Implementing XKCD 1425 with YOLO, Darknet and Python

Colin Sauze
<cos@aber.ac.uk>

SUPERCOMPUTING WALES
UWCHGYFRIFIADURA CYMRU

Cronfa Datblygu Rhanbarthol Ewrop
European Regional Development Fund

Overview

From: https://xkcd.com/1425/

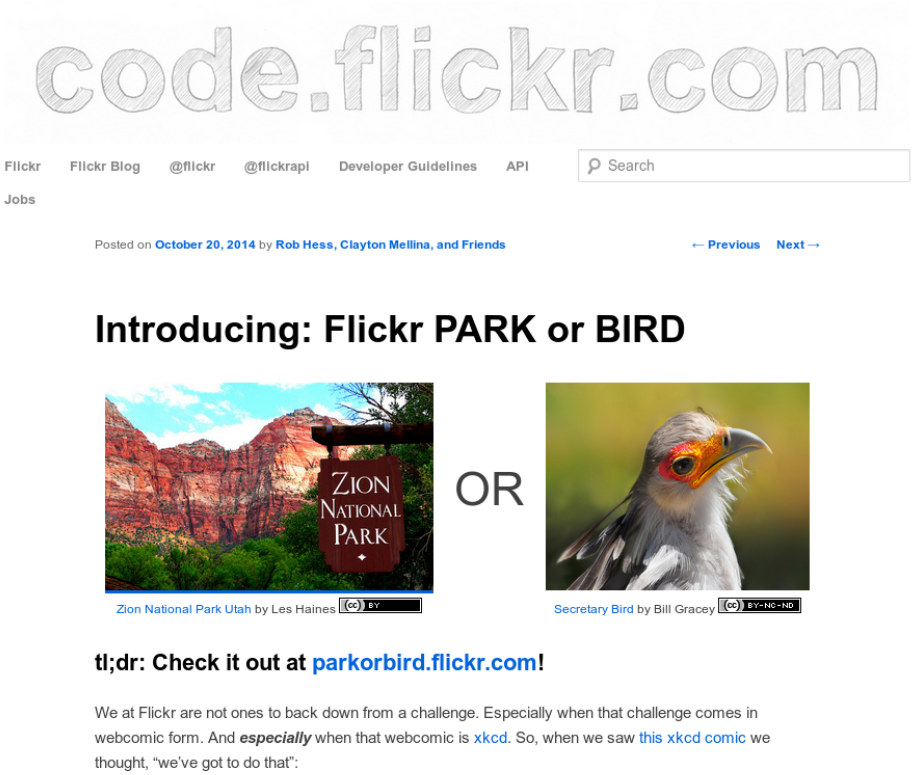
Published: September 24th 2014
(LESS THAN FIVE YEARS AGO!)

- Motivations
- Has this been done before?
- YOLO and Darknet
- Are we in a national park?
- Putting it all together
- Results

Motivations

- I’m a Research Software Engineer for Super Computing Wales
- Help researchers across the university use High Performance Computing
- Lots of people asking me about Deep Learning/CNNs/Image recognition
- Decided to read up on it
  - Found Darknet/YOLO
  - Realised implementing XKCD1425 would be easy with this
  - Nice demo to show what it can do

Has this been done before?



code.flickr.com

Flickr Flickr Blog @flickr @flickrapi Developer Guidelines API Search

Jobs

Posted on October 20, 2014 by Rob Hess, Clayton Mellina, and Friends ← Previous Next →

