Real-Time Object Detection

Midpoint Spotlight

Xiaolan Cai, Warren Ferrell
Chu-Sheng Ku, Soham Shah, and Ryan Skinner

Theory Progress

Encompasses network architecture, training, and post-processing output, and finding ways to speed up the algorithm

- Read papers and audited Coursera course on CNNs/YOLO; now have good understanding of network structure and post-processing
- Learned to use Keras with simple fully-connected networks
- Set up CUDA for a GTX 960 to test how training scales with size of YOLO network
- Set up AWS account for full-scale training on 1-16 Tesla K80 GPUs in parallel, targeting spot instances
- Identified ways to increase speed of network evaluation
    1. Smaller input image resolution (likely much faster but accuracy may suffer, since small features are important for detection [1]).
    2. Replace FC layers between CNN layers and output tensor with a ConvDet layer [2] (likely faster without much loss in accuracy)
    3. Reduce the number of object classes (likely a little bit faster without losing accuracy)

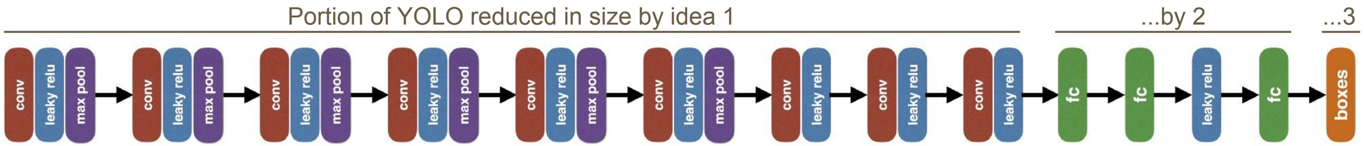

Figure 1. Network diagram of YOLO, with the layers affected by each of our proposed cost-saving measures listed above.

Hardware Progress

Encompasses implementation on Raspberry Pi, hardware acceleration with Pi GPU, and real-time webcam interface

- Installed Darknet on RPi and tested YOLO v1,2,3 and Tiny-YOLO v1,2,3 [3]

Model	**Classification Time (sec)**	**Correct Results?**
YOLOv2	156	Yes
Tiny-YOLO	38	No

- Used NNPACK [4] to accelerate neural network computations

Model	**Classification Time (sec)**	**Correct Results?**
Tiny-YOLO	1.245	Yes

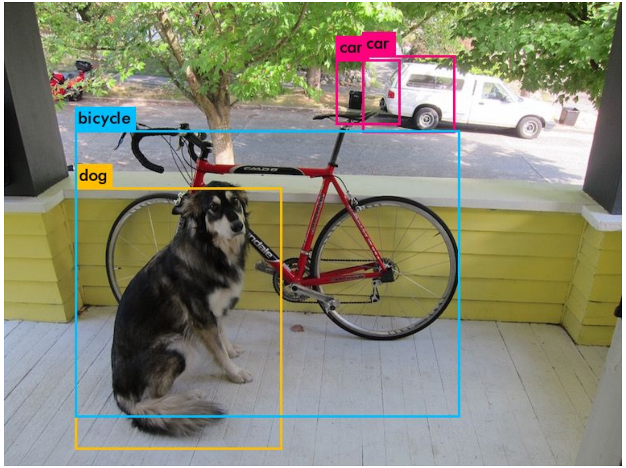

Figure 2. Detection result on sample image, run through Tiny-YOLO

Future Work / Challenges

Future work

- Build YOLO architecture in Keras
- Perform scaling studies on GTX 960, then scale up to AWS GPU-accelerated instances for training
- Explore small off-the-shelf CNNs we can use as pre-trained heads for the YOLO architecture
- Interface with Pi camera to perform real(ish)-time object detection
- Modify YOLO as outlined on Theory Progress slide to decrease detection latency

Challenges / seeking feedback

- Mostly seeking input on contingency plan if real-time (<1 fps) on the Raspberry Pi is infeasible.
- An alternative would be to run YOLO on a more powerful machine, and stream images and object detections back and forth between the RPi client and the YOLO server.
- Any other ideas? Communication protocol advice for above idea (bluetooth, WiFi, etc.)?

Works Cited

[1] Redmon et al., *You Only Look Once: Unified, Real-Time Object Detection*. 2016, v5. https://arxiv.org/abs/1506.02640

[2] Wu et al., *SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving*. 2017. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8014794&tag=1

[3] YOLO official website. https://pjreddie.com/darknet/yolo/

[4] YOLO with NNPACK. https://github.com/digitalbrain79/darknet-nnpack