Parallelized Interactive Machine Learning on Autonomous Vehicles

Xi Chen
Markey Cancer Center
University of Kentucky
billchenxi@uky.edu

Caylin Hickey
Research Computing Infrastructure
University of Kentucky
caylin.hickey@uky.edu

Abstract

We report on the use of parallelized interactive machine learning with AirSim, a simulator for drones and cars built on Unreal Engine by Microsoft. The early work with platforms like AirSim, or similar simulated learning agents, has been focused with deep learning, computer vision, and reinforcement learning algorithms for self-driving/autonomous vehicles. The advantage of these approaches is the ability to train in an environment where the cost of failure is reduced to merely time rather than loss of vehicles or environmental damage

Introduction

Autonomous wayfinding has become a large market in recent years, with proposed applications and major breakthroughs in self-driving cars frequently making headlines in the technology community. While their developments and milestones garner much praise in the news, the companies that are working on these systems work tirelessly to keep the data that is used to train their navigation, and with good reason.

In a standard supervised learning environment, one in which annotated data is used to develop a classification system, such as classifying the next appropriate action to take, the amount of data needed to ensure the competency of the resulting model can at times be unknowable. This is a result of trainer having little insight into what the accuracy of the trained model is at any given time as it pertains to real-world performance.

In this paper, we describe a method to leverage interactive machine learning to expose to the user real-world performance of model accuracy to facilitate more targeted training feedback. This is accomplished through the use of a highly detailed simulation system that combines the physics simulation systems built into the Unreal Engine with driver and camera system simulations developed by Microsoft.

Progress

The authors admit to the scale of the problem they have they have selected to pursue and have worked hard to tackle some of the problems inherit in using a system this complex. We report here the progress we have thus far made on the project.

Project Design

Most elements of the project design have been discussed and are in the process of being implemented. A brief overview of the project is as follows:

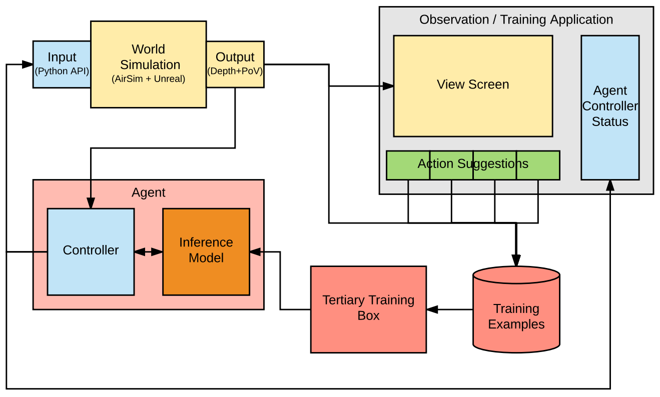

Figure 1: Training Pipeline

Agent The agent is to be written in Python with a module that takes input from the AirSim API location and exportation function, infers its next action from the current model, and reports the selected action back to AirSim to advance its movement towards the intended goal. The agent will also report its operational conditions to the Observation / Training Application, described in the next section.

Observation / Training Application In order for the oracle to provide feedback to the agent, it must have a view into what the agent is doing. Here we provide the oracle with an export of the current view of the agent, the set of actions from which the agent will select provided as buttons for oracle suggestion, and the current operating conditions of the agent (what action it has selected to perform next.) Upon user provided feedback via the pressing of one of the action suggestion buttons, the application will call the AirSim API to export an image, annotated with the selected action, and upload that action to the repository used by the Tertiary Training Box, described in the next section.

Tertiary Training Box This machine will pull down the annotated examples, retrain the model using a network similar to that described for autonomous MAV navigation in trails, called TrailNet (Smolyanskiy et al. 2017), and upload the new model to the agent. This machine has been tested for compiling models via Tensorflow to ensure suitability for this experiment.

Experimental Setup In this experiment, we are hoping to show how the principles of interactive machine learning can shorten the length of time it takes to train a model suitable for navigation in autonomous vehicles. As such, we must also have a control. In this case we will be comparing our system to one trained in the traditional supervised manner, specifically by annotating an entire oracle-controlled run and using that as training data for an agent. This will then be compared to using multiple runs where an oracle merely suggests better actions as they see the agent struggling to perform correctly.

AirSim

AirSim (Shah et al. 2017) is a simulated environment for vehicular control built upon the Unreal Engine by Microsoft. It provides controller access via flight controllers, Xbox controllers, or various API methods by which actions can be given to the vehicles to control their movement. It is in the environment that we plan to conduct our experiment.

Installation The installation of the Unreal Editor and AirSim have proven to be more of a challenge than initially thought and added considerable delay to current developments. Both authors of this paper conduct their work via MacBooks that run macOS, which is the one operating system the won't natively build both applications necessary for this work. Fortunately, after much experimentation, both authors have managed to get the program to compile and run under both Windows and Ubuntu-flavor Linux operating systems to a degree suitable to finish the project.

API The authors have conducted preliminary testing to familiarize themselves with the Python version of the API used to interact with the AirSim agent inside the Unreal Engine simulation.

Left To Do

While we have most of the large pieces required to finish this project, ample work is left to do. That work is described in the remaining sections of this paper.

Agent

The agent, as described above, still needs to be completed. As mentioned in the progress section, the Python API layer is roughly working. What remains to be completed is integrating this with both the inference system from a compiled model as well as the posting of the agent's state to the training application, to be described in a later section.

Supervised Learning Control

As described before, for a comparative control we intend to build a similar network to that used in TrailNet for use in our experiment. TrailNet itself is a modified version of ResNet with trail specific data for loading the weights the network. Initially this will be trained by an oracle (likely one or both of the authors of this paper) by navigating the course and extracting actions / training data to provide to the tertiary box to be compiled into an inference model until the agent can accurately navigate one run of the course without collision.

Training Application

The training application, used in the interactive machine learning portion of this project, will consist of two parts: the user interface to allow the oracle to interact with the system, and an API layer to allow the agent to push its current state to the oracle to allow for accurate feedback to be given. The design of the interface has been roughly assembled, but still remains to be implemented. For the agent-to-oracle API layer, there still remains work to accomplish on what is relevant in the agent's state. Once this is established, the programming required for the API layer should follow quickly.

Tertiary Training Box

While the tertiary training box has been tested in basic Tensorflow work flows, it needs to have the specific network training code for this project as well as defined storage for the training examples that will be provided. To clarify, storage is available on this machine, it just needs to be defined for the training application and subsequent training script such that the model compilation work flow will proceed automatically when a run is completed.

References

Shah, S.; Dey, D.; Lovett, C.; and Kapoor, A. 2017. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *arXiv preprint arXiv:1705.05065*.

Smolyanskiy, N.; Kamenev, A.; Smith, J.; and Birchfield, S. 2017. Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness. *arXiv preprint arXiv:1705.02550*.