Pokédex - A Machine Learning Project

Team Tyrannosaurus

Dylan Painter, Shirley Qi, AJ Reiter, Vicky Xie

CPEN 291

Table of Contents

Description

Inspired by the popular anime and video game franchise, Pokémon, this project replicates the main gadget used by all Pokémon trainers, the Pokédex. In the game and show, the Pokédex acts as an autonomous encyclopedia that gives a brief description of a Pokémon when encountered. Our Pokédex runs as a dynamic website application coded in Flask that takes, as an input, a JPEG image and feeds it into our trained ResNet50 model. The model will then predict which Pokémon is in the image and output information such as that Pokémon's name, Pokédex number, type, abilities, evolution, forms, and much more. The webpage will also display the confidence of its prediction as a percentage, along with the next four highest prediction accuracies at the bottom of the page in case the predicted Pokémon is not what you are looking for.

The model is currently trained to classify the first 75 Pokémon registered in the national Pokédex. The images that the model is able to accurately classify are those that contain Pokémon similar in appearance to the original design, such as an image from the anime, or a real-life plushie. Images that don't contain a clear image of a Pokémon are predicted less accurately, as shown below.

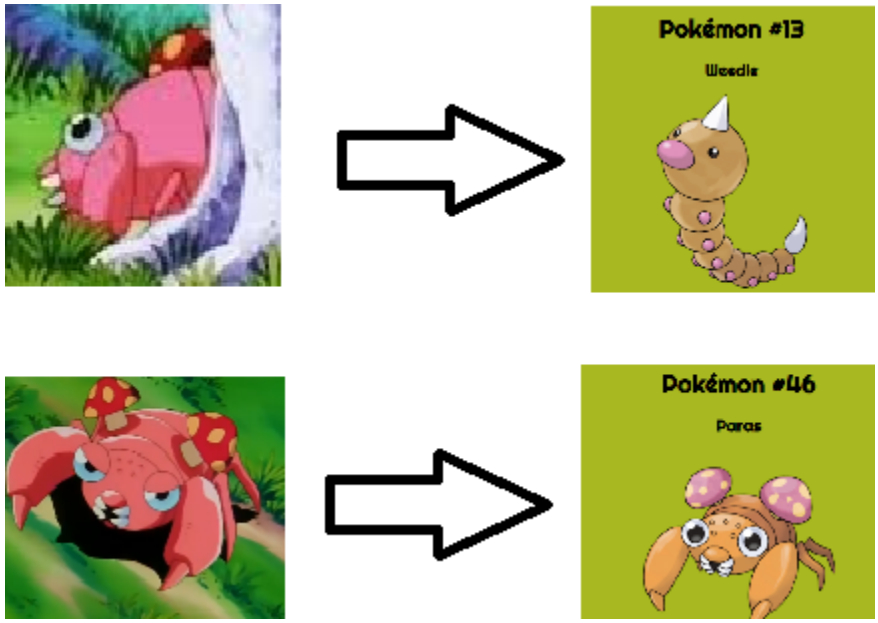

Figure 1: Bad Vs. Good Prediction

Walkthrough of Pokédex

To use the Pokédex, a JPG image file is needed. As an example, we will take a screenshot of the Squirtle in the YouTube video shown in figure 2. An important note to consider is that the Pokédex will only take JPG files, so a JPG converter may be needed if this type of file is not available.

Figure 2: Getting an Image

Once a desired image has been obtained, download the website.ipynb notebook from the GitHub repository and open it in Google Colab. In the notebook, there will be 3 commands. In order for the website to launch, the first two commands must run first (labelled [1] and [2] in figure 3). After, run the third command starting with “!python” to start the website script. As the script is running, some text will appear below the command. Wait until the line shown by the red arrow in figure 3 appears as this is the link used to launch the Pokédex.

