Aerial Object Detection

Pranav Dulepet, Mukund Shankar, Anirudh Poruri, Sathya Gnanakumar

The Challenge

Prompt: ***Write a program that ingests a provided image set or video feed and then produces an event stream / log of objects identified*.**

We chose to detect aerial objects which may pose a threat to national security, such as…

“Balloons”

While many models currently exist to identify objects in images or videos, they are highly inaccurate (as we will demonstrate).

Our goal is to train a model such that it is able to **correctly** and **accurately** identify objects (potential airborne threats) using computer vision algorithms.

Our idea

Unidentifiable aerial objects are a threat to national security.

In light of recent events, we have seen the magnitude of threat these objects pose.

We need to be able to identify these objects as soon as possible to take the appropriate action.

The Opportunity

Generic CNN-based Object Detection models such as YOLO

Needs more training for specific context

The Opportunity

Figure 1: A balloon as (incorrectly) classified by generic YOLOv5 model

Figure 2: A group of hot air balloons as (incorrectly) classified by generic YOLOv5 model

The Opportunity

Figure 3: A video of a balloon as (incorrectly) classified by generic YOLOv5 model

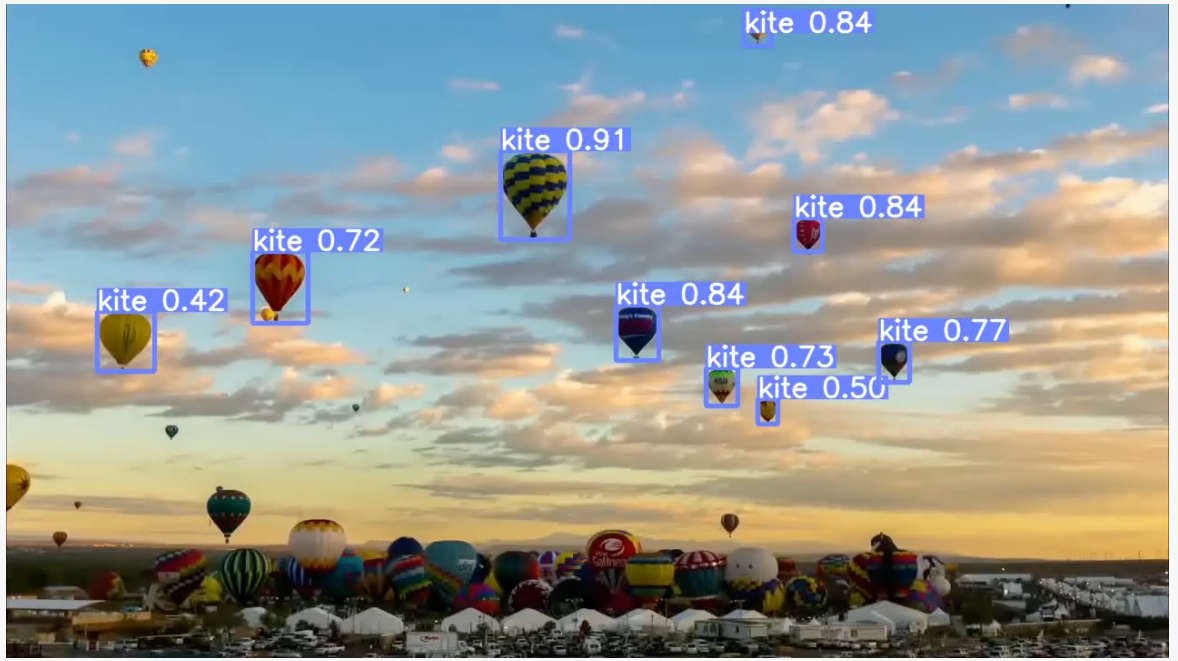


Figure 4: A video of a group of hot air balloons as (incorrectly) classified by generic YOLOv5 model

Our Solution

Generic CNN model

Transfer learning with contextual data

Inferencing

How does it all work?