Introducing: Flickr PARK or BIRD

OR

Zion National Park Utah by Les Haines (cc) BY

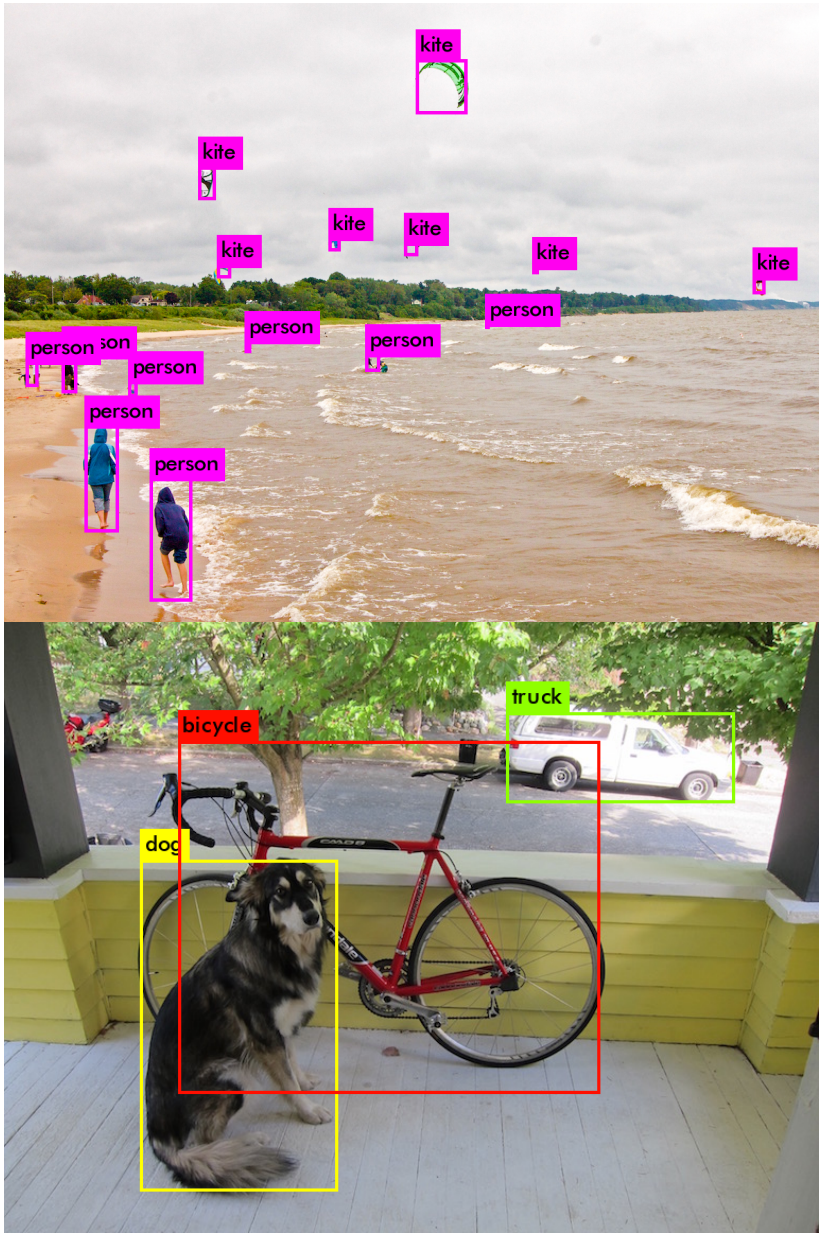
Secretary Bird by Bill Gracey (cc) BY-NC-ND

tl;dr: Check it out at parkorbird.flickr.com!

We at Flickr are not ones to back down from a challenge. Especially when that challenge comes in webcomic form. And ***especially*** when that webcomic is xkcd. So, when we saw this xkcd comic we thought, "we've got to do that":

- October 2014, Flickr released "Park or bird"
- Tells you if an image is a park OR a bird
- Comic said "In a park AND a bird"
- **http://parkorbird.flickr.com/**
  No longer online

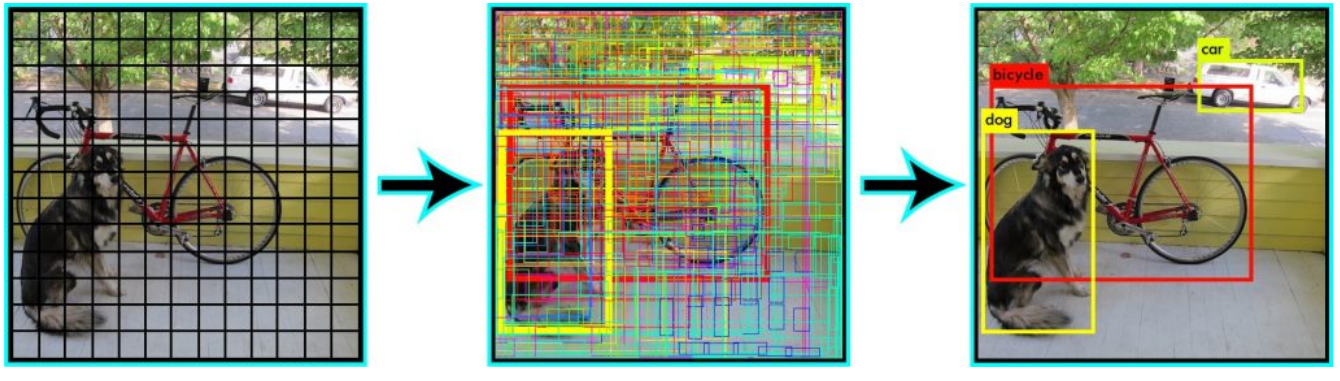
YOLO and Darknet



- Darknet very efficient neural network library
- YOLO = You Only Look Once
  - Object recognition + localisation using Darknet
  - Single neural network does the entire task
- https://pjreddie.com/darknet/yolo/
- Video demo:
  - https://www.youtube.com/watch?v=VOC3huqHrss