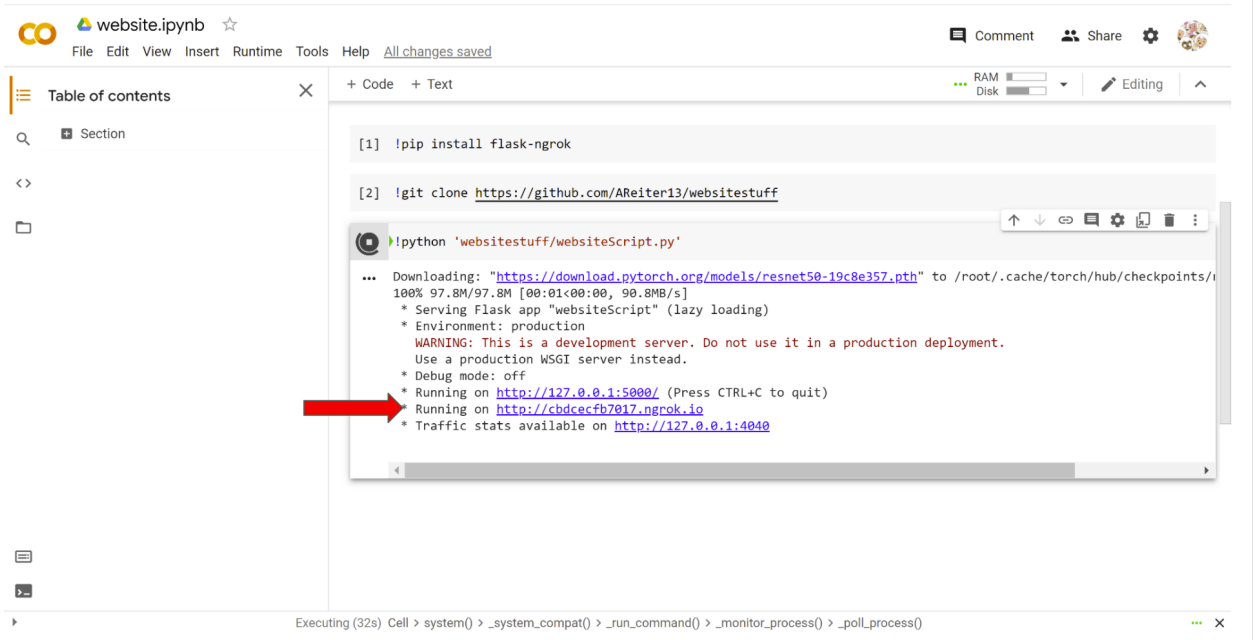

Figure 3: Launching the Website

After clicking the link, a new tab should appear that looks like figure 4.

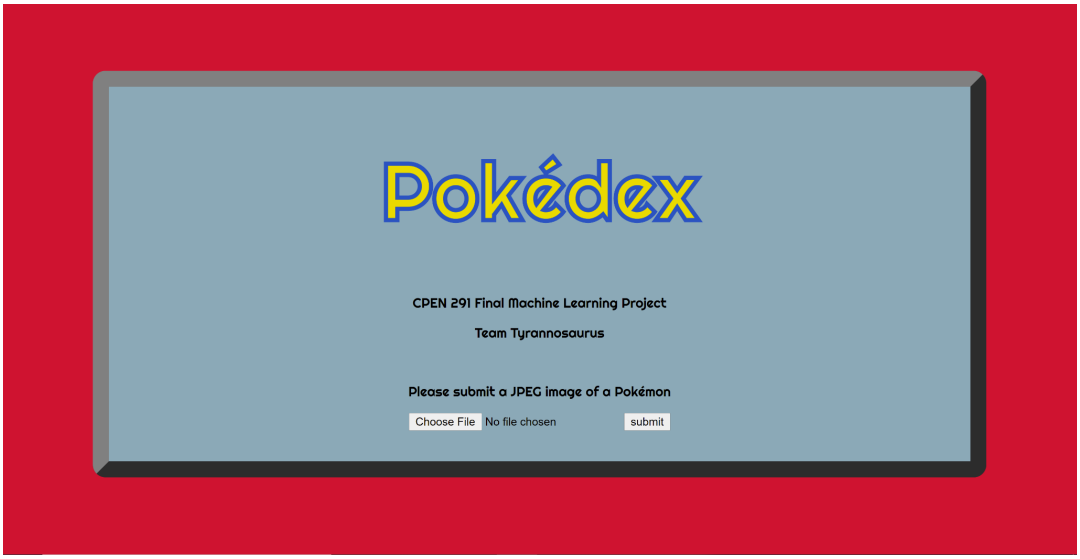

Figure 4: Pokédex Home Page

Next, click the “Choose File” button and select the image to predict. The file upload will be successful if “No file chosen” in figure 5 is replaced by the name of your image.

