NaluNet

Darknet + YOLO v3 + Python/Flask

By @pjcr and @lemiffe

Special thanks to @jessedobbelaere, @bugragokalp, @tmeire_, @pjreddie, @wonko_be

THE PROBLEM

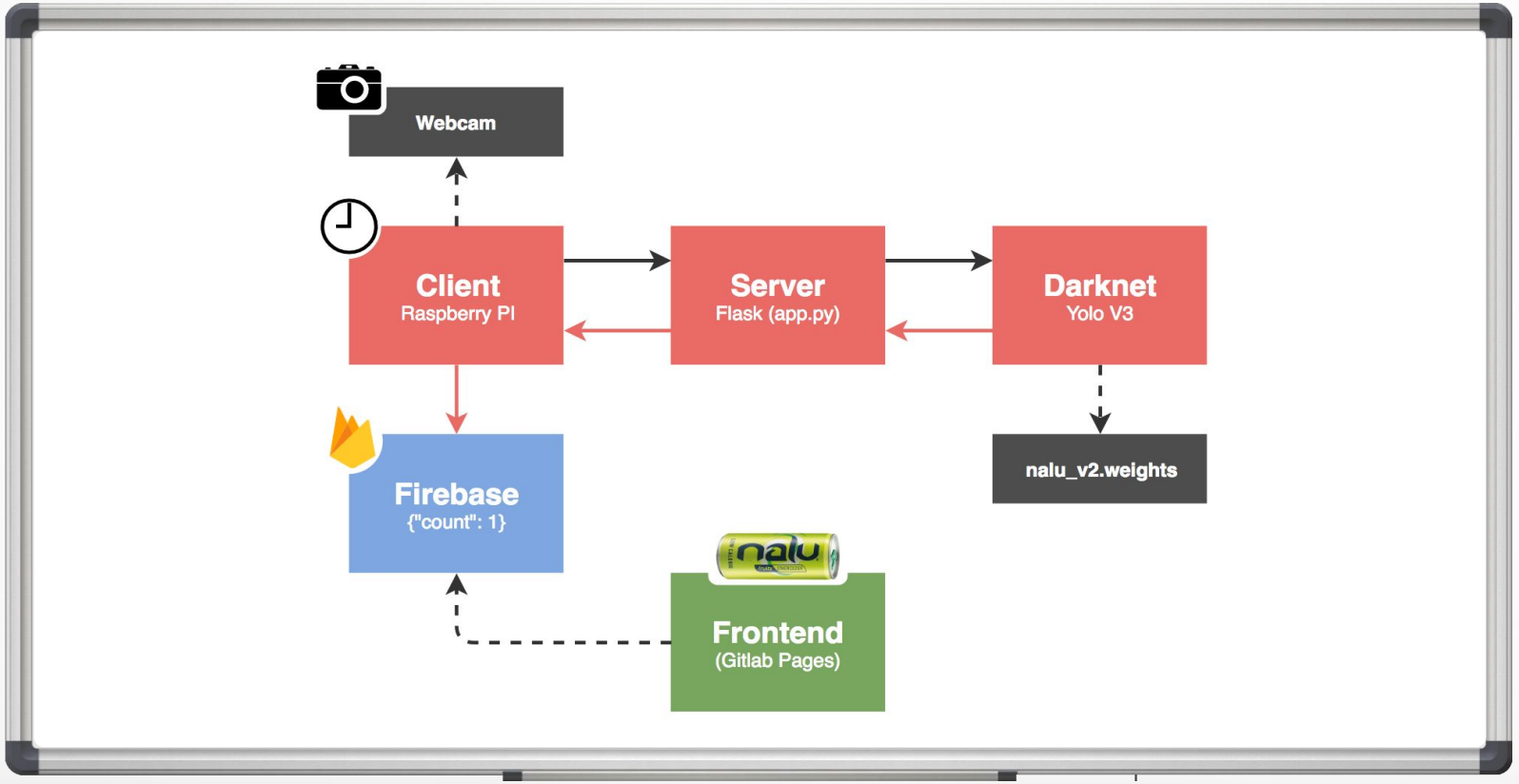
IS THERE ANY NALU?

THE GOAL

CREATE A **NEURAL NETWORK** THAT CAN DETECT THE **PRESENCE OF CANS OF NALU** AND UPDATE A WEBSITE WITH THE ESTIMATED COUNT EVERY ~10 SECONDS.

(YES I KNOW, WE ARE LAZY :D WE COULD JUST STAND UP, WALK FOR A BIT AND CHECK, BUT THIS IS WAY MORE FUN)

THIS IS HOW IT WORKS...