YOLOv5

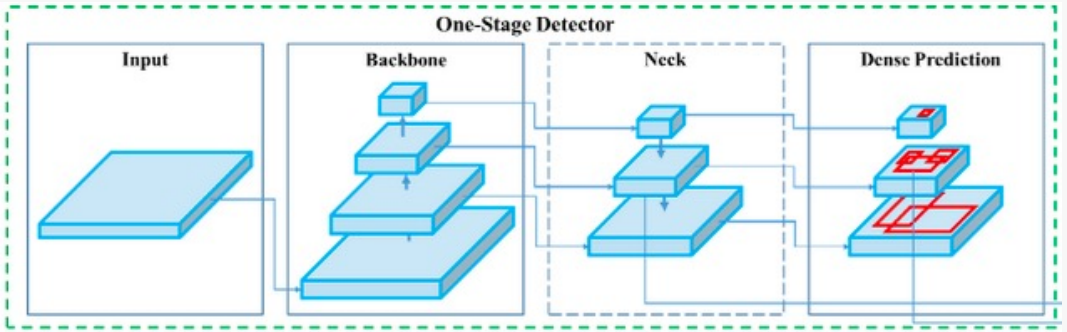


Figure 5: YOLOv5 architecture

Inferencing

Figure 6: Diagram describing inferencing

Transfer Learning

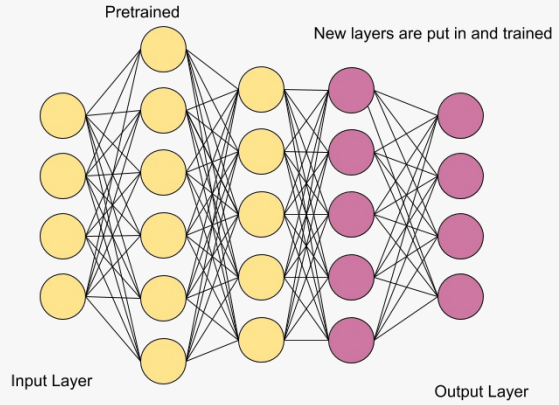


Figure 7: Diagram describing transfer learning

The code

This code snippet is used to acquire required libraries, load our dataset into the working environment, train our data, display metrics used to measure performance of our model, and export our model

Some intermediate steps were skipped from this code snippet (such as defining training/testing/validation data splits) as they were done separately.

```
# First, we identify our model
!git clone https://github.com/ultralytics/yolov5
# Then, we import our required libraries
import torch
from yolov5 import utils
import torch
import utils
from IPython import display
from IPython.display import clear_output
from pathlib import Path
import yaml
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import glob

# Then, we load our dataset into our working environment
!unzip '../balloons_data.zip' -d '../balloons-data'

# INTERMEDIATE STEP: ANNOTATE DATASET AND IDENTIFY TRAINING/VALIDATING/TESTING DATA SPLIT

# After making sure our dataset is present, we train it
!python train.py --batch 32 --epochs 150 --data '../balloons-data/data.yaml' --weights 'yolov5s6.pt' --
project 'aerial' --name 'feature_extraction' --cache --freeze 12

# Check training data to look at our training data statistics (such as precision and classification
loss)
display.Image(f"aerial/feature_extraction3/results.png")

# Look at our precision-recall curve
plt.plot(figsize=(20,20))
plt.title('Precision Recall curve', fontsize=20)
plt.tick_params(left = False, right = False , labelleft = False, labelbottom = False, bottom = False)
plt.imshow(mpimg.imread('aerial/validation_on_test_data/PR_curve.png'))

# Use Nvidia's tensorrt library to export our trained model into several practical, usable formats
!python export.py --weights 'aerial/feature_extraction3/weights/best.pt' --include engine onnx --data
'../balloons-data/data.yaml' --device 0 --imgsz 640 640
```

The code

This code snippet is what we used to perform inferencing.

First, we used the generic YOLOv5 model.

Then, we used the weights defined by our trained model (after applying transfer learning). Results coming up.

```
# Acquire the basic YOLO library (with no transfer learning)
!git clone https://github.com/ultralytics/yolov5.git

# Run object classification with no training
!python detect.py --source white-balloon-img.jpeg
!python detect.py --source hot-air-balloons-img.jpeg
!python detect.py --source white-balloon-vid.mov
!python detect.py --source hot-air-balloons-vid.mov

# Use the weights defined by our model and try object classification
#   on the same images/videos again
# after transfer learning
!python detect.py --weights best.pt --source white-balloon-img.jpeg
!python detect.py --weights best.pt --source hot-air-balloons-img.jpeg
!python detect.py --weights best.pt --source white-balloon-vid.mov
!python detect.py --weights best.pt --source hot-air-balloons-vid.mov
```