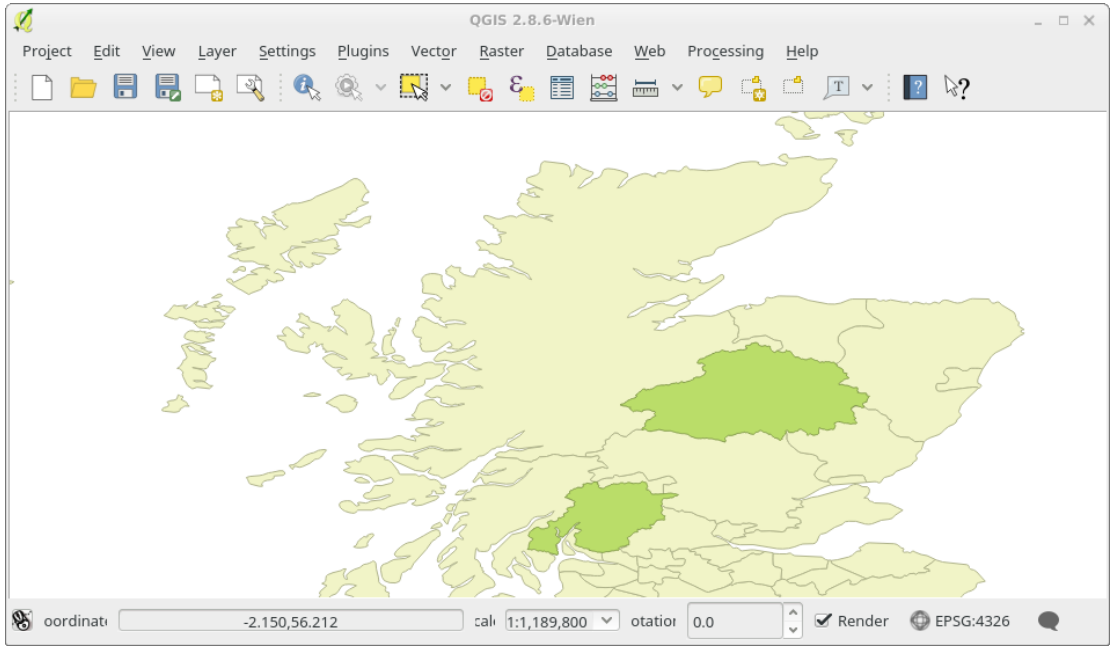
How YOLO Works

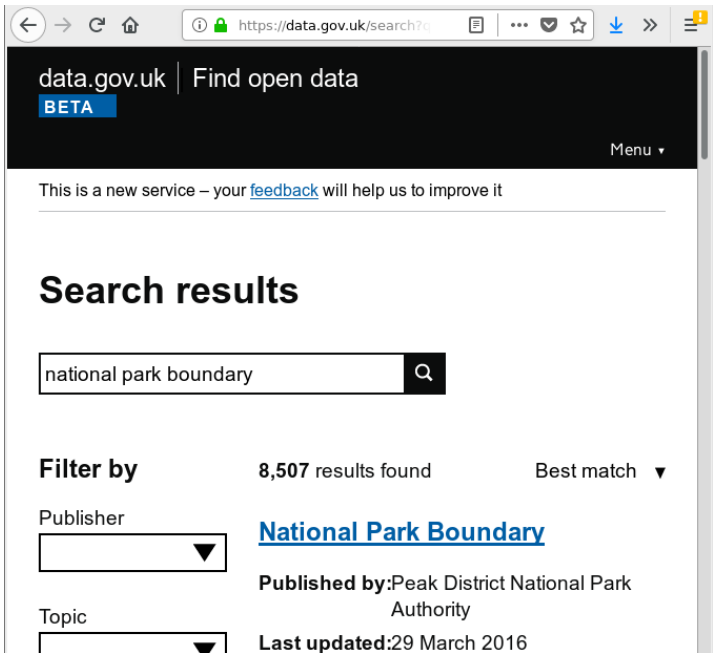
- Single Neural Network processes whole image
- Divides image into regions, makes bounding boxes
- Predicts what class that box belongs to
- Faster than alternatives, comparable accuracy
- Trained on Common Objects in COnext (COC) dataset
  - 300k tagged, segmented images
  - 80 categories, including birds



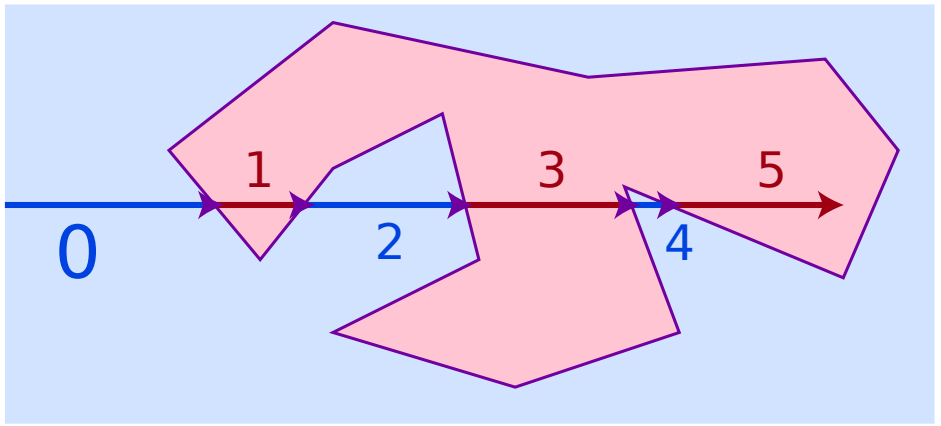
Getting Park Boundary Data

- Data.gov.uk UK national park boundary data
  - Each park is separate download
  - Haven't looked at other countries yet
  - Open Street Map could supply data too
- Converted coordinate format with GDAL (Geospatial Data Abstraction Library)