STEP 1

TRAINING THE AI


FIRST WE TOOK OVER 100 PHOTOS

Labelbox

Tasks Mike Warner

home / Nalu

Nalu Detection System

START LABELING

You've labeled everything available to you. Other labelers must finish the remaining labeling tasks.

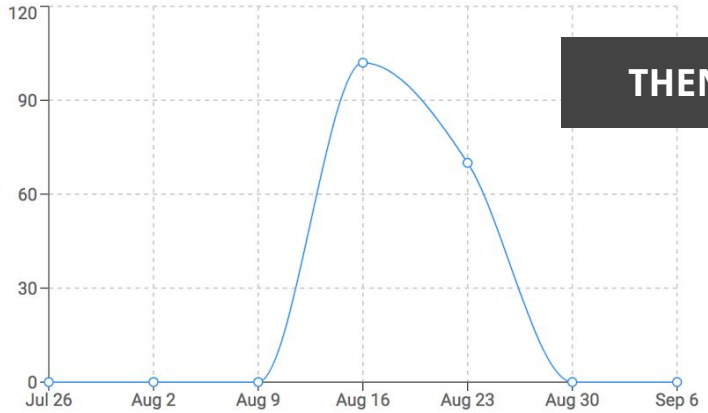
OVERVIEW ACTIVITY PERFORMANCE EXPORT SETTINGS

Progress

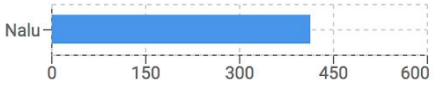
155	70	17	71%
Submitted	Remaining	Skipped	Complete

Labeling Activity

Weekly



Object Count



THEN WE PUT BOUNDARIES AROUND EACH NALU CAN

```python
import pandas as pd
from shutil import copyfile
from PIL import Image, ImageFont, ImageDraw, ImageEnhance
from os import path

boundingBoxesFile = 'nalu.json'
picturesFolder = 'Nalu_Pictures'
prefix = 'data'
outputFolder = 'obj'
checkFolder = 'check'

df = pd.read_json(boundingBoxesFile, orient='values')

def check():
    pass

def parseBounds(coordsArr, img_height, img_width, label):
    x_l = coordsArr[0]['x']
    y_l = coordsArr[0]['y']
    x_h = coordsArr[2]['x']
    y_h = coordsArr[2]['y']

    x_avg = (x_h + x_l)/2.
    y_avg = (y_h + y_l)/2.
    width = abs(x_h - x_l)
    height = abs(y_h - y_l)

    return [label, x_avg/img_width, 1. - y_avg/img_height, width/img_width, height/img_height]

with open('train.txt', 'w') as train_file:
    for i, row in enumerate(df.iterrows()):
        tmpDict = row[1]['Label']
        if(type(tmpDict) == dict):
            image_name = row[1]['External ID']
            train_file.write(path.join(prefix,outputFolder,image_name) + '\n')
            image = path.join(picturesFolder, image_name)
            im = Image.open(image)
            width, height = im.size

            copyfile(path.join(picturesFolder, image_name), path.join(outputFolder, image_name))
            source_img = Image.open(image)
            draw = ImageDraw.Draw(source_img)
            with open(path.join(outputFolder, image_name.split('.')[0] + '.txt'), 'w') as f:
                for box in tmpDict['Nalu']:
                    labelProps = parseBounds(box['geometry'], height, width, 0)
                    x1 = (labelProps[1] - labelProps[3] / 2.) * width
                    y1 = (labelProps[2] - labelProps[4] / 2.) * height
                    x2 = (labelProps[1] + labelProps[3] / 2.) * width
                    y2 = (labelProps[2] + labelProps[4] / 2.) * height
                    draw.rectangle(((x1, y1), (x2, y2)), fill="black")
                    f.write(' '.join(str(x) for x in labelProps) + '\n')
            source_img.save(path.join(checkFolder, image_name), "JPEG")

```

DATA PREPARATION

THEN WE **EXPORTED THE FILE** AS A .JSON, PLUS ALL THE PHOTOS.

AFTERWARDS WE PERFORMED SEVERAL **TRANSFORMATIONS** ON THE RESULTS AND EXPORTED THE DATA TO A **TXT FORMAT** THAT DARKNET CAN INTERPRET.

WE HAD TO **INVERT THE COORDS** AS DARKNET/YOLO READ THE Y AXIS AS STARTING FROM BELOW.

