pepperbit

julia Bucharest

Intro to Deep Learning with Julia

Bucharest Julia Meetup

9th of October 2019

Traffic Sign Detector

- Task:
  - Create a traffic sign detection library which can be used standalone or interfaced with smartphone Android app
- Constraints:
  - Localize and classify traffic signs in a series of video frames
  - 1fps processing speed
  - Run on server CPU

Traffic Sign Detector Cookbook

- Ingredients:
  - Deep learning library:
    - For defining and training a <u>model</u> and/or additional image <u>processing</u>
  - Beer:
    - For all your friends who look funny on the street taking pictures of traffic signs for the <u>dataset</u>

Traffic Sign Cookbook

- Ingredients:
  - Deep learning library:
    - for defining and training a model and/or additional processing: Knet.jl, Flux.jl
  - Dataset:
    - German Traffic Sign Dataset, cleaned and updated with Romanian traffic signs -> 60 classes and counting
  - WebApp/Server library:
    - Genie.jl
  - Database library:
    - SQLite.jl
  - Server (hardware)
  - Android app:
    - For taking pictures and sending them to the server for storage

Deep Learning Library

- Knet.jl:
  - Implemented in Julia (so is Flux.jl)
  - Straightforward tutorials backed in the package itself
  - Good GPU integration
  - Good memory management

Dataset

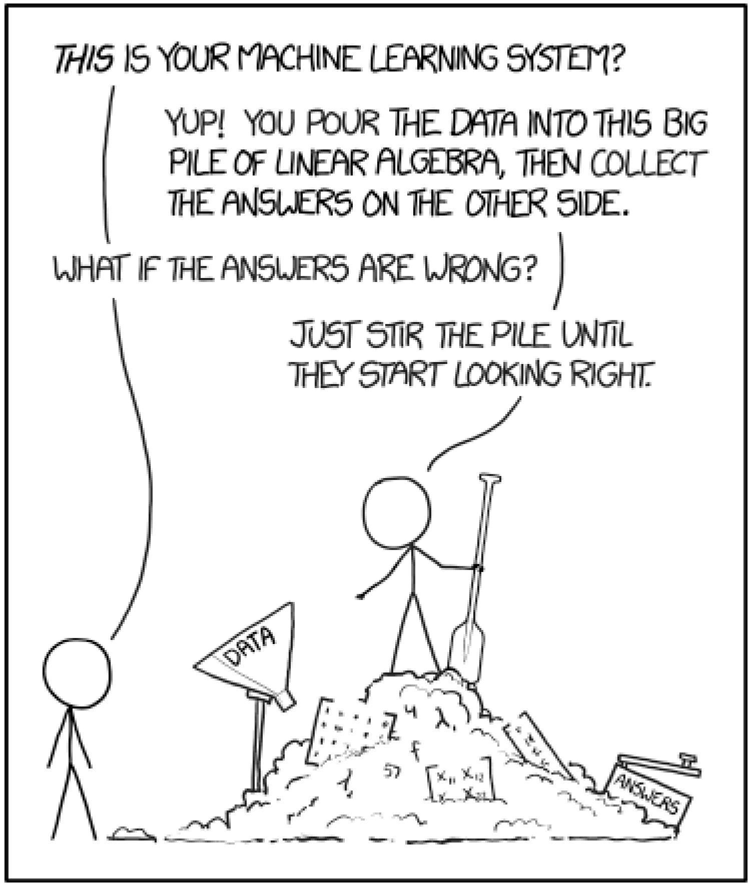
- German Traffic Sign Dataset:
  - Advantages:
    - ~40 classes
  - Disadvantages:
    - Some signs are different from the Romanian ones
    - Some important signs are missing
    - Blurred images
    - Dark images
    - Low resolution images
- Fix the shortcomings:
  - Gather more pictures!
  - Apply data augmentation

WebApp/Server Library

- Genie.jl:
  - Implemented in Julia
  - Used for running a BE server which:
    - Receives images from the Phone App
    - Stores images
    - Stores additional data about images in SQLite database
    - <u>Processes images in order to detect traffic signs</u>