Are we in a national park?




```
# read the shapefiles
parks[0]= shape(shapefile.Reader("Cairngorms.shp").shapes()
[0])
parks[1]= shape(shapefile.Reader("LochLomond.shp").shapes()
[0])
#store the park names
parknames = [ "Cairngorms", "Loch Lomond"]


def get_park_name(lat, lon):
    """Returns the name of the park this point is in"""
    point = Point(lon, lat)
    for i in range(len(parks)):
        polygon = parks[i]
        if polygon.contains(point):
            return parknames[i]
```

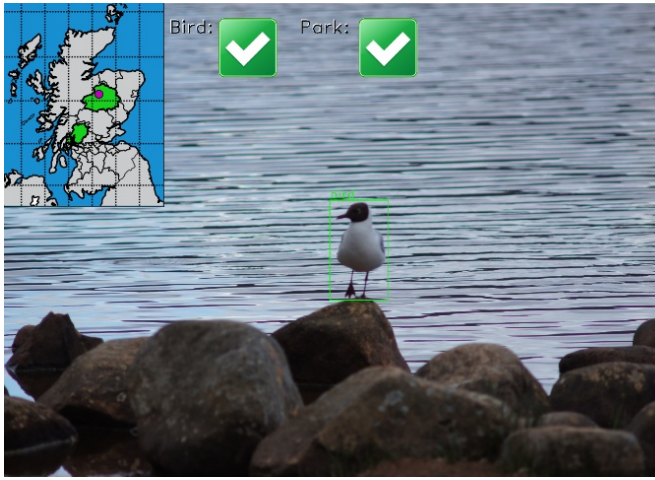
- Latitude/Longitude of an image from its EXIF JPEG header
  - Assumes camera has a GPS and its accurate
- Draw a line from the point through the park boundary polygon
  - Even number of intersections = outside park
  - Odd number = inside park
- Python Shapely library does this for us

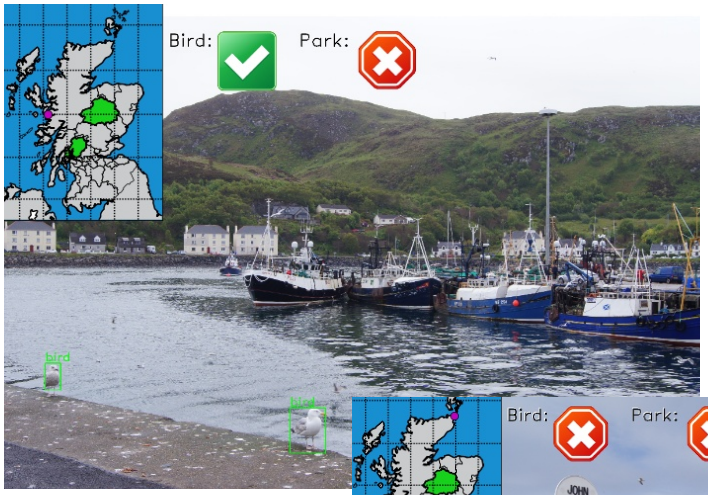
Putting it all together

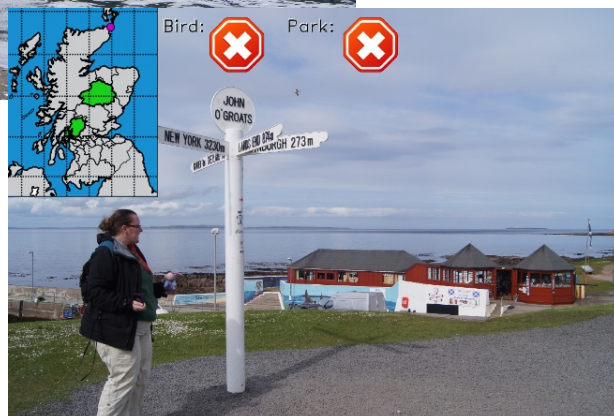
- Approximately 200 lines of Python
  - Mostly utility stuff like loading files
- Darknet code requires 4 lines:

```
dn.set_gpu(0)
net = dn.load_net("cfg/yolov3.cfg", "yolov3.weights", 0)
meta = dn.load_meta("cfg/coco.data")
r = dn.detect(net, meta, "image.jpg")
```