AI TRAINING SERVER

nalu us-east1-c [redacted] (nic0) [redacted] SSH

WE THEN TRAINED THE MODEL USING DARKNET/YOLO ON A SERVER:

8 vCPUs, 30 GB RAM

1 x NVIDIA Tesla K80

WHAT IS DARKNET & YOLO v3?

WATCH THIS TED TALK BY JOSEPH REDMON FOR AN OVERVIEW

DARKNET

DARKNET IS AN OPEN SOURCE **NEURAL NETWORK FRAMEWORK** WRITTEN IN C AND CUDA.

IT IS FAST, EASY TO INSTALL, AND SUPPORTS **CPU AND GPU** COMPUTATION.

YOLO v3

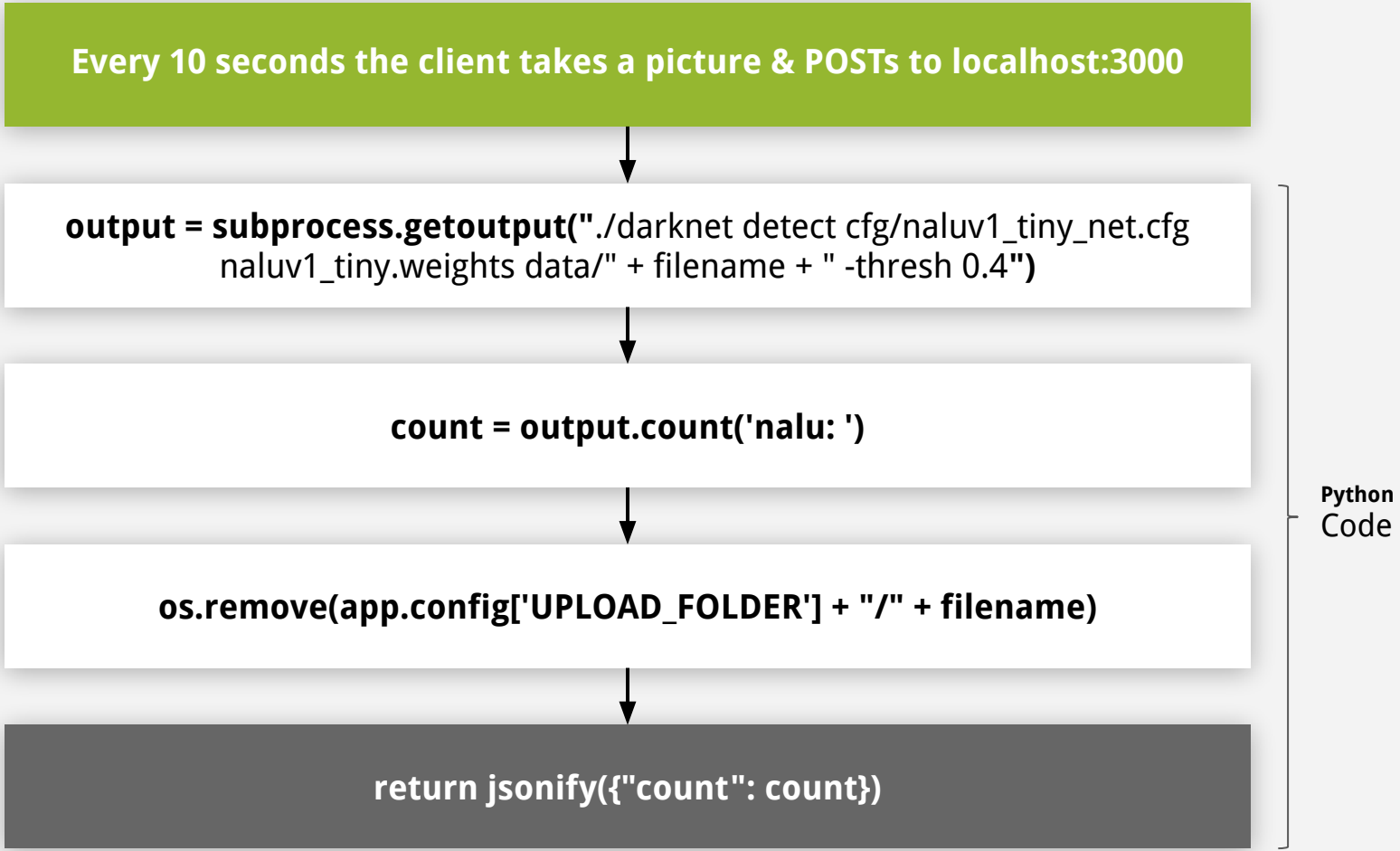
YOU ONLY LOOK ONCE (YOLO) IS A STATE-OF-THE-ART, REAL-TIME **OBJECT DETECTION SYSTEM**.

