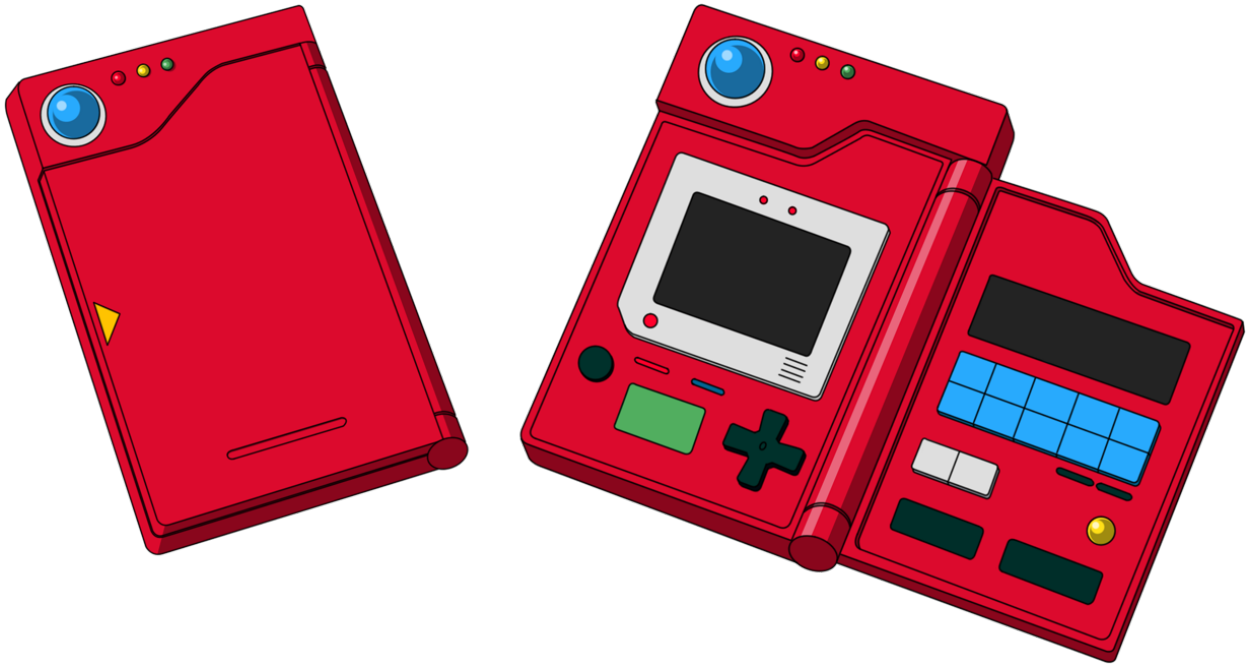
Proposal

ǟ I wanna build a Pokédex, like no one ever has! ǟ
I would like to build a computer vision program that correctly identifies Pokémon from an image that is fed into the program.



Domain Background

(approx. 1-2 paragraphs)
I have been able to find a few other similar projects that provided some inspiration for me, but none that actually do exactly what I intend to do here. There are some that fill in the colour of a Pokémon based on its type (https://github.com/hemagso/Neuralmon), others that predict type of Pokémon (ie Grass, Fire, Water etc.) from its colour (https://github.com/juandes/PokemonTypesDeepLearning) and some that look at the pixel art from a GameBoy screen and return what the Pokémon is (http://www.pyimagesearch.com/category/building-a-pokedex/). The last project I mention is indeed very close to what I would like to achieve, however I want to work with the anime depicted Pokémon, not game sprites.

Problem Statement

(approx. 1 paragraph)
The idea is to have the machine correctly identify a Pokémon image from a set of twenty as I was advised 151 would be too time consuming. The program will be fed an image and using its trained model output a response that names the Pokémon that appears in the input image.
A large amount of images for each Pokémon are needed in order to train a classifier to be able to do this. So the images will be need to be collected and stored in some way first.
The output can be measured by using an assessment method such as a confusion matrix.

Datasets and Inputs

(approx. 2-3 paragraphs)
The data set I had in mind will be based on web scraping images from the internet, sites such as Bulbepedia and Smogon contian high quality images of every Pokémon including the Pokémon in various diffferent poses and angles.
The dataset will be stored locally at first but may be uploaded to a database if needed.
All images will be saved a .png files and I will be scraping 100 images per Pokémon if possible. If this is not possible then I will impliment a work around detailed below. Here are a few images as exmaples:

Solution Statement

(approx. 1 paragraph)
Based on prior reading and Udacity support staff I have decided to train a ConvNet. The solution will use Convolutional Neural Netwroks (ConvNets or CNNs) to decipher which Pokémon is present in the image.

Benchmark Model

(approximately 1-2 paragraphs) The benchmark model will be a simple random selector of Pokémon recognition with no learning, ie. the name of the Pokémon is picked at random from the twenty different names available. This will be measurable and can be used to benchmark the learning algorithm.

Evaluation Metrics

(approx. 1-2 paragraphs)

A simple accuracy measure would be how many times a Pokémon is correctly identified in a trial. ie

Number of correctly identified Pokémon / Number of images fed into program

A Confusion Matrix is also important to understand the precision and recall of the machine learning algorithm.

Project Design

Firstly, the correct environment will be needed. To develop ConvNets I wanted to use TensorFlow, but I also want to be able to harness my GPU's capability for better processing peformance. Since the machine is a Windows based platform I had to install a new distribution of Python namely 3.5.x to allow for the installation of module *tensorflow-gpu* via pip3. Docker would have been another solution for running TensorFlow but doesn't currently support Nvidia's CUDA technology.

I will then write a webscraping program using *BeautifulSoup* and the *requests* modules to collect sample images for each of the twenty Pokémon and save them in their named folders locally.

Next I will need to import the required modules into the Python environment such as *Keras*, *sklearn*, *NumPy*, *SciPy*, *pillow* and others. Keras will be used as a simplified wrapper for *tensorflow-gpu*. This will shortly be followed by loading the image datasets required and then apply processing to make sure they are all of equal dimension. In order to make the most of the images, I will augment them via a number of random transformations, so that the model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalise. This can be performed in *Keras*

The X_train and X_test arrays are reshaped to match 2D matrices. This is because some of the *sklearn* methods that will be used later (for example RandomizedSearchCV) assume a 2D shape for the input features. The X_train and X_test 2D matrices will however be reshaped to take their original format (n_examples, height, width, n_channels) before being fed into the ConvNet.

Then I will need to pre-process the class labels for the images and then define our model architecture. At first only one fully connected layer with 20 units will be used (as there are 20 Pokémon) and a *softmax* activation function. This is akin to a linear classifier. Then more hidden layers will be added between the feature extractor and the output layer, leading to more complex decision boundaries.

This is then followed by training the model and evaluating it. This will include tuning the following hyper parameters:

number of feature maps in Convolutional Layers
dimensions of feature maps in Convolutional Layers
dimensions of pooling matrices in Pooling layers
number of hidden units in layer between the feature extractor and the output layer.

After tuning the model, the above steps will be repeated until an accuracy above the bench and above a defined threshold has been reached. Further to this a confusion matrix will be used to identify the model's performance.

Finally predictions can be made by uploading an unclassified image.
by Lee Joshi-Jones