Machine Learning Nanodegree

Capstone Proposal

Isaac Chan

March 14th, 2018

Combining Computer Vision and Supervised Classification to Label Standardized Video Game Objects

Domain Background

When I was a part of chess club in elementary school, I'd learned about chess's many legends, but latched onto former World Chess Champion Gary Kasparov. I learned of Kasparov's victorious but hard-fought battle against Deep Blue, and then his eventual and inevitable defeat by its evolved form. Even as an 8-year-old, I understood the significance of his loss. Artificial intelligence had defeated one of humanity's greatest intellectual champions, and it was only going to get smarter.

Since then, I've continued to play a variety of videogames, from Halo to Age of Empires. However, I was always disappointed by each game's AI and felt like none of them came close to human skill. When AlphaGo defeated Lee Sedol in 2016, it was touted as another breakthrough in AI. Then in 2017, OpenAI designed a Dota 2 bot that could defeat every pro player it faced. I felt inspired to use my machine learning tool kit and passion for video games to create an undefeatable opponent myself.

The game I've chosen is League of Legends, which is similar to Dota 2. My goal is to use computer vision and unsupervised clustering techniques to build the foundation of a videogame bot by teaching the bot how to recognize the objects it will interact with.

Problem Statement

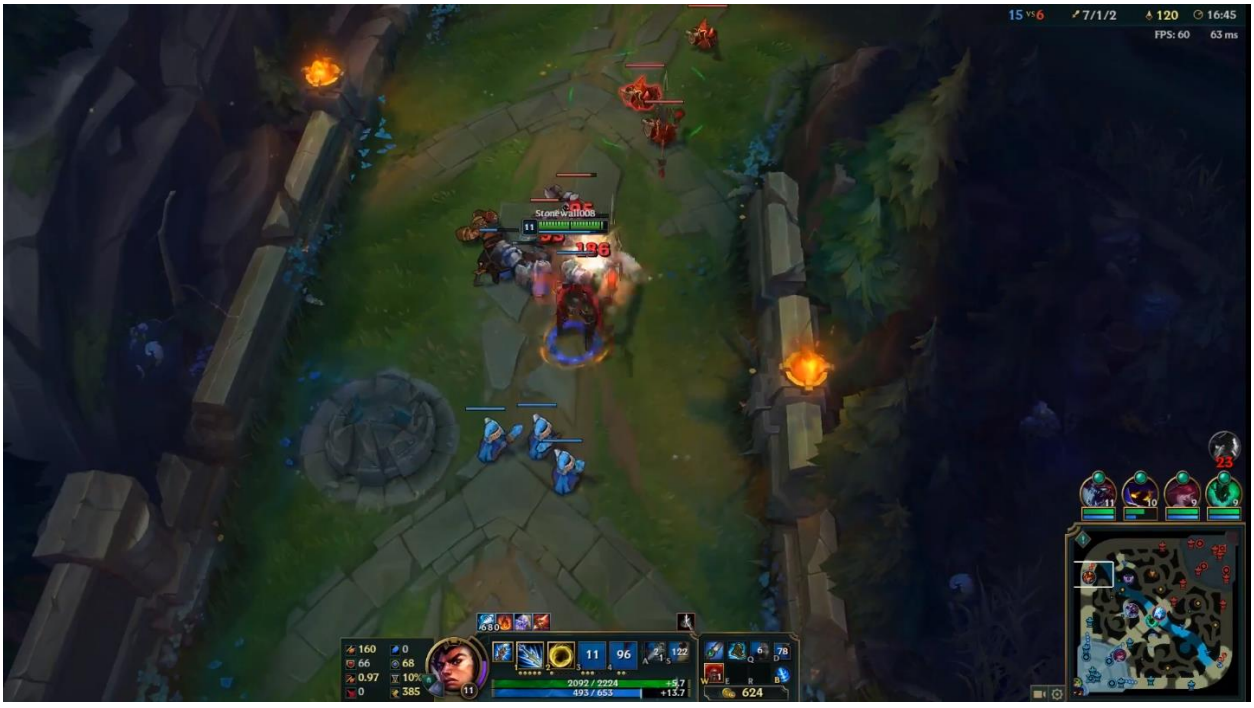
Computers no longer have any difficulty defeating humans in board games and "old-school" videogames. However, board games and older video games often have very limited inputs and states (up, down, left right, move, stay), and can be solved using brute force alone.

Winning in modern videogames is often a matter of milliseconds and can have as many inputs as there are pixels on the screen. For this reason, in only a single simplified version of a modern videogame has AI been successful. Creating an AI that can generalize to modern rather than Atari games would be one step closer to creating true artificial intelligence. The first step would be to recognize the objects within that environment at an interactable rate.