Our model in action

Figure 8: A balloon as classified by our model

Figure 9: A group of hot air balloons as classified by our model

Our model in action

Figure 10: A video of a balloon as classified by our model

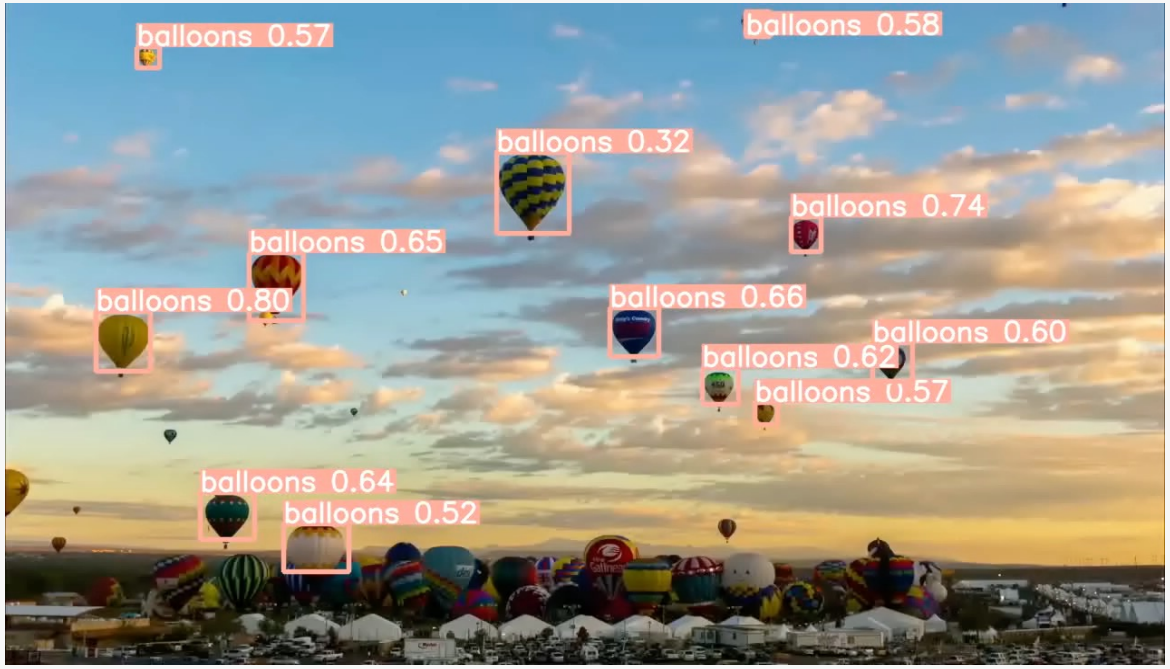


Figure 11: A video of a group of hot air balloons as classified by a our model

Metrics

