Alexander Pinkerton

PokemonGO Project

11/30/2016

Pokemon Go Stat-Recognition

In order to retrieve stats from screenshots of Pokémon Go I decided to use two different approaches which depended on the type of information that was to be extracted. When attempting to extract the Pokémon ID from the images, I decided it would be best to train use a Convolutional Neural Network to perform the classification of Pokémon IDs. Unfortunately, I was unable to fine-tune and existing model due to the read-only format of the network architecture. Since our training data was quite limited and that I was unable to edit the network architecture, I decided to use the bag-of-words approach. In order to train the bag of words image classifier, I cropped the input images to only contain the Pokémon regions and scaled them all to the same size. Once the vocabulary was created and the classifier was trained, I was able to receive about 65 percent accuracy on the validation set. Although I was decently pleased with 65 percent accuracy, I wanted to try and implement a different algorithm. I decided to implement the eigenfaces algorithm. In order to implement this algorithm, I first converted the cropped training set into columns vectors. These column vectors also had the training set mean subtracted from them in order to center the data around the origin. This ensures that all of the data being learned pertains only to important features that define each image. Once this training set was completed, I performed PCA to obtain the coefficient columns for each image to use as training data for a classifier. I used these coefficients to train a SVM using the labels. As a sanity check, I also reconstructed the eigenvectors or principle images to see if everything was working and they were upright and had progressively finer variation as I stepped through them which told me that PCA had done its job. Although all of this seemed to work, when projecting a new image into the eigen space and using the SVM to classify the image as Pokémon, I had trouble with it classifying everything as 133 “Evee”. I decided to inspect the dataset and found that Pokémon 133 – “Evee” had the most images in the training set. I feel as though this was the main reason my classifier was not working correctly but I am still unsure as I feel as though I implemented the Eigenfaces algorithm correctly.

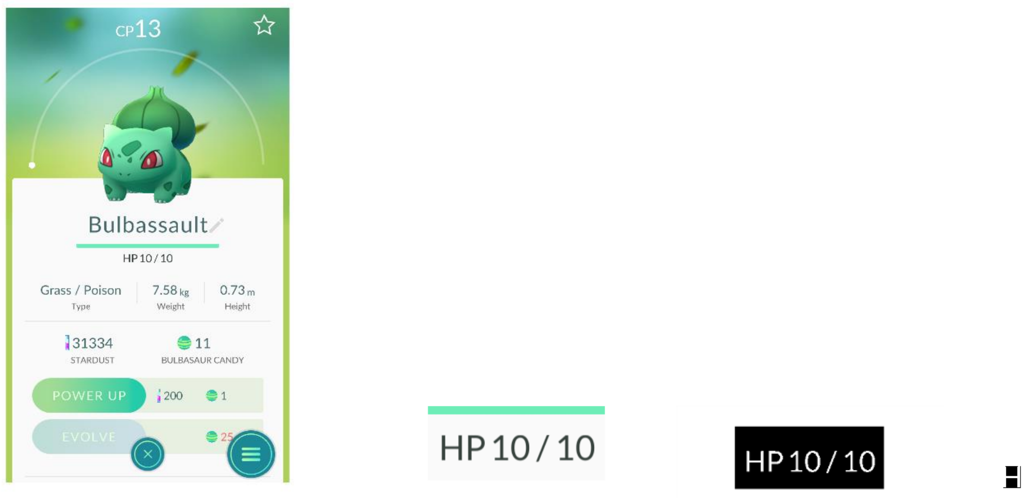
Extracting the text from the screenshots was a different problem entirely. For this task, I initially focused on how to locate the text in the image. To accomplish this, I cropped the input images to isolate the information to be extracted by using percentiles of the screen. This cropping was not perfect for every picture due to certain aspect ratios amongst other things, however, this seemed like the best option. With the cropped regions, I multiplied the contrast and then used a threshold to binarize the crop. Once the cropped region was binary, I used the MATLAB function ‘regionprops’ to locate candidate regions for bounding boxes of objects. Since I had already isolated the data in question, I was able to easily extract each of the characters.

Since this process was consistent, I extracted every recognized character in all of the training data and saved it into a folder. I used this data to create a custom training set consisting of characters which my algorithm was locating. I used around 30 examples for each character. Once the dataset had been created, I needed to train a classifier which would be able to tell me what character was most likely when having one as input. I decided to use a multiclass SVM as my classifier and trained it with HOG features of the dataset images. After training the SVM, it was able to recognize text at around 65% accuracy on the validation set.

The extraction of the Pokémon level and the circle center of the CP bar were quite similar. To find the level of the Pokémon, I thresholded the image and used the circular Hough transform to locate the circle of the dot. To find the circle center, I used the dimensions of the image to estimate the position which is quite accurate but could certainly be improved.

The combination of the two primary models allows me to reliably extract the stats off of a Pokémon go screenshot, however, this algorithm can certainly be improved. One of the major ways I could improve my algorithm is by creating a better strategy of finding the regions in the image with the information which needs to be extracted. Since I am simply cropping out certain portions of the image, it is very sensitive to translation and different aspect ratios.

Pipeline for Text Classification

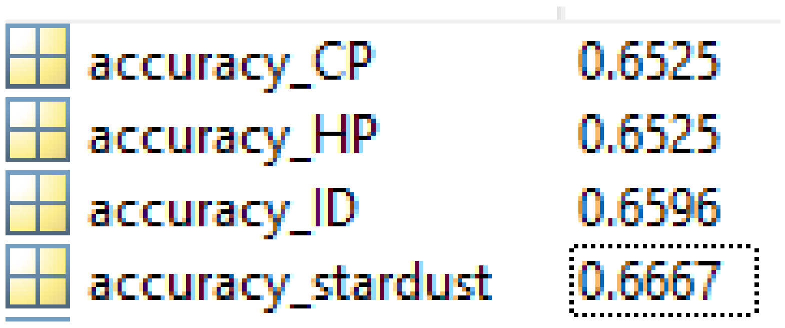


- Concatenate all successfully recognized characters in image.
- Sanitize output by removing unwanted characters

Pipeline for Pokemon ID Classification

1. Create Vocabulary
    a. Extract HOG features for every image
    b. Cluster all features to form vocabulary
2. Train Classifier
    a. Extract HOG features for every image
    b. Train multi-class SVM to classify HOG features to label
3. Test on Image
    a. Extract HOG features for character image
    b. Pass to classifier and obtain label as result

Results on Validation Set



accuracy_CP	0.6525
accuracy_HP	0.6525
accuracy_ID	0.6596
accuracy_stardust	0.6667