Figure 5: Attaching an Image File

When ready, click “submit” to send the image to the model and give a prediction. Because we are using an image of Squirtle, we should expect the webpage with Squirtle to load as seen in figure 6.

NOTE: If you select an image file that is not JPG, pressing “submit” will not do anything and the file name beside “Choose File” will be replaced with “No file chosen”. Please choose another image that is in JPG format or use a JPG converter.

Figure 6: Squirtle Webpage

Your image has now been predicted! You can now browse all the information the page has to offer. In the event that this Pokémon is not what you are looking for, on the bottom of the page, there will be a text box with the top 5 accuracies the model predicted. What you are looking for may be in that list.

Modules

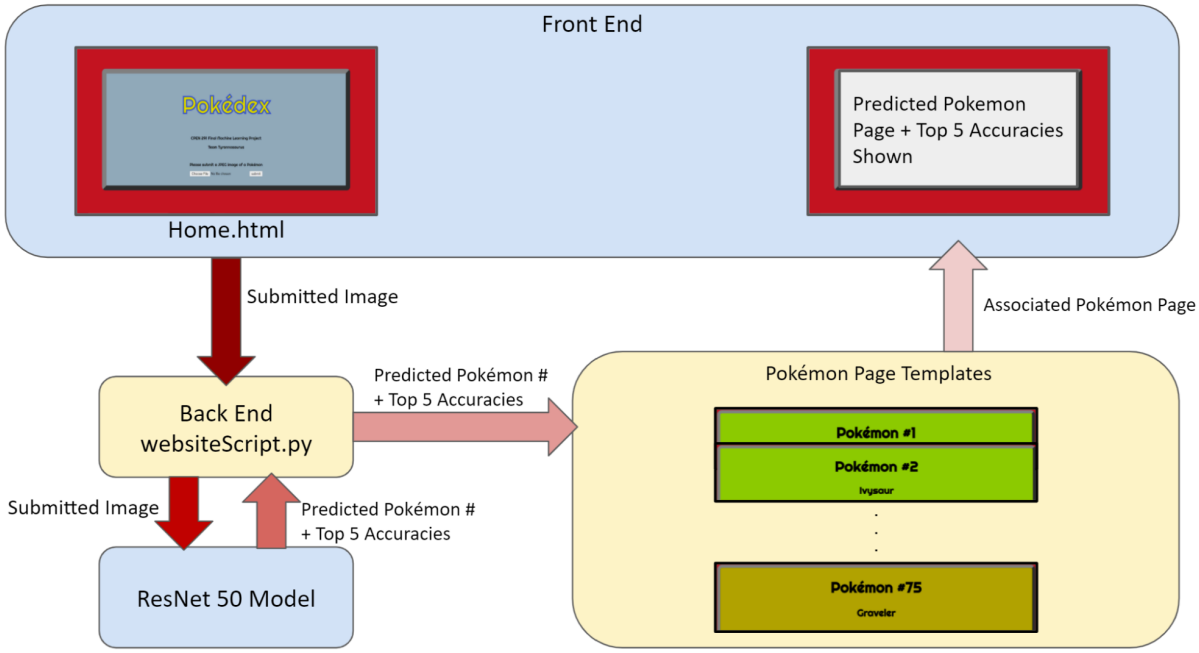

Figure 7: Component Diagram

Web Front End

The front end consists of a home page, and the page of the predicted Pokémon.

The home page (Home.html) is used to introduce the information of the project and accept the files for prediction. Its background colors are red to match the color of the Pokédex. The name of the project, and the name of our group is displayed in Righteous font type. These headings are coded using simple HTML commands. Below these headings is the input channel. This is where the desired JPG image is submitted. This is implemented using a simple HTML action form using the POST method to send the submitted file into the website's backend script. The backend is built to only accept JPG images, so if any non-JPG image is submitted, the backend reloads Home.html.