- Annotated image output
- Outputs a CSV file listing each image



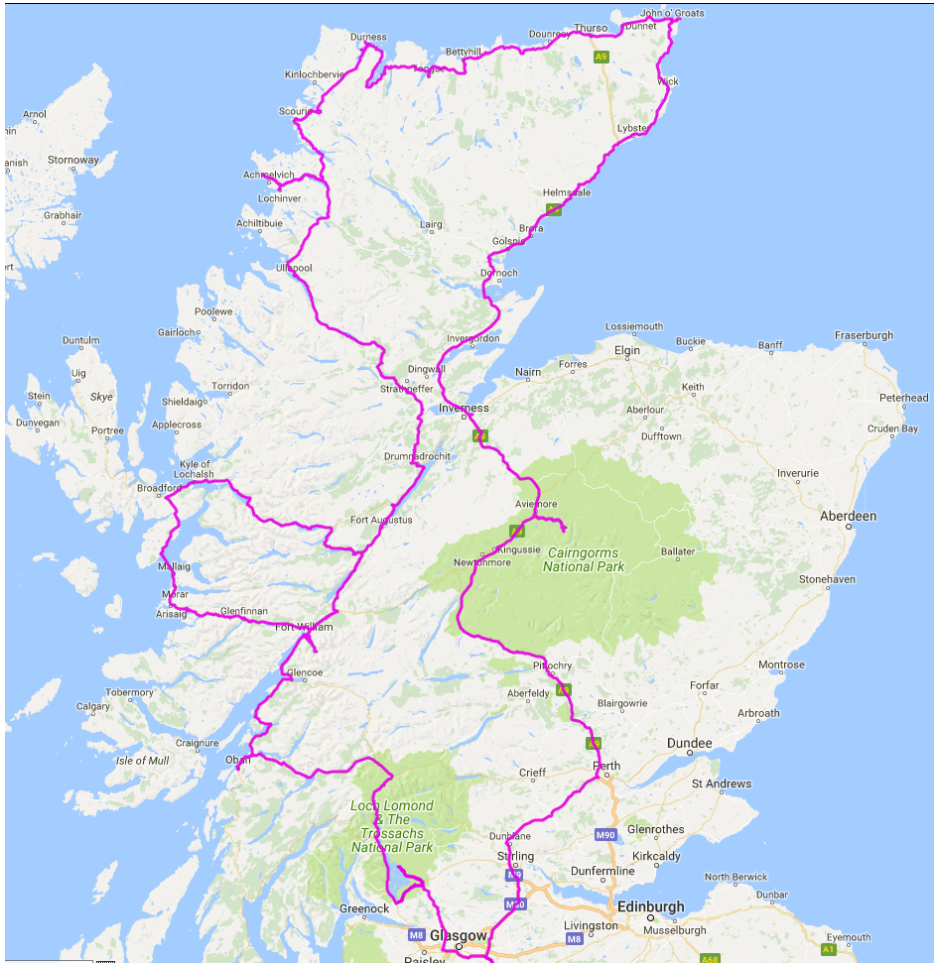




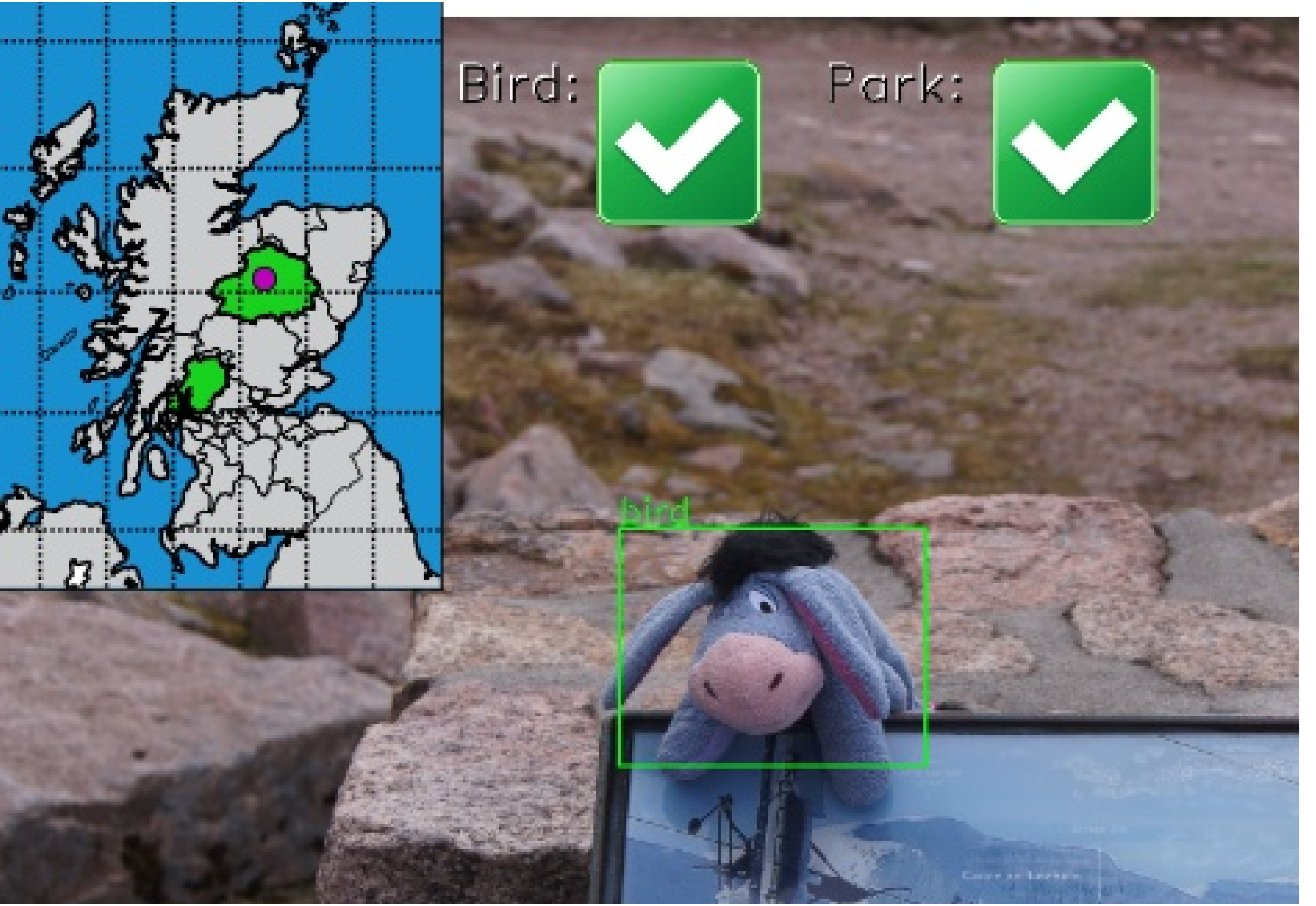
Basic algorithm:

