Introduction
Grand Theft Auto V
The Dataset
Neural Network
Motion Simulation
Results
References

GTA V SELF DRIVING CAR WITH MOTION

Evan Miller, Michael Dahl, John Apo, Alex Redei

Redei Labs: Central Michigan University

July, 2021

An Overview

- Grand Theft Auto V
- The Dataset
- Neural Network using RNN and CNN structures
- Motion Simulation
- Results
- References

The Project

In this project, we set out to create an autonomous driver for the game Grand Theft Auto V. This will be accomplished by training a Neural Network on a set of images taken from actual players driving in the game. As a secondary goal, we aim to integrate the game's telemetry data tracking into a program that controls a motion simulator in our lab.

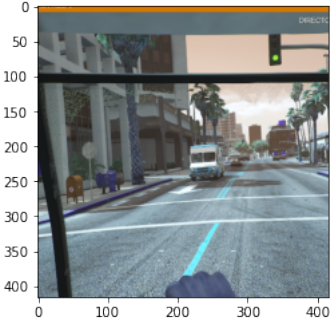
The Project (1)

This project is an attempt to improve upon the methods used in Aiden Yerga Gutierrez and Iker Garcia's project *GTAV-Self-driving-car* [A]. In their documentation, they noticed that their network was driving by focusing on the path highlighted on the GPS while ignoring the actual environment (This behavior can be observed in this recording of the model being tested `https://youtu.be/M-fWHox6ke0`). With this in mind, we have 3 goals for the AI portion of this project

1. Remove UI components that allow the model to "cheat" as it learns
2. Encourage the model to react to objects like cars and pedestrians by highlighting them in the dataset
3. Rewrite training functionality using Generators to allow for easier modification and ease of updates to modern versions of Tensorflow

The Game

Grand Theft Auto V [G] is an open-world rpg game where players commit crimes throughout the city of Los Santos. Our group has chosen this game for our project because it features an expansive world with realistic graphics. It also has an active modding community, which made it easy to find information on how to interface with the game.



Settings

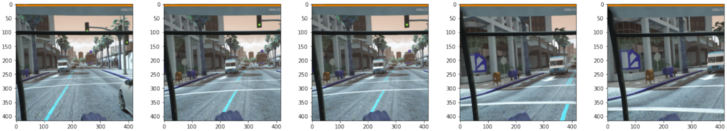
- Resolution is set to 1280 x 1024, and the aspect ratio is set to 5:4 (these settings are for desktop monitor viewing, other kinds of screens would require alternate settings)
- Game window is placed in the top left corner of the screen using WinExplorer [WE]
- Director Mode is used to enable the following cheats
  - Time of day is Morning
  - Weather is Clear
  - Wanted Status is Disabled
  - Restricted Areas are Off
  - Invincibility is On
- First Person Auto-Center is turned on
- Radar, HUD, GPS Route, and Subtitles are turned off

The Route

To minimize the amount of data necessary to train our model and reduce ambiguity of player actions at turns, our training drivers followed a specific route, shown below. They first got in a Golf Cart at the Golf Course, then drove along the yellow path to the circuit outlined in green before recording themselves.

Image Sequences

As mentioned our datset is comprised of screenshots taken while actual players drove around our training route. These screenshots are grouped into sequences of 5 and saved as 3 channel Numpy arrays, which are then paired with the combination of buttons pressed on the last image. We choose to analyse sequences of images because they encode information about momentum and direction of movement for our model.

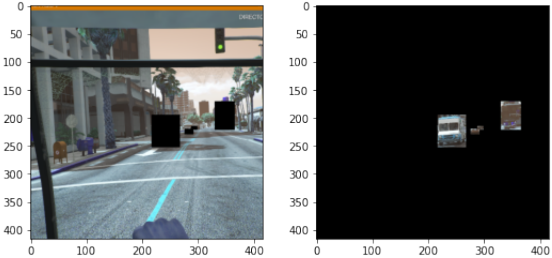


Object Detection and Highlighting

After this, we utilize two techniques to highlight the objects in our images. Both of these methods are based on utilizing the YOLO object detection algorithm [Y] pretrained to recognize objects that are commonly found on the road such as cars, bicycles, and people. This network was trained by Lavanya Shukla as part of the Lyft 3D Object Detection for Autonomous Driving challenge on Kaggle [TY].

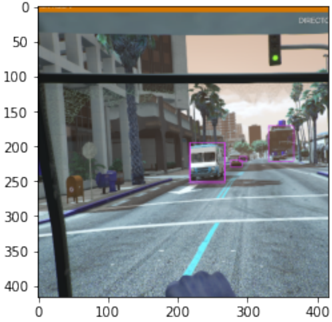
Object Detection and Highlighting: Separating Images

Our first method uses our object detection model to create bounding boxes for any objects found in our image. The coordinates for these boxes are then used to create a mask that we apply to our images to separate them into two: one that contains objects and one that contains the background. Each image will be sent as an input to separate Neural Networks, which are later concatenated, so as to allow the model to react to the environments and objects differently.

