**Pokemon Trading Card database and identification software using object detection, image classification and OCR**

**I. Introduction**

Trading cards are an enormous industry with a market size of $956million (in 2021), with forecasted growth to reach $1.64billion by 2031. As such, large numbers of tools are needed for collectors and trading card fans to document and record their collections.

There currently exists a large number of trading card databases and apps, such as TCGCollector, Collectr, Digital Card Manager, etc. Additionally, card scanners currently exist for other systems such as Delver Lens for Magic the Gathering or Ludex for Sports Cards. There is however, no all-in-one solution app for Pokemon TCG.

**II. Project Objective**

The project I am proposing is to create a web app which is able to **scan** a trading card, **identify** that card in real time, **organize** it into a collection, and allow the user to **track** their collection. This project will involve several disciplines, some outside the scope of this bootcamp, but will involve the use of **pretrained CNNs, training my own CNNs**, using **OCR**, all in real-time to scan and identify the card. Upon identifying the card, the app will be able to store to user account a complete collection by leveraging **SQL** and **database management techniques**. Additional tools and functions may be added to the app (time permitting) which may include functions like price predictive models.

**## III. Data Description**

The datasets for individual Pokemon TCG cards are available online either through open-source API or through associated GitHub. For new sets which may not yet be added to the database, data-scraping may be necessary.

**Source(s) of the data**

* English language cards Open-source API
  + <https://pokemontcg.io/>
* English language cards GitHub
  + <https://github.com/PokemonTCG/pokemon-tcg-data?tab=readme-ov-file>

**Optional**

* Japanese language cards
  + <https://jpn-cards-site.readthedocs.io/en/latest/api_docs/pokemon/v2/v2_api/>
* European language cards
  + <https://github.com/kirbyUK/ptcgo-data>

API database is in \*.json format with the following Classes and Properties.

**Classes**

* Card
* Set
* Type
* Supertype
* Subtype
* Rarity

**Set Properties**

* Id
* Images
* Legalities
* Name
* printedTotal
* ptcgoCode
* releaseDate
* series
* total
* updatedAt

**Card Properties**

* abilities
* artist
* ancientTrait
* attacks
* convertedRetreatCost
* evolvesFrom
* flavorText
* hp
* id
* images
* legalities
* regulationMark
* name
* nationalPokedexNumbers
* number
* rarity
* resistances
* retreatCost
* rules
* set
* subtypes
* supertype
* tcgplayer
* types
* weaknesses

**## IV. Methodology**

To understand the methodology, one needs to understand the specifics of the Pokemon trading card.

There are over 20,000 English Pokemon cards divided into approximately 100 Sets, with each Set containing at least 100 cards (see data description). Each card has a symbol, indicated in either the lower left corner of the card, or the middle right of the card indicating which “Set” the card belongs.



Additionally, the Pokemon’s name is found in the top left in text. However, the same Pokemon may contain multiple different cards within the same set, and this can be differentiated either by abilities, artists, image, or collector number. The collector number is the most exact readout, however is the smallest in font and likely the hardest to read using OCR (particularly if camera resolution becomes an issue). It may be necessary to utilize some combination of the above to further identify the correct individual card.

The overall project should be considered in three parts, each of which should occur roughly concurrently.

1. Image Processing in real time
   1. Must be developed, tested, and function in real time.
   2. Must be able to determine one specific card from a larger set
      1. This set may be all Pokemon cards in existence, all English Pokemon cards in existence, or all Pokemon cards from a specific series (grouping of Sets)
   3. Must be able to identify a card being held in front of the camera
      1. **Mask R-CNN** or other (**MobileNet**, **YOLOV5**, etc.)
   4. Must be able to use a trained CNN to determine the “Set” to which the card belongs
      1. **Train a CNN** using dictionary “images” and the key “symbol”
   5. Must be able to use OCR to narrow the field (either using creature name or collector number)
      1. **Pytesseract** or other OCR
   6. May be processed locally, or better processed via server
      1. Microservice may be necessary to handle image processing/detection and OCR
   7. Should be integrated into the app using the phone’s camera.
   8. Minimally, Image Processing should resolve with sufficient information to identify one member of the database.
2. A database
   1. Once image processing has completed, sufficient information should be collected to narrow the choice to a single card in the database.
      1. If multiple cards are identified as possible candidates within a specific margin of error, a popup to allow manual selection of the correct card.
   2. Must have a database of all possible cards as defined in 1.b.i.
   3. Must have data security and privacy measures (encryption and secure data storage)
   4. Must have CRUD operations for adding, viewing, and deleting cards from personal “collection”
   5. May be locally hosted or integrated in cloud-based database (Firebase, PostgreSQL, MongoDB)
3. A website or app
   1. Must have user authentication (signup, login, logout, password reset)
   2. Must have data security and privacy measures
   3. Must have basic account management
   4. Deployment may require sideloading into a smartphone and/or backend API on a cloud provider.

**## V. Expected Deliverables**

At minimum I would like to deploy a website or phone app, that can take real-time input (webcam/cellphone camera) or static images (upload an image), identify the card in question, and add one to the database. The backend would include a series of CNNs (pretrained image classifier, personally trained image classifier, and pretrained OCR) to identify a specific member of the database.

**## VI. Timeline and Tasks**

A diagram with a green corner

Description automatically generated with medium confidence

A number of these tasks will be undertaken concurrently. Outside resources may be used to assist with Website/App Deployment in the early stages.

**## VII. Potential Challenges**

This project is big, bulky and presents a large number of challenges. To enumerate and address some of the most obvious ones:

1. Amount of unrelated work
   1. Some of the work can be dropped in favor of a less user-friendly deployment
   2. The timeline presented will provide me time to request assistance and pivot as needed if the app deployment strategy is unable to be met
   3. Minimally, the image detection portion (real-time or static) and the database portion should be intact and usable.
2. Real time image detection
   1. Some of the portions the project will be requesting will be extremely small portions of the card (set symbol, collector number). The resolution of the camera and therefore of the image will be heavily dependent on the model of phone, the quality of hardware, and the closeness of the card to the camera.
      1. A digital outline can be shown in the frame to indicate where the card should be placed to maximize resolution
      2. If this is still insufficient, the program will request that the user zoom in on the areas that lack sufficient resolution (set symbol, collector number)
   2. Should the real-time image detection not work, consideration can be given to modification of the project into static image categorization.
      1. Users would upload an image of their card and the CNN ensemble will identify the card.
      2. Upon identification of the card, connection via API to card pricing websites such as TCGPlayer can be created and charts can be displayed to show historic prices and predict future prices.
         1. This may be a further functionality of the app/website regardless.