The other component of the front end is the web page of the predicted Pokémon. This webpage is selected from 75 possible web pages, one for each Pokémon. Each web page that could be displayed is also coded in HTML. These web pages consist of many classes used to implement various header, bars, images, and more. All the information is collected from Bulbapedia, and the background colors are selected based off of the Pokémon's primary type. Each webpage takes, as an input, the top 5 prediction accuracies, along with their associated Pokémon ID number. These are used dynamically to display at the bottom of the page which 5 Pokémon the model thinks the image most likely contains.

Web Back End

The back end consists of websiteScript.py, a trained ResNet50 image classifier model, and a folder of HTML web pages.

The model we chose for prediction is ResNet50. ResNet50 is a variant of the ResNet model with 48 Convolution layers, one Max Pool, and one Average Pool layer. We replaced the fully convolutional layer with a linear layer that has 75 output channels. The ResNet50 model was trained on over 5000 images from 75 categories of Pokémon. When an image is input into the classifier, it creates a value for each of the 75 Pokémon representing how likely it is that the image contains that specific Pokémon. We take the highest value as the predicted Pokémon, along with the next four highest values as potential other Pokémon. The advantage of ResNet50 is that its batch normalization can increase the network's performance by adjusting the input layer.

The Flask application, websiteScript.py, is coded in Python and aims to build bridges between other modules. This is the file that must be run to launch and run the website. This script first loads the state dict of our trained ResNet50 model stored in the same directory. Then, it creates a dictionary of Pokémon names and ID's implemented using a list. Next, it launches the website and creates links. Whenever the request method is GET, such as when the website is launched, the home page is displayed. Whenever the submit button is pressed, the request method is changed to POST, and the script then takes this image, checks that it is a JPG file, and feeds it into the image classifier model. If it is not a JPG file, or the file input is empty, the script reloads the home page, and the request method is set back to GET. When the model receives an image, it returns a prediction, along with the top five values as described above. When the script receives the prediction, it then uses 'if, else' statements to select the correct webpage from the folder of Pokémon web page templates. Finally, it passes the top 5 values as percentages, along with their corresponding name dictionary entries into the web page and redirects to that page.

Finally, the last element of the backend is the folder of Pokémon web page templates. As described in the previous paragraphs, there are 75 templates in this folder. The Flask script uses the prediction to select a file from this folder to display on the front end. The contents of these files are described above in the front end.

Partner Contributions

Dylan Painter

- Collections pictures for pokemon #1-18
- Tried multiple ways to get the pokemon images into Google Colaboratory, but wasn't used as an easier way was found
- Tested the model with different weight decay, dropout, etc
- Created HTML templates for pokemon 1-18
- Made rough draft diagram + Description

- Planned about half of the video
- Recorded part of the video

Shirley Qi
- Collections pictures for pokemon #19-37
- Created a public github with the pokemon images so it could be used to download the images into Google Colaboratory
- Wrote code to load data from github and to initialize the dataset with said data
- Design HTML templates for the home page of the website
- Created HTML templates for pokemon #19-37
- Added code to show the models top 5 predicted pokemon to the pokemon pages
- Recorded part of video
- Completed walkthrough of report

AJ Reiter
- Collected pictures for pokemon #38-56
- Figured out how to save and load the model and wrote code for it
- Tested different image transformations on training/testing results for the model
- Created HTML templates for pokemon #38-56
- Wrote websiteScript.py (excluding the code to find and display the top 5 prediction accuracies.)
- Partially wrote and revised the modules section
- Revised the description and created figure 1.
- Recorded the 3rd part of the video (how the website works)

Vicky Xie
- Collections pictures for pokemon #57-75
- Completed part of the code for training the model
- Created HTML templates for pokemon #57-75
- Designed the HTML templates for the pokemon (everyone else took that and updated with data specific for each pokemon)
- Recorded part of video
- Completed part of report