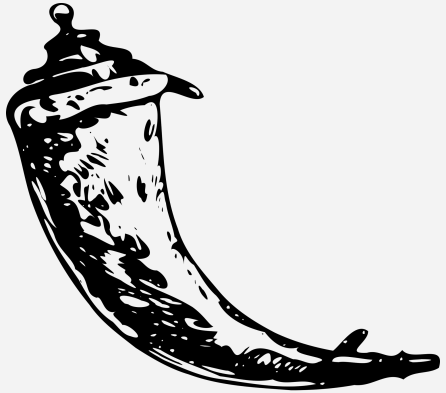
IT TOOK ABOUT **1.5 HOURS** TO TRAIN THE MODEL AND OBTAIN A “WEIGHTS” FILE

```
   7 max          2 x 2 / 2    52 x  52 x 128   ->    26 x  26 x 128
   8 conv    256  3 x 3 / 1    26 x  26 x 128   ->    26 x  26 x 256  0.399 BFLOPs
   9 max          2 x 2 / 2    26 x  26 x 256   ->    13 x  13 x 256
  10 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
  11 max          2 x 2 / 1    13 x  13 x 512   ->    13 x  13 x 512
  12 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
  13 conv    256  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 256  0.089 BFLOPs
  14 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
  15 conv     18  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x  18  0.003 BFLOPs
  16 yolo
  17 route  13
  18 conv    128  1 x 1 / 1    13 x  13 x 256   ->    13 x  13 x 128  0.011 BFLOPs
  19 upsample            2x    13 x  13 x 128   ->    26 x  26 x 128
  20 route  19 8
  21 conv    256  3 x 3 / 1    26 x  26 x 384   ->    26 x  26 x 256  1.196 BFLOPs
  22 conv     18  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x  18  0.006 BFLOPs
  23 yolo
Loading weights from nalu_bu/yolov3-tiny-obj.backup...Done!
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005
Resizing
576
Loaded: 0.000061 seconds
```

STEP 2

WEB SERVER + DETECTION


Every 10 seconds the client takes a picture & POSTs to localhost:3000
output = subprocess.getoutput("./darknet detect cfg/naluv1_tiny_net.cfg naluv1_tiny.weights data/" + filename + " -thresh 0.4")
count = output.count('nalu: ')
os.remove(app.config['UPLOAD_FOLDER'] + "/" + filename)
return jsonify({"count": count})
Python
Code