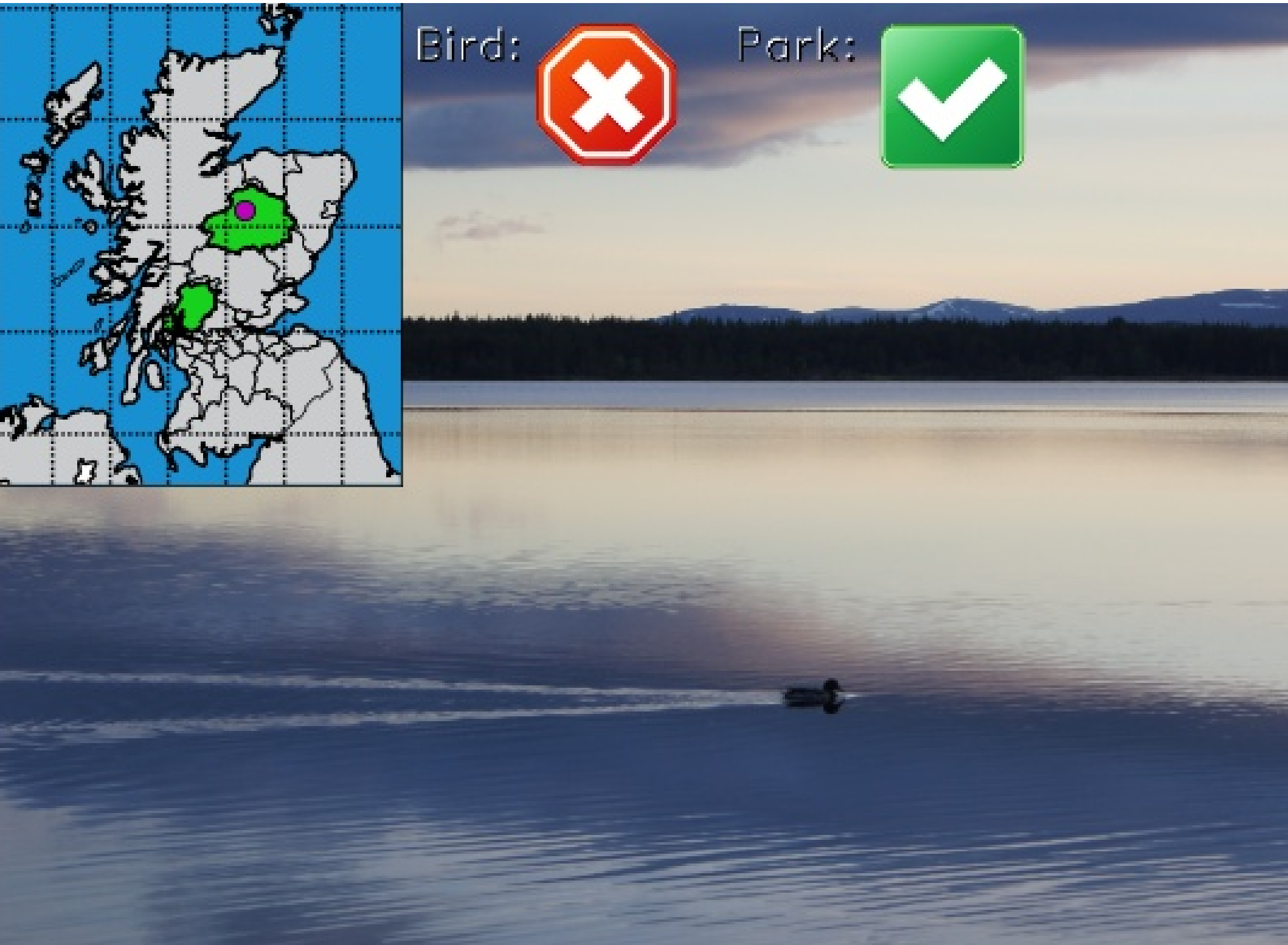
- Are we in a park?
  - Yes: Run bird detection
    - Was a bird found?
    - Yes: Draw bounding box
    - No: Go to next image
  - No: go to next image

