CS 5600/6600: F24: Intelligent Systems
Project 1: Exploring A Tiny Tucson, AZ USDA-ARS Bee Hive Frame Image Dataset

Vladimir Kulyukin
Department of Computer Science
Utah State University

September 25, 2024

Learning Objectives

1. Deep Learning (DL)
2. Training and Testing YOLOv8n on Annotated Image Datasets

Introduction

The main objective of this project is to give you exposure to YOLO (You-Only-Look-Once), one of the state-of-the-art DNN architectures used in many image and video analysis projects. For my service to the scientific community, I serve on the editorial boards of two scientific journals. I try to focus my peer review efforts on publications about precision apiculture and pollination. In the past 4 years, I have not seen a single submission (that I reviewed) that did not contain a YOLO version either as the main DL model or as a benchmark. So, my hope is that playing with YOLO is a useful experience to you, regardless of whether you plan to go into the industry or pursue a research-oriented academic career.

YOLOv8 was released in 2023. So it's considered bleeding edge. YOLOv1 was released in 2014/15. When I played with it, I was not impressed. I ignored YOLOv2 because of my experiences with YOLOv1. But, when I tried YOLOv3 (it was released in 2018) and was pleasantly surprised, and have been using YOLOv3, YOLOv4, and YOLOv7 in some of my research projects. One nice thing about YOLOv4, YOLOv7, and YOLOv8 is that they have tiny versions (the letter `n` in YOLOv8n, for example, abbreviates *nano*). The tiny versions do not require GPUs. The YOLOv8 documentation claims that the size of YOLOv8n is 6.5MB, which is amazing to me.

My Lecture 09 PDF in Canvas gives you some research background and motivation for this project. I also published Reagan Hill's guest presentation on using YOLOv8n to classify bee hive frame images. You may want to take a look at these documents before moving on.

Project Scope

Your aim in this project is to train and test YOLOv8n on the annotated dataset of 52 hive frames from the Tucson, AZ USDA-ARS (Agricultural Reserarch Service) frame image reservoir collected by Dr. William Meikle's research program. Dr. Meikle is a research entomologist at the USDA-ARS in Tucson, AZ.

The zip archive of this project contains the folder `data/obj/` that has 52 PNG and 52 txt files (or 52 PNG-txt pairs). These PNG files were annotated with LabelImg for use with YOLO. Each image, e.g., `img8604.png`, has the corresponding txt file, i.e., `img8604.txt`. You do not need to change these files. The txt files contain the YOLO annotations. Each line of a txt file looks as follows.

```
4 0.8783854166666667 0.22083333333333333 0.0109375 0.017592592592592594
```

The first number is a label mapped to a non-negative integer. Recall from Lecture 09 that we have 7 bee frame cell categories: CappedHoneyCell, CappedWorkerBroodCell, EmptyCombCell, PollenCell, UncappedNectarCell, UncappedWorkerLarvaCell, BeeHiveFrame. Thus, CappedHoneyCell is mapped to 0, CappedWorkerBroodCell – to 1, EmptyCombCell – to 2, PollenCell – to 3, UncappedNectarCell – to 4, UncappedWorkerLarvaCell – to 5, and BeeHiveFrame – to 6. The other four floats on the same line are the normalized coordinates of the top left and bottom right corners. These numbers are automatically generated from the annotation XML after an image is annotated.

You install YOLOv8 with the following command.

```
pip install ultralytics
```

Let's suppose you install it in the directory `yolov8`. You can then create `data.yaml` file in that directory that should look as follows.

```
train: data/obj
val: data/obj

nc: 7
names: ['CappedHoneyCell', 'CappedWorkerBroodCell',
        'EmptyCombCell', 'PollenCell', 'UncappedNectarCell',
        'UncappedWorkerLarvaCell', 'BeeHiveFrame']
```

The `train` attribute indicates the path to the folder where the PNG images and the corresponding text files used for training are saved.

The `val` attribute indicates the path to the folder where the test/validation images and the corresponding text files used for testing.

Since our dataset is so small, you may want to put 80% (i.e., 42 images and text files) of the images and the their text files into the train directory and 20% (i.e., the remaining 10 images and the corresponding text files). You can either select the training image-text pairs randomly or just take the first 42 image-text pairs. I don't think that for the datasize this small it'll make a difference.

The `nc` (number of classes/categories) and names attributes of the `yaml` file should remain the same. To train a model, you can save the following Python code in `v8_custom.py`.

```
from ultralytics import YOLO

model = YOLO("v8test.pt")

train_results = model.train(data='data.yaml',
                            epochs=100,
                            imgsz=640,
                            device=0)

# evaluate performace on the validation set
metrics = model.val()

# perform obj detection on image
results = model("../data/obj/2013_07_28_CHBRC_010.png")
results[0].show()

# export format
export_model = model.export(format="onnx")
```

We can run this file from the command line as

```
python v8_custom.py.
```

This file trains the model, evaluates the model on the validation set, shows the results of evaluating the trained model on one individual image. The model then is saved in a `pt` (PyTorch) file. Each training and validation run creates the directory called `runs`, where the weights are saved as `best.pt` and `last.pt`.

The first time you train the model, you can use a default model (I think it's called `yolov8n.pt`). The subdirectory called `runs/detect/train` contains all kinds of performance statistics files, e.g., `F1_curve.png`.

Slide 11 in Reagan Hill's presentation contains a few links to online documentation.

Write a 5-page report documenting your experiences with YOLOv8n. The parameters you played with, the number of epochs, the F1 curve, the mean average precision, the hyperparameters you tuned. The tools you have found and used. Your installation experiences on a particular operating system. Did you use a GPU or Colab?

What to Submit

1. Your 5-page report saved as `cs5600_6600_F24_project_1.pdf`.
2. Your Python source code files you have written for this project. For example, you may write some additional tools to help you with your analysis. You may also come across some useful blogs whose code segments you have used. Please mention them in your report and explain what those are. Even an unsuccessful attempt shows me your effort.
3. The three best `pt` models you have trained.
4. Zip all your materials into `project_1.zip` and submit it in Canvas.

Happy Exploration!