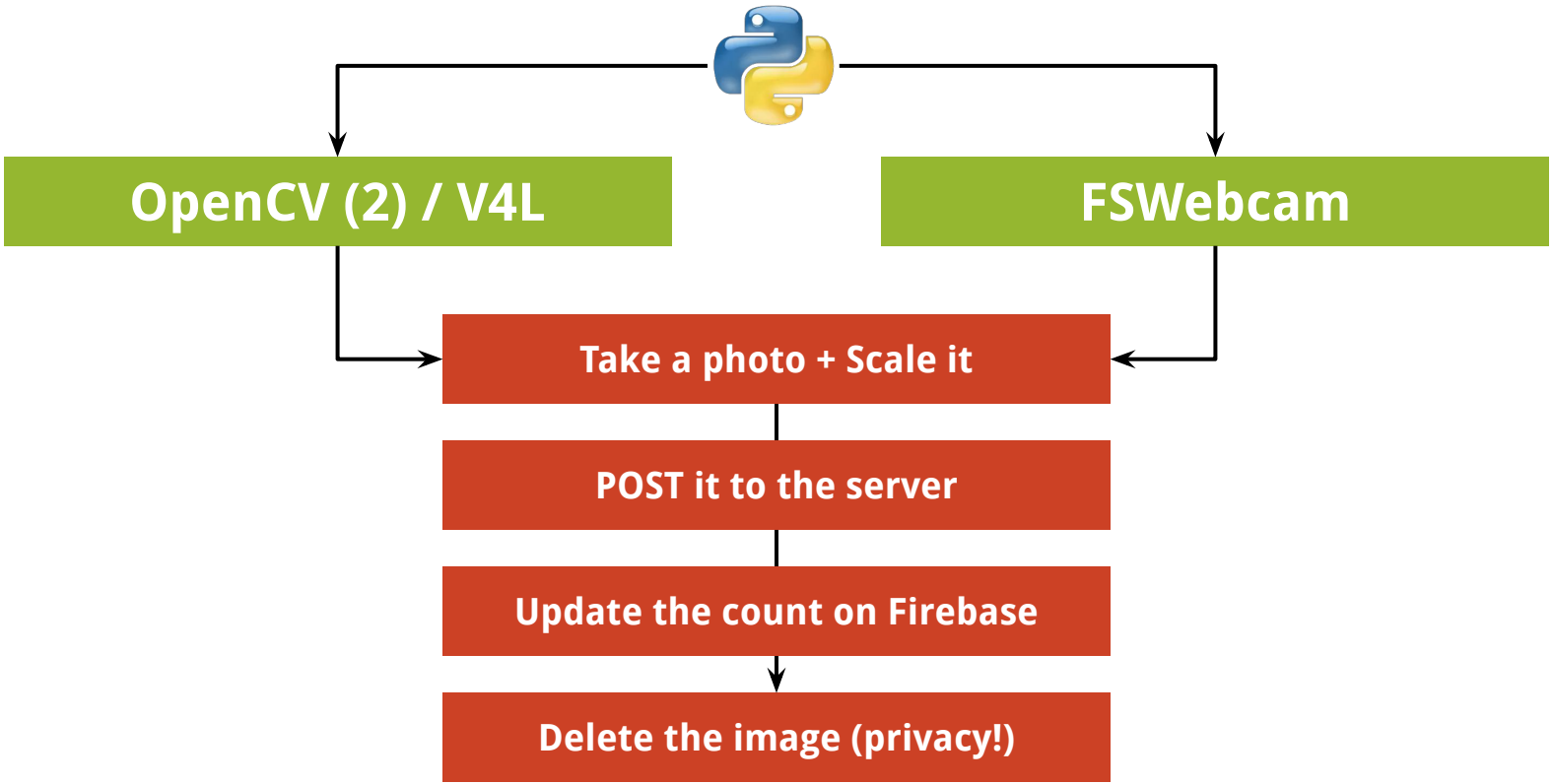
Flask

WE USED A SIMPLE FLASK SERVER (SYNC/BLOCKING)
TO AVOID OVERSATURATING DARKNET

STEP 3

THE CLIENT

IT'S A SIMPLE PYTHON SCRIPT WHICH CAN RUN ON A MAC WITH OPENCV, OR ON LINUX

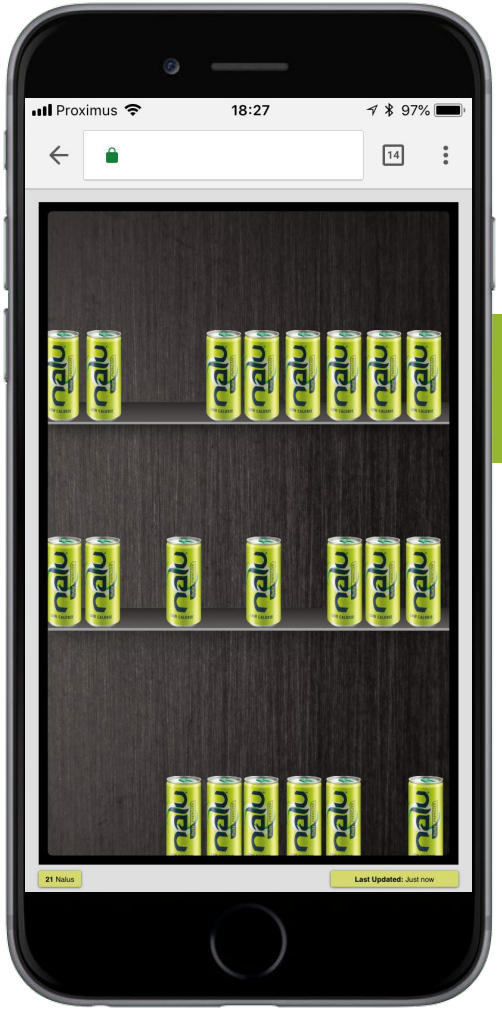



NOT PERFECT, BUT GOOD ENOUGH...

STEP 4

THE FRONTEND

Simple frontend (imitating a fridge) written in one HTML file with a sprinkling of jQuery (for nostalgia)



LOOKS MORE LIKE A FRIDGE ON MOBILE!

SOURCE CODE

https://github.com/lemiffe/nalu-net/

(The source code is crude as this started out as a simple hack to demonstrate it was possible)

RESULTS

We figured out after installing it that the current model does not work well with **Nalus that are over 4-5 meters away**, or if they are behind the glass doors of a refrigerator.

The model can be re-trained to recognise Nalus in more scenarios if needed. It requires more training data, ideally over 500 photos. The model currently has an **accuracy of 70-80%** depending on the resolution, lighting conditions, and the threshold.

THE END!

DRINK 

RESPONSIBLY