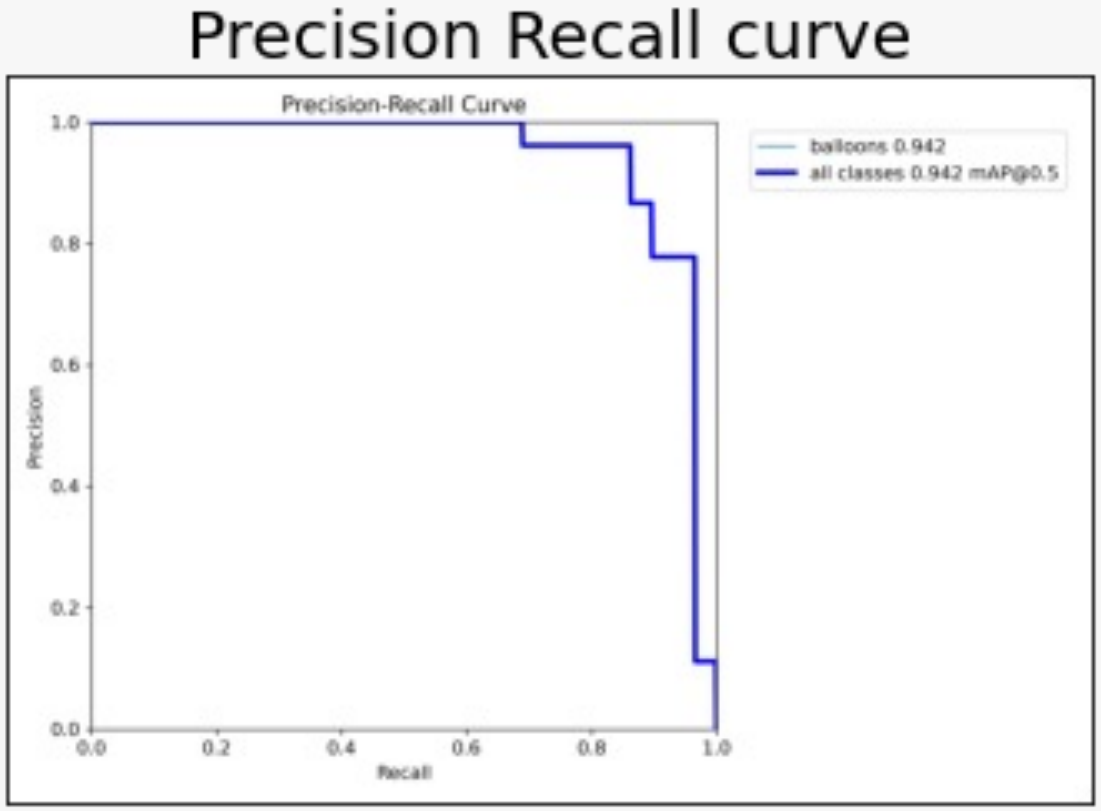


Figure 12: Quality and quantity of correctly classified inputs

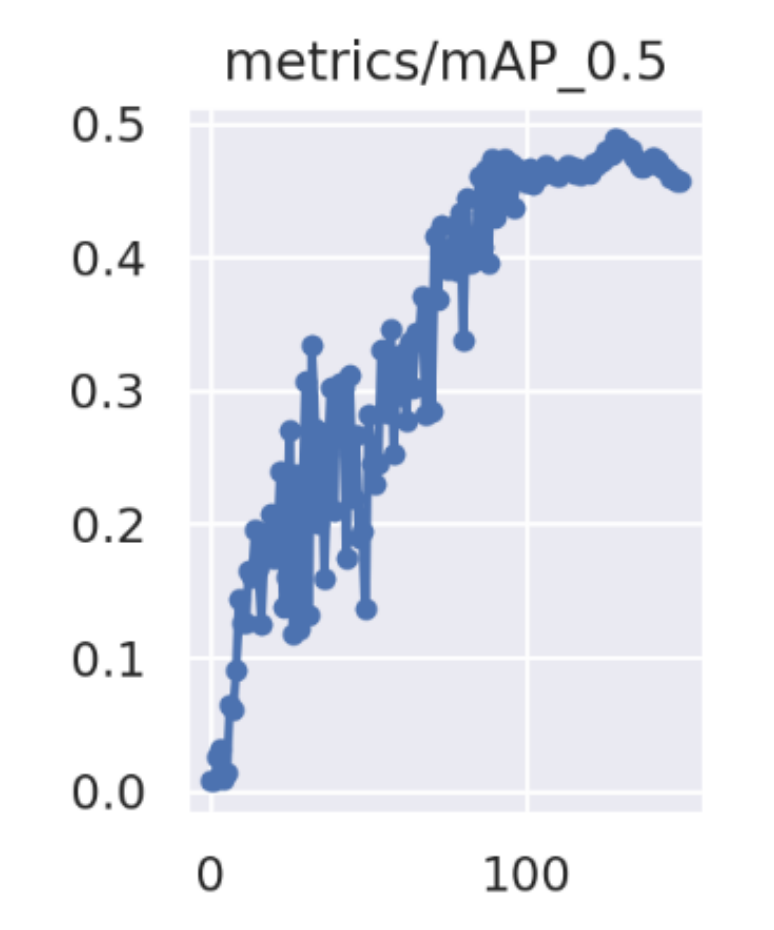


Figure 13: mAP score by epoch

Future Improvements

Another important factor to consider is how falsely flagging allies as hostiles and vice versa can be avoided.

To mitigate these risks, some steps can be taken:

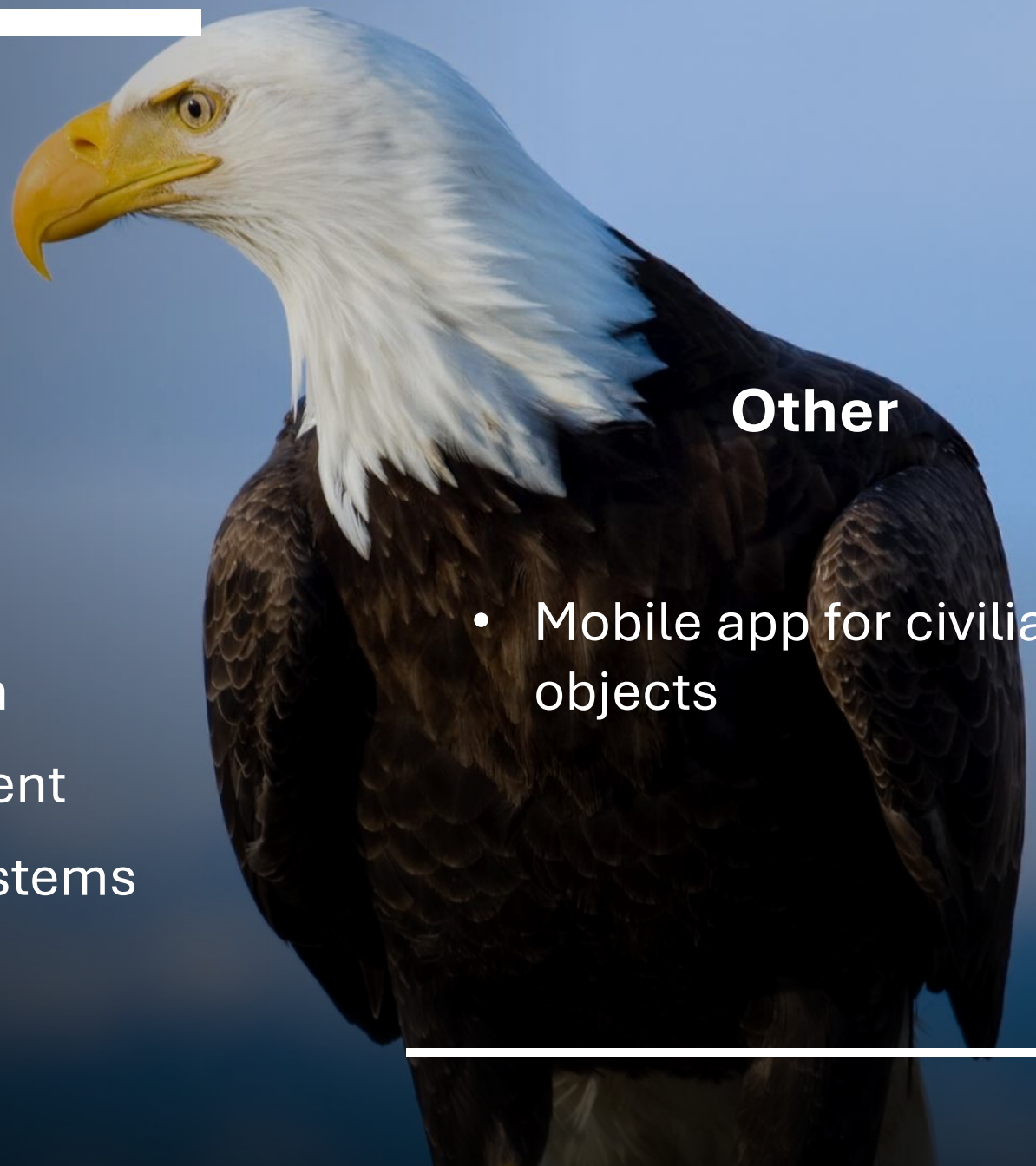
- Train on more contextual data
- More robust fine-tuning including changing weights, adding custom layers, etc.
- Incorporate a feedback loop

When identifying threats, we need to keep some threat factors in mind:

- ✈ Flight path
- ✈ Shape & Size of object
- ✈ Maneuverability
- ✈ Time in sky
- ✈ Time when object is detected

Threats could be larger, carry more equipment, have more maneuverability, and appear at odd times during the day

Use cases



Business

- Drone Imagery
- Ground Troop Protection
- Precise Target Engagement
- Autonomous piloting systems

