CYNDER PROJECT

IMAGE ORIENTED CLASSIFICATION.

Brytan M Kelly

Table of contents.

- Uses cases (1)
- History (2)
- Files (3)
- Naming conventions (4)
- What am I looking at? (5)

Use cases

The Cynder model has many applications and use cases, a few of which are

Image segmentation,
Distant object detection,
Static and dynamic object detection,
Live time detection and efficient framework,
And even more!

History

Cynder was made due to the despair of one certified AI architect and engineer named Brytan Kelly. He is fourteen years old and a proficient coder in thirteen coding languages. He wasn't satisfied by the models on Open Model Zoo when he realized the limitations of the classification AI’s like Yolov3, and MobilenetSSD. He was working on a project that involved the classification of firearms and there were no good models or datasets for this task, as such he created his own system for creating, training, and running his models. Cynder was the product of this creative process. Cynder was named after a purple dragon from Skylanders (An old WII game) who was once evil and then was turned to the light side. He believed that this AI perfectly models his transition in the technological industry from a grey hat hacker into a pure white hat creator and hacker.

Files

There are many files in the folder containing the models, all of which have the “Cynder” naming convention. There are two “.h5” files which are pretrained models that can be loaded with the attached model loaders. The model trainer and model loaders are pretty self explanatory, but the image loader uses openCV2 to read images and pass them through the model, while the Camera loader utilizes your devices main camera to detect the classified object utilizing your camera. Meanwhile the training data section is meant to train a custom model using a training batch of photos which can be PNG, or JPEG, or BPM format. The input format for the trainer looks like this

The training data should be organized into subdirectories based on class, with each subdirectory containing all the images for that class. For example, if you have two classes "cat" and "dog",

you should have two subdirectories "cat" and "dog", each containing all the images for that class. The validation data should be organized in the same way as the training data, but in a separate directory.

You can use the ImageDataGenerator class to apply data augmentation to the training data. This class generates batches of augmented data in real-time during training.

You can use the image_dataset_from_directory function to load the validation data. This function creates a tf.data.Dataset object from image files in a directory, and can automatically label the data based on subdirectory names. In the code provided,

As an example
the training data is located in the directory "C:\Users\NULL\PycharmProjects\Object_Classification\venv\Data\Spyro\train",

and the validation data is located in the directory "C:\Users\NULL\PycharmProjects\Object_Classification\venv\Data\Spyro\Test". The training data is loaded with data augmentation using an instance of ImageDataGenerator, and the validation data is loaded using image_dataset_from_directory.

Trained model links

https://drive.google.com/file/d/1Vsa7SPXIvA_Itu3q556T6E4S_WmiyY0F/view?usp=share_link Person detection
Works well with video loader.

https://drive.google.com/file/d/10BW8ZBpK5qokYsQpvizvbHj4QO7mUZqO/view?usp=share_link Pneumonia detection
Works well with image loader

What am I looking at?

You are currently viewing a combination of blood sweat tears, and three days of editing hyperparameters. In all seriousness though what you are viewing is an object classification and detection module that can also classify images. I have tested it on person detection, hand detection, action figure detection, and pneumonia classification. This framework allows for quickly made well trained models   and seamless exporting and importing from multiple devices. This framework also allows you to share models with ease and load them even easier. Please keep in mind that this code was written by a fourteen year old who was incredibly tired and surviving only off of caffeine and boba tea so if you get an error that is to be expected.

Requirements

To run the code, you will need: Python 3.x installed on your system TensorFlow 2.x (or higher) library installed. You can install it using the command: pip install tensorflow OpenCV (cv2) library installed. You can install it using the command: pip install opencv-python Numpy library installed. You can install it using the command: pip install numpy The Spyro dataset downloaded and saved in the appropriate directory. A webcam connected to your system (for object detection). Once you have installed all the required libraries and downloaded the dataset, you can run the code in a Python environment such as Anaconda, Jupyter Notebook, or any other Python IDE.

“AI is the future, we cannot deny it, we must embrace it, from NLP to classification, to augmentation, Ai is everywhere, as a future Ai engineer and a youth of this world, I must say that I am excited to see where this rabbit hole takes us.” -Brytan Kelly

Brytan Kelly

3/26/2023
Visit Brytankelly.com to email me with questions, comments and ideas!