Choosing an architecture…

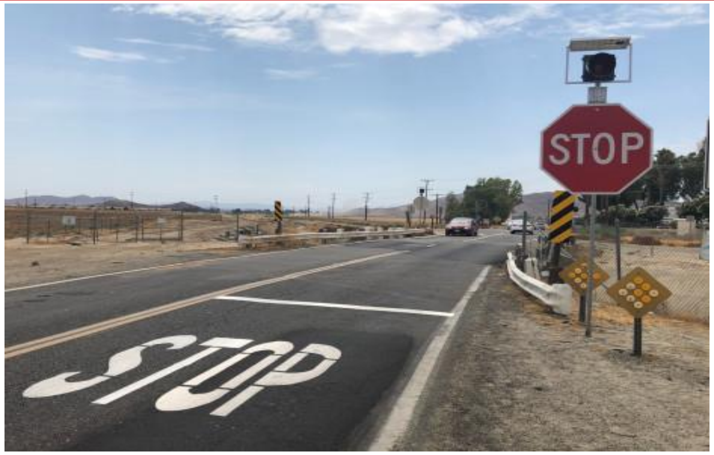
- R-CNN
- Faster R-CNN
- YOLO
- Single Shot Detector


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

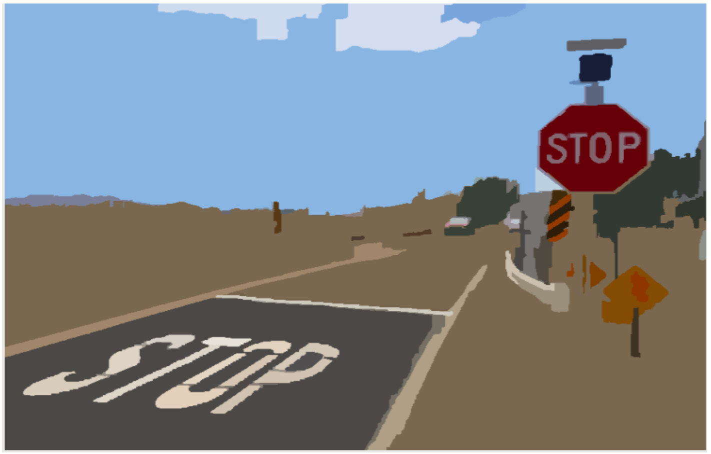
R-CNN

- aka: Region Proposal Conv. Nets
- Two stage processing:
  - Classical image processing:
    - Region Proposal (ROI) by applying image segmentation
  - DL classifier :
    - applied on the ROIs

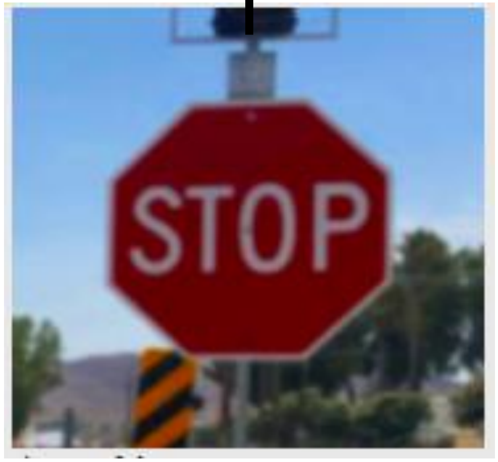
Processing steps



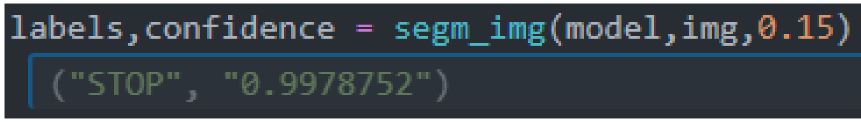
Segmentation



Crop & process



Apply model



```
labels,confidence = segm_img(model,img,0.15)
("STOP", "0.9978752")
```

Label & confidence level

Conclusions

- What we achieved (so far):
  - Server: running
  - Dataset: growing
  - Database: in place
  - Image segmentation and region proposal: done
  - DL model: defined from scratch and trained
  - * Android app: functional (used it to send images to server)

- What needs improvement:
  - Better image preprocessing to accommodate different lighting conditions
  - Optimize ROI selection and cropping speed
  - Try other architectures: R-CNN, YOLO (work in progress)

Q&A