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

In this document we describe the design of a deep learning system developed to learn how to play the arcade game Galaga. Galaga is an arcade shooter which was released in 1981 by Namco [1]. This system is designed with the ultimate goal of being turned into a course for the NVIDIA Deep Learning Institute. The documentation will outline exactly what hardware was used, how the system was put together, and what methods were used in a way that allows others to recreate this project.

Contents

1 Overview

1.1 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.

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. To that end the documentation has been designed to provide readers a clear path towards building this system themselves.

1.2 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.

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.

3.5 API: Application Programming Interface

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

3.7 OS: Operating system

3.8 OpenCV: (Open Source Computer Vision) is a library that can be used with different languages(C, C++, Java, Python, etc.). It provides standard functionalities such as image capture, Faces recognition, Gesture recognition, Motion tracking, Mobile robotics, Object identification and Image manipulation.

3.9 Galaga: Galaga is an arcade video game released in 1981 by Namco [1]. The player takes control of a small space ship and must shoot at and dodge waves of alien space ships. Victory in a level is achieved by destroying all enemy ships.

3 Design

3.1 Introduction

This section lays out several viewpoints from which we have approached the design of this project. Each component will be matched under an appropriate viewpoint that gives the reader a clear idea of how the design was apparoached. Each subsection details a different set of components, many of which will be lumped together, which constitute a major piece of the final design. There are three main viewpoints from which we have approached the design:

- Developer Viewpoint: Design elements approached from the eyes of a developer trying to recreate this project.
- Data Viewpoint: Design elements approached from a perspective of data flow operations and data processing.
- Hardware Wiewpoint: Design elements approached from the perspective of the hardware level. Particularly focused on hardware setups required for the rest of the process.

Each viewpoint is covered in more depth in its respective section.

3.2 Development viewpoint

3.2.1 Set up image dataset

OpenCV is a useful tool for this deep learning project. It will be used as a helper tool to prepare the images for each object in the game such as players, enemies, and bullets. A script will be written in python which will use an OpenCV to crop, resize, and filter the game images when needed. The goal is to have a full set of the game images and their labels which will be used to train the neural network. However, using OpenCV at this phase is optional, but due to the need of fast image manipulation, OpenCV can be used to do this task easily as it will help to save time and efforts. Although, collecting objects' images can be done manually, eventually, OpenCV will be used to resize the collected images to a consistent size.

3.3 Neural network setup

3.4 Data viewpoint

3.4.1 Image processing

After having a trained neural network, a software written in python will start by loading the necessary dependencies which are OpenCV, Caffe, and the trained neural network. At this step, it's important to convert the given neural network to an object that OpenCV can understand. Then, a video capture of the game will take a place to record the monitor where the game is being played using OpenCV's API [1]. Since videos are just consecutive of images, OpenCV will be used to process the images by comparing them with the trained neural network. The idea is to identify the location of each dangerous objects that would reduce the player's life so the Jetson can send a command to the game controller which moves the player away from the harmful object.

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. This would require a lot of additional coding to map the Jetson to the controls. Instead we decided to find a game controller that could be hardwired to the Jetson TX1.

[1] The hardware connection of the camera as well as the control mechanism of the game shall be explained in a different section.