Professor Oak: A Keras-based convolutional neural network to predict Generation 1 Pokemon from terrible MS paint drawings

The Infamous Gentlemen

June 18, 2020

Introduction

This document accompanies the 'Professor Oak' program. Professor Oak is a convolutional neural network (CNN) which has been trained on images of all Generation 1 Pokemon (i.e, original 151, Red/Blue/Yellow). This network then attempts to predict the class (i.e, Pokemon species) of unknown images supplied by the user. Following this, the image is then 'scored' based on the confidence of the network's predictions, and the similarity of its 'best guesses' to the target species.

The 'readme' file in this repository serves as a quick reference guide to Oak's various modules, including inputs and outputs. This file will be a more in-depth guide, as well as detailing the training process and model architecture.

Files Description

Python Scripts

First, we will describe the various Python files supplied with the project. These are separate Python sub-functions that work together. They were ported into Python from one R file which handled the whole process. The decision was made to move to Python for 2 main reasons:

1. Ease of deployability (standalone application or web interface)
2. Increased speed, especially for image processing and the Keras/TensorFlow library

professor_oak.py

This is the central function which handles user input/output and co-ordinates the other three sub-functions. It first loads those sub functions, as well as the *json*, *numpy* and *tensorflow* libraries, and the *professor_oak_cnn.h5* network model (described below).

The main inputs to the Oak function are the name of a target species (needed for scoring purposes), and the path to the target image. Other options are available to control output. The function takes these inputs, and calls each of the three sub-functions in order; *image_preprocssing.py*, *model_inference.py* and *generate_scores.py*. Details on these are given below. By default it then prints the output of these scores in a human-readable format. This printing function can be disabled with the *print_out* argument, and a JSON-formatted output can be enabled with *return_JSON*.

image_preprocessing.py

This function uses the *PIL*, *math* and *numpy* libraries. It takes an image path, loads it, and prepares it for model inference. First, it must be noted that the model only works for images with a white background and no transparency. Secondly, these images must have dimensions of 86×86 pixels, and three colour channels.

The image is loaded, and cropped so it extends to the limits of the canvas. This cropping assumes the image has a white background. The image is rescaled using bilinear interpolation such that its longest axis is 64 pixels. White space is then used to pad the shorter axis to 64 pixels also. The result is a 64×64 image, which is then placed in the centre of a 86×86 white canvas.

This was originally done to match up with the training data (see 'Training Details'). In brief 64×64 was the size of the smallest training images. Larger images were resampled down to this value to standardise training. The buffer region to 86×86 was added to allow the rotating of training images without clipping during augmentation.

Thus, this process is now somewhat outdated. It must be noted that all input images will therefore have a white border, which is not necessarily the case for training images. Future plans is, after some testing, to remove this functionality and simply scale images to 86×86.

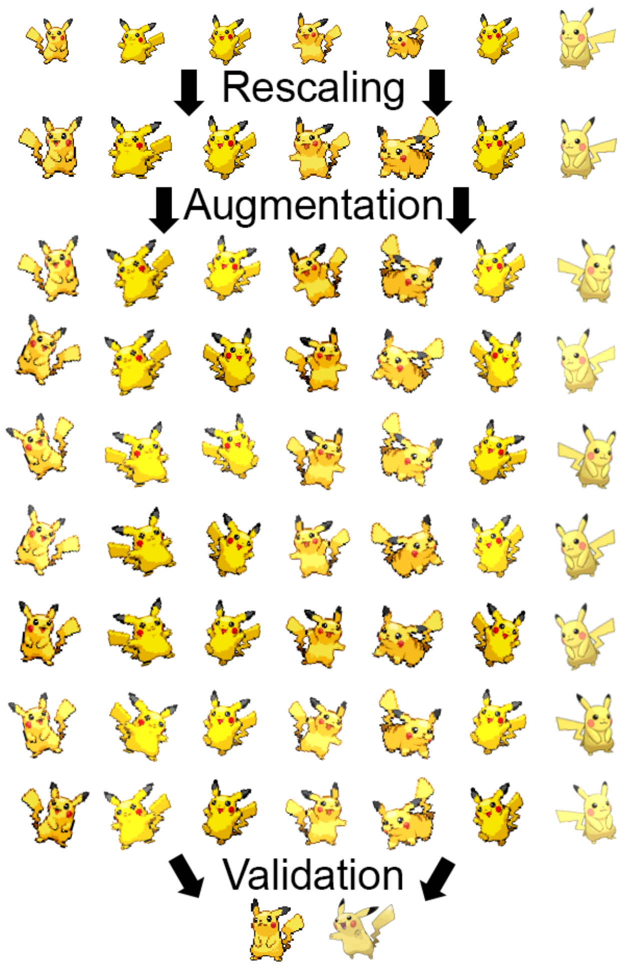

Figure 1: Example training image augmentation pipeline. ELABORATE

model_inference.py

generate_scores.py

Auxiliary files

professor_oak_cnn.h5

pokedex.json

Network Architecture

Training Details

Image Augmentation

In order to make the final model more robust to changes in pose, lighting, or indeed, large-scale morphological features, training images were run through an image augmentation pipeline. Parameters of this pipeline can be found in Table XXX, and example augmented images in Figure 1.

Validation Set

To ensure the model's ability to generalise to new data, a validation set was created consisting of 2 images per species not used in training. These consisted of sprites from the Gen V games (Black/White), and the official Ken Sugimori artwork for each species.

The Gen V sprites were chosen because they are the highest resolution ones available (96×96), and such often require downscaling to match Oak's required resolution of 86×86. As we believed it likely that most input images would be drawn at a greater resolution than this, it was prudent to ensure that Oak's performance be validated on accurate but downsampled images.

The Ken Sugimori images were chosen as they are less like sprites, and more like drawings. They tend to have softer lines and less saturated colours than sprites, and with more variation in pose. As such, using them for validation ensured Oak would be better at picking out patterns of colours and features to generalise with, rather than specific collections of pixels defining the sprite-like images.

Inference Details

Scoring System

Tips for Use and Limitations