardware setup

A game controller will significantly reduce the work load because it will already contain a lot of built in functionality that we need to connect it to the Jetson. The Jetson will be able to control the game using the controller's built in functionality.

Figure 3.1 shows the basic layout.

Since the Jetson has to learn how to play the game it needs to be able to see the game. For this we decided to use the camera sent to us by our client. This was the simplest option seeing as we can connect the camera directly to the Jetson. The camera will be connected to the Jetson and aimed at another screen displaying the game. The Jetson will first be trained by being given the video input to recognize and characterize objects within the game. We will set up tables so that the neural net will be able to store information such as the locations of objects. Once those things are set up it will be able to train itself by playing the game using the controller. It will base its decisions on the table of information that it stores and will increasingly be able to survive the game longer. Eventually it will learn to respond to certain patterns in the table in order to become better at the game.

4 Conclusion