Pokemon TCG Scanner & Price Checker

The functionality of this prototype app is to allow a user to provide input in the form of a still image of a Pokemon TCG card. A still image may be considered as a photograph or a frame of a camera feed input. The still image will then undergo processing via machine learning which will then produce a string. This string is then used as a query term to compare to a database of Pokemon TCG card data. Finally, the user will receive a high resolution image of their inputted still image of a Pokemon TCG card. Three major components were required as part of this project.

1. A database
2. Machine Learning models
3. A medium to visualize and display the queried information

The Database

To establish a database, we performed a pull on the Pokemon TCG API to obtain large quantities of data. For the purpose of a prototyping, we opted to perform a pull on only the Sword and Shield Series of cards, where within a series are sets of cards released every quarter. Once we pulled the data, we recognized that for our intended use, we could remove some of the data that was provided in order to clean up and reduce how much data we needed to store. In the end, we went from 24 columns of data to 9 columns of data. The database would then be usable to query a specific entry or entries in order to further narrow down what information can be used to present to the user. For prototyping, the database is also only setup to contain one entry of each Sword and Shield Series card however one of the columns of data is a generated Datetime field. In further development, the database could hold more than one entry of each Sword and Shield Series card and be scripted to pull regularly from the Pokemon TCG API to create a log of prices by date. Further expansion would include adding additional vendor sourcing and other API pulls. The Pokemon TCG API specifically uses pricing information from TCGPlayer.com. Due to the nature of legal play rotation, the database could also be scripted to delete entries beyond a certain date threshold.

Machine Learning

Application of machine learning in this project encountered resource capability difficulties. The intention for machine learning use in this project was to train a deep neural network for object detection for a Pokemon TCG card within a still image. The technology opted for proof of concept for this was Tensorflow, an open source platform for machine learning. Upon being able to detect the card, the image would then be subjected to processing through cropping the image around the detected object. The intention from there is to further zooming in to a certain pixel height/width size while maintaining aspect ratio. A correctional rotation can then be applied should the card in the still frame not be dead on. The image may be further cropped at this point and zoomed in again. These three steps are within the capabilities of OpenCV, the technology opted to try for this project and a library of programming functions aimed towards computer vision. This step may or may not be necessary depending on the competency of optical character recognition (OCR), which is another form of machine learning. However, before OCR is done, a region of interest could be trained on the newly cropped and zoomed image using Tensorflow as a platform and OpenCV backing it up since OpenCV supports model execution for machine learning. The region of interest we were interested in would be the lower 15% of the card in modern Pokemon TCG cards. Specifically, the lower 15% of a card contains the collector number and set symbol used to identify it. The set symbol could be compared using OpenCV’s image differences and scored using a reference of trained set symbol images. Alternatively, the set symbol could be detected using object detection models through Tensorflow. Similarly, using Tensorflow and OpenCV, it would be possible to object and pattern detect for the collector number, as in modern Pokemon TCG cards, the set symbol is located next to the collector number. While vintage cards may not follow this same pattern, the possibility to expand the machine learning aspects to include the identification of collector numbers and set symbols would not be beyond the realm of possibility. Finally, once the collector number location has been identified, OCR is required to obtain a queryable string for the database. Tesseract, an open source OCR engine and application of machine learning, was experimented with for the OCR portion. The scope of the layered machine learning described above is complex but within reason due to being able to break the process down into smaller steps. However, the difficulty encountered during the execution of the project was partially resource based. Even with the included use of the GPU, my (Ian’s) personal computer ran out of memory while attempting to train 50 sample images resulting in an incomplete model that could not be used for evaluation. There were also errors involving windows access denial. Unfortunately due to these circumstances, the machine learning aspect of the project would have to settle for something more readily available as opposed to attempting to custom train an object (re: Pokemon TCG card) detection model. Instead, Google Vision is already able to perform the OCR across a noisier image. We were able to feed the cloud vision API an image where the machine learned OCR was capable of recognizing more and more accurately, due to more advanced training of Tesseract in Google’s Cloud Vision. With more being read, it was then necessary to use regular expressions to filter down to the collector number from the expanded string that the OCR provided. The collector number was extracted and then able to be passed onto the database for querying.

Displaying the Data & Visualization

Once the database has been queried, entries are narrowed down and can be assigned as a dictionary(s) and used in visualization methods. For our purposes, we opted for Flask implementation which required HTML, Javascript, CSS, and Jinja2. Further to that, we used SQLAlchemy, Boto3, to connect our Flask application to the database and storage to provide a URI to the Google Cloud Vision API. The low level proof of concept visualization is to present the high resolution image of the card being scanned/taken the still image of as well as have drop down selection dynamically built using Javascript to display what the prices of the cards are. Future development could include a graph plotting the prices as a graph over time assuming the database were further developed in the mentioned methods in its description. The visualization via HTML would be built with mobile in mind. This is because the app’s common use case scenario would be to be combined with an on-machine camera like a mobile device.