The specific game I've chosen as my domain is League of Legends, a game similar to Dota 2. The objects I am trying to detect and label are called 'minions'. They are non-player-characters that players kill to earn gold and experience to become stronger. There are 2 different colors and 3 unique types of minions.

Datasets and Inputs

The datasets I plan to use will be based off my own screen captures and video captures of the video game *League of Legends* using the PIL library. Capture size will be 1920x1080, but after a layer mask is applied to remove extraneous pixels the final image size will be closer to 1024x768. The initial minion recognition and labeling algorithm will be trained and tested on static images, and then altered for live screen capture capabilities, which are exactly what a human player would see. Sometimes there may be 0 minions on screen, other times there might be 12+. While live, the framerate (screen captures per second) could range from 5-12 fps depending on performance demands.



Sample Screen Capture

Sample Training Image

SIFT features will then be extracted from the post-masked images using the opencv3 library. Features consist of two parts: key points, which are the physical pixel location on the screen represented using XY-coordinates, and the descriptors, which describe the structure and gradients around the key point. Besides the key points, the only other information that will be used are RGB pixel values from the screen capture to determine the test object's color.

Local outlier detection will then be used to remove features based on pixel location that do not appear to be part of an unidentified object cluster. The remaining 400-500 features will then be clustered using k-means through sklearn based on the key point pixel location to create potential objects. Creating an ROI around each cluster's centroid, the test object will be passed through an HSV filter to filter out non-red or blue pixels. The remaining pixel's RGB values will then be used for labeling which color the minion is.

Solution Statement

The solution to solving the problem of video game object classification without manually creating and labeling datasets of thousands of images is to exploit the uniformity and standardization of these objects. While there are tens of thousands of

different models of cars, along with different viewing angles, conditions, and colors, each video game object will always be the same save for variations in viewing angle and object action. Template matching using SIFT works only in situations with even less variation than video game environments, whereas object detection works in many different situations but requires thousands of times more data. The solution is to combine computer vision techniques with unsupervised learning algorithm's that can extract maximum utility from the data.

Benchmark Model

Object detection given a large enough training set is a tried and true accurate and generalizable method. I have no doubt that with tens of thousands of training images for each object we wish to detect, an accuracy of 99% would not be difficult. Even accurately classifying these objects in real time with over 16 frames per second is possible thanks to algorithms like YOLO9000. On the other side, pursuing a computer vision-based model that uses template matching to label clusters would have similar to random accuracy, but take a fraction of the time to setup and have higher frames per second thanks to its low processing demands. I will be comparing my model's accuracy against the template matching model (better than random), but comparing my models training and fps against estimated object detection model accuracy and training times.

Evaluation Metrics

The two-main metrics I'll be judging my algorithm on will be accuracy and speed. Accuracy is simple. If there is a minion on the screen, detect it. For each minion, accurately label its color. For the scope of this project, I will not be determining which type of minion the unidentified cluster is, only it's color.

Speed is measured in how many frames per second the image capture device can perform while simultaneously performing the necessary calculations.

To accurately measure how well k-means clustered the 450-500 locally distinct key points into minions, each cluster's silhouette score will be calculated. In order to achieve the best score, hyperparameters such as n_neighbors can be tweaked.

Project Design

To find the baseline accuracy of a pure template matching classifier, first create a set of training and testing images that both include one image of each minion type. There

should not be similar image in both sets. Then, start extracting features from the 6 test images and the 6 training images using ORB, an upgraded version of SIFT, through the opencv library. However, without utilizing some machine learning techniques, adapting this to real time labeling would be impossible.

For each test image's key points, begin brute force matching against each other training image's key points and return the score. The best scoring set of training key points and its corresponding minion type is what the testing image would be labeled by.

Modern object classifiers have a tried and true accuracy and speed that does not need to be replicated in this scenario. With a powerful enough GPU and enough training images, the YOLO9000 real-time object detection system reports accuracies of 99% on thousands of different objects. The goal is to create a classifier that doesn't require exorbitant labeling and training data.

The solution is to combine ORB and unsupervised learning tricks to take advantage of how standardized these objects are. The first task is to identify, given a screen capture, how many minions are in the environment.

Running ORB on the screen capture, 450-500 key points will be found. Some will undoubtedly be outliers that don't belong to a minion. Using the pixel coordinates of all the key points, local outlier detection through sklearn will remove features that likely do not belong to minion and are a feature of the landscape or a projectile. With the pixel coordinates of the remaining key points, use k-means through sklearn to cluster key points into minion clusters.

For each cluster, rasterize an ROI around the centroid and convert the RGB values to HSV. Using a mask created through cv2, filter out pixels with low saturation values, leaving only the saturated blue and red of the minion. Running k-means again on the remaining pixel's RGB values, determine if the most prevalent color within the ROI is red or blue and label as such.