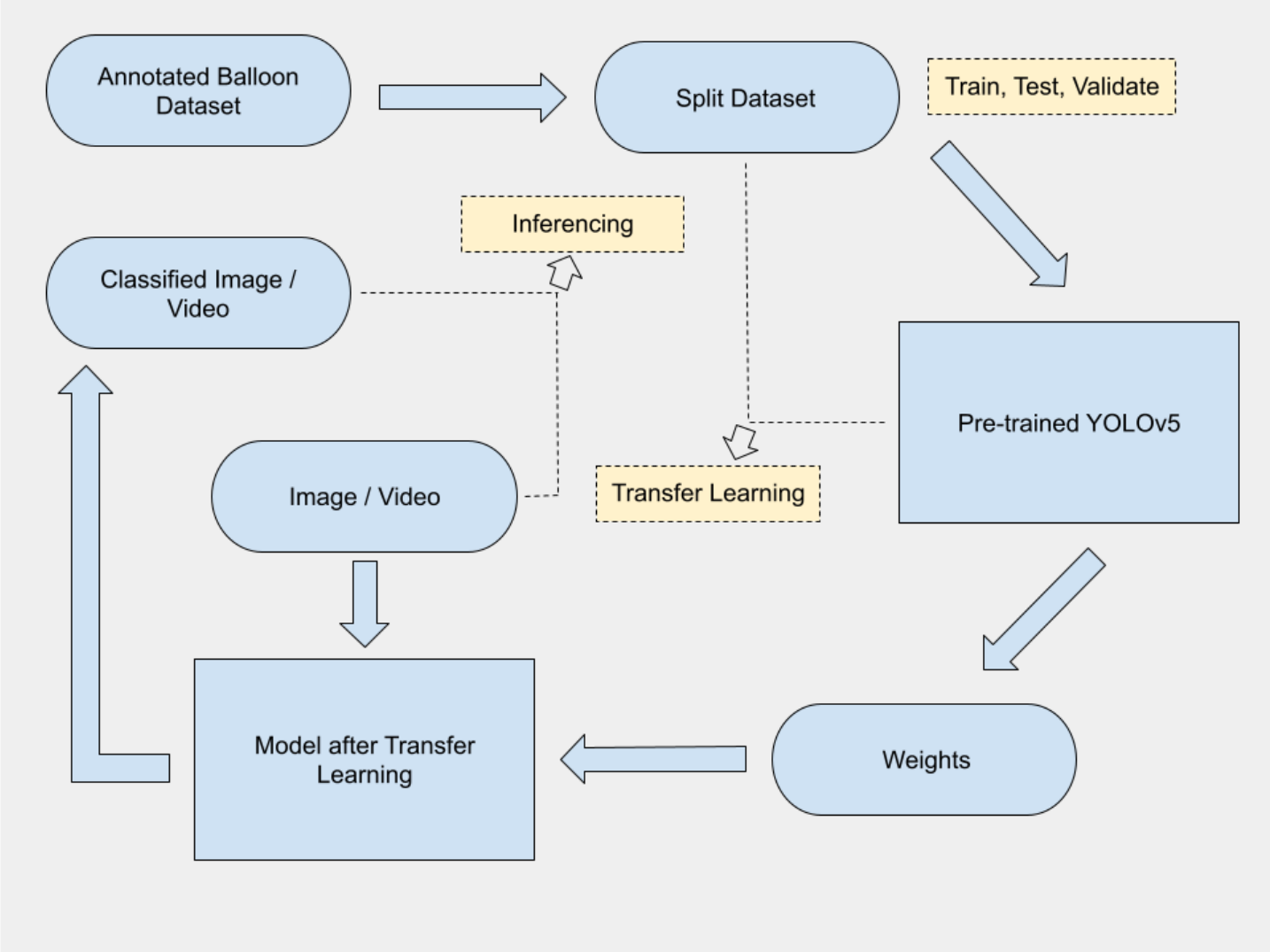
Other

- Mobile app for civilians to detect objects

Appendix

- High-level conceptual model
- References

High-level Conceptual Model



References

- Figure 1: https://www.nytimes.com/2023/02/04/us/politics/chinese-spy-balloon-obsession.html
- Figure 2: https://www.dailysabah.com/life/hot-air-balloon-festival-kicks-off-in-turkeys-famed-cappadocia/news
- Figure 3: https://www.tiktok.com/@tripadvizr/video/7196418536355548422?_r=1&_t=8aFwx2sxDf1
- Figure 4: https://www.youtube.com/watch?v=QGAMTlI6XxY
- Figure 5: https://iq.opengenus.org/yolov5/
- Figure 6: https://www.analyticssteps.com/blogs/how-transfer-learning-done-neural-networks-and-convolutional-neural-networks
- Figure 7: https://www.exxactcorp.com/blog/HPC/discover-the-difference-between-deep-learning-training-and-inference
- Figure 8: https://colab.research.google.com/drive/1E19M4QNhCYRgPpp8NUHwRt0NWobxXGZO?usp=sharing
- Figure 9: https://colab.research.google.com/drive/1E19M4QNhCYRgPpp8NUHwRt0NWobxXGZO?usp=sharing
- Symbols: https://developer.apple.com/sf-symbols/