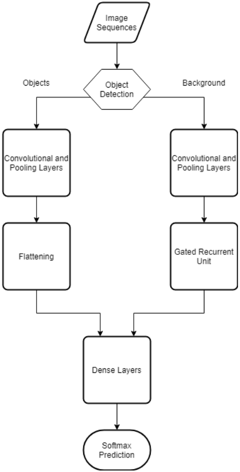


Object Detection and Highlighting: Object Framing

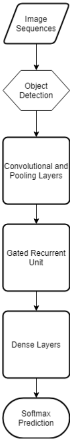
For our second method, we will draw our bounding boxes directly onto our images. While this requires that both the objects and backgrounds are processed by the same network, it greatly reduces the complexity of the inputs to our model, which theoretically makes it easier for our network to learn patterns.



Separate Image Model



Highlighted Image Model



Network Overview

The hyperparameters for both networks are adapted from Aiden's project, since we had little time to tune our model ourselves. The training process was completed using a custom Tensorflow data generator that reads data from our saved arrays and preprocesses them by performing object detection before grouping them into arrays. This was done as the entire dataset of images is too large to fit into working memory in its entirety. We also chose to calculate training weights for our classes using the formula $\frac{\textit{num largest class}}{\textit{num class}}$ where *num class* is a count of all instances of a specific key combination in our dataset and *num largest class* is the maximum of *num class* over each class. We used this method instead of balancing our dataset because it is important that the long straightaways are all recognized, and balancing the dataset significantly reduces the model's exposure to these areas.

Our Motion Simulator

We used our labs custom Motion Simulator, which has been in development since 2017, for this project. The cockpit came from Tennessee, the motors came from Florida, and the inverters came from Utah. We have two primary electric motors which are geared to deliver approximately 2000 ft-lbs of torque in an instant, which is roughly the equivalent of two semi trucks chained together. The simulator can be seen in the video in the Results section.

The Plugin

We wrote a custom plugin using OpenIV [O] to connect our simulator to GTA V. Fortunately, the game keeps track of the telemetry data for a vehicle the player is in, so these values can be monitored and passes to an interpreter for the motions of our machine.

What Went Well

Introduction
Grand Theft Auto V
The Dataset
Neural Network
Motion Simulation
Results
References

In many ways, our project was successful. Both the separate image and image highlighting methods achieved reasonable accuracies on our dataset, and performed similarly when tested in the game. The motion simulation plugin works as intended, the training process has been simplified using a generator, and the AI demonstrated behavior that showed it was recognizing both background environments and vehicles. The model excelled particularly in recognizing turns and backing away from vehicles without the assistance of our escape algorithm. However, our model is far from a passable driving AI.

What did not Go Well

When humans drive a car, each action we take is part of a planned sequence of actions. For example, if we slow down while approaching a turn, it is for the purpose of meeting a prerequisite speed before actually turning. Our model learns entirely based on imitation, so it is incapable of making these complex plans. This causes an oversimplification of certain necessary tasks. One example of this was observed when the AI approached a turn during training. It began to slow down to take the turn, but it was slowing down too quickly. Since it was still recognizing an environment where it should be holding the S key, it slowed all the way to a stop and began to go backwards. After backing up for a few seconds, the model again recognized an area where it should go forwards and began to repeat this cycle.

What did not Go Well (1)

In addition to the model's limited capacity to encode complex behaviors, another factor that makes our AI prone to this issue is that key features that tell the AI to perform an action can be recognized from any distance, with the only difference between the correct and incorrect time to take an action being a change in apparent size. As a example, we may know to brake when we see a stop sign, but our complex understanding of how to drive informs us of exactly how far away we should begin to brake. A stop sign is easily recognizable from a great distance, so how should we correct our model's tendency to predict actions too early?

Further Research

Introduction
Grand Theft Auto V
The Dataset
Neural Network
Motion Simulation
Results
References

To address the issue with recognizing distance in our images, one could also pass as an input to our network a heatmap corresponding to the distance to objects in each image. Many algorithms to do this exist, and they could easily be implemented during the preprocessing phase of our training generator.

Further Research (1)

Introduction
Grand Theft Auto V
The Dataset
Neural Network
Motion Simulation
Results
References

Alternatively, the option of utilizing a Reinforcement Learning Algorithm has shown largely superior results across the literature for autonomous locomotion. This could help our model encode more complex behaviors in response to stimuli. While setting up a learning environment in the game may be more challenging than taking screenshots, it may be the necessary step to take to achieve a better AI.

Further Research (2)

Introduction
Grand Theft Auto V
The Dataset
Neural Network
Motion Simulation
Results
References

We welcome contributions and suggestions, so please check out our project GitHub page at (https://github.com/asmalex/GTA_V_SelfDrivingCar) and our labs AI and Machine Learning division at (http://www.waynenterprises.com/ai-ml)

Sample Video

A video of our project live test can be viewed here (`https://youtu.be/Y2iQMnG_Kvc`). We first see the model reacting to a turn by backing up. I then manually maneuver it back on course to the bend. Last, the model reacts to the turn again by moving backwards.

References

- G: https://store.steampowered.com/agecheck/app/271590/
- WE: https://www.nirsoft.net/utils/winexp.html
- A: https://github.com/aidenyg/GTAV-Self-driving-car
- TY: https://www.kaggle.com/lavanyashukla01/yolov3-lyft-dataset/?select=model.h5
- Y: https://pjreddie.com/darknet/yolo/
- O: http://openiv.com/