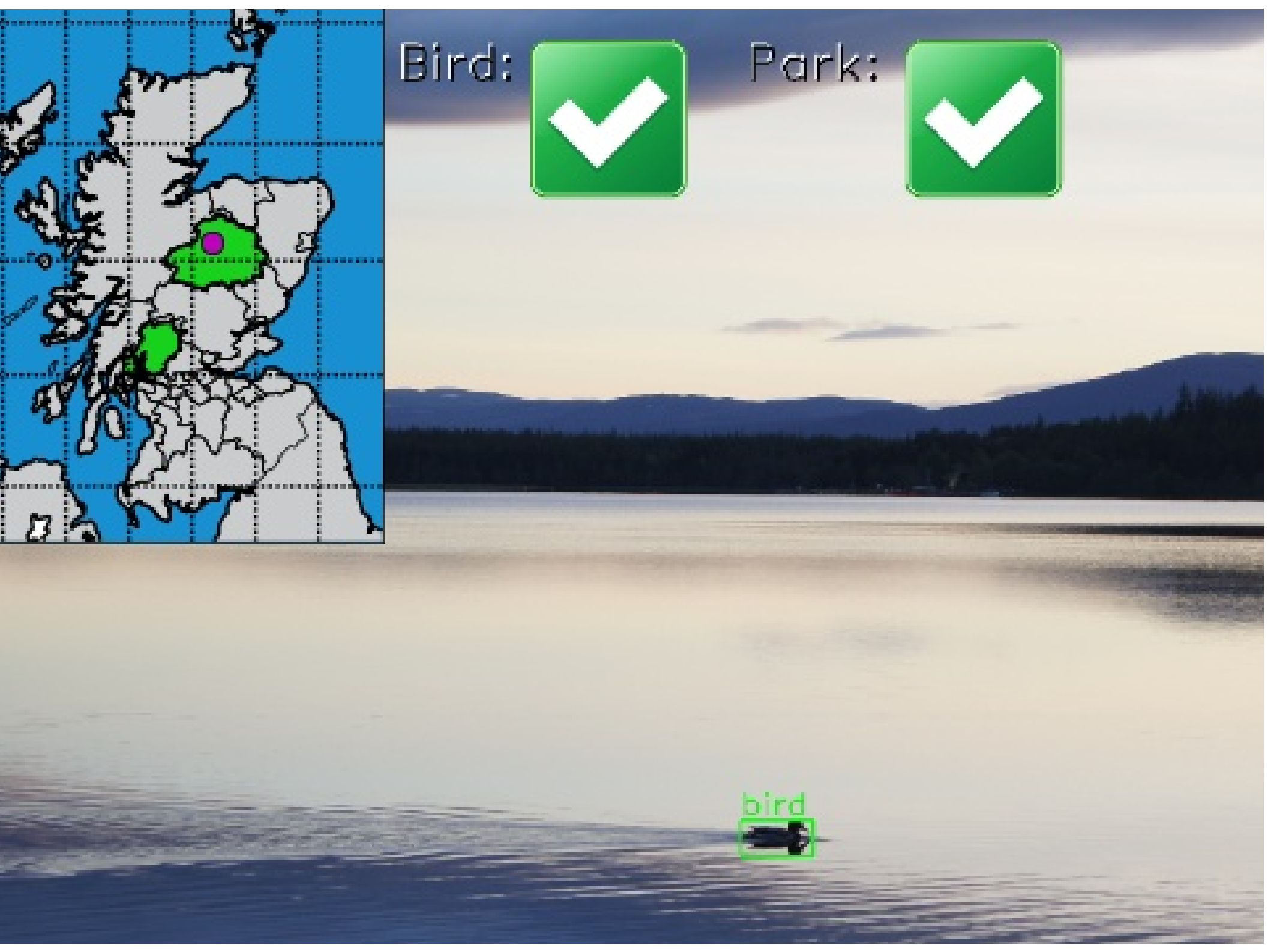
Testing – My Scottish Holiday 2015



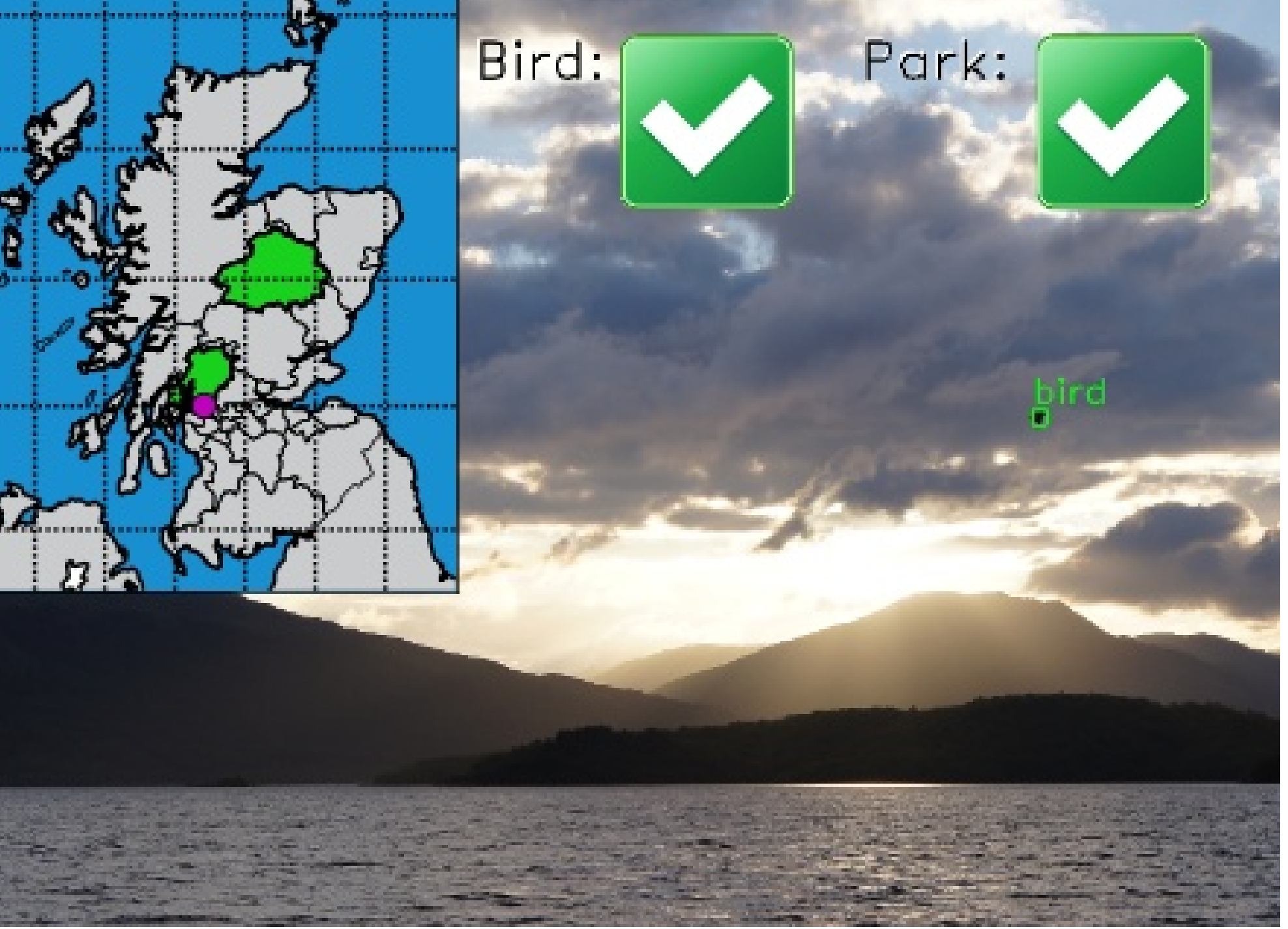
- One week camping in Scotland
- Visited Cairngorm and Loch Lomond National Parks
- 506 Geotagged images from Sony A55 camera
- 35 contain birds
- 12 identified correctly
- 12 false positives
- 23 false negatives


Bird:
Park:
bird


Bird:
Park:


Bird:
Park:
bird


Bird:
Park:
bird
bird
bird


Bird:
Park:
bird

Performance

- YOLO/Darknet can do real time 30fps on a GPU
  - I haven’t got a suitable GPU….yet
- 2 hours, 40 minutes to check 506 images
  - Intel i7-7500U CPU, quad core but only using one core.
  - Just under 19 seconds per image

Conclusions

- Not perfect performance
  - Some of the false negatives surprised me
  - Stuffed donkeys are apparently birds!
  - Threshold changes might have reduced false negative rate
- I’m still impressed by YOLO
  - Writing this was easy
  - I did no training of neural nets or tweaking of threshold parameters
- COCO only has 3362 bird images, most fairly close up
  - http://cocodataset.org/#explore
  - Could use ImageNet data.
    - Millions of images
    - Not annotated, no localisation
- Code available from https://github.com/colinsauze/xkcd1425-with-yolo