Recognition STOP and GO Operations by Autonomous Automobile IARA

Eduardo Frigini, Deivison Vitória

***Abstract*— We present in this work an algorithm for the Intelligent Autonomous Robotic Automobile (IARA) to be able to identify STOP and GO operations during a one-way route. IARA is a fully autonomous car that uses object detection to identify obstacles. We use YOLOv3 which is a state-of-the-art real-time object detection system. We use a database of STOP and SIGA signaling images to train the network and we have achieved good results in detecting these objects.**

***Index Terms*—STOP, GO, YOLO, detection.**

I. Introduction

Since your introduction by LeCun in the early 1990's, Convolutional Networks (convnets) have demonstrated excellent performance at tasks such as hand-written digit classification and face detection.

Most notably, (Krizhevsky et al., 2012) show record beating performance on the ImageNet 2012 classification benchmark, with their convnet model achieving an error rate of 16.4% (Supervision Network / AlexNet), compared to the 2nd place result of 26.1% (ISI Network).

Several factors are responsible for this renewed interest in convnet models. First the availability of much larger training sets, with millions of labeled examples. Second, powerful GPU implementations and lastly, better model regularization strategies, such as Dropout (Hinton et al., 2012).

Ever since then, the companies have been using deep learning at the core of their services. Facebook, for exemplo, uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, the Instagram for their search infrastructure, among many other applications.

II. Literature Review

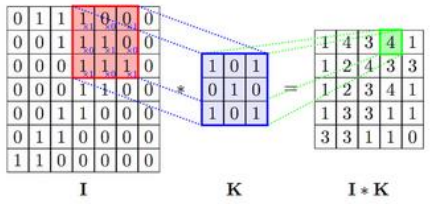
A. Convolutions

The convolutions is a very efficient way of pulling this off, and it makes advantage of the structure of the information encoded within an image. It is assumed that pixels that are spatially closer together will "cooperate" on forming a particular feature of interest much more than ones on opposite corners of the image. Besides, if a particular (smaller) feature is found to be of great importance when defining an image's label, it will be equally important if this feature was found anywhere within the image, regardless of location.

Given a two-dimensional image, I, and a small matrix, K of size h×w, (known as a convolution kernel), which we assume encodes a way interesting of a extracting image feature. Then we compute the convolved image, I∗K, by overlaying the kernel on top of the image in all possible ways, and recording the sum of elementwise products between the image and the kernel:

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} \cdot I_{x+i-1,y+j-1}$$

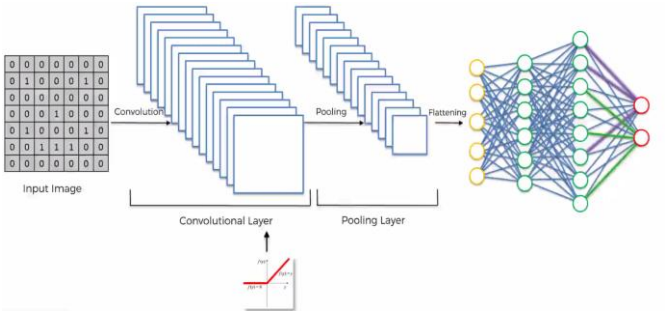
The images below show a diagrammatical overview of the above formula and the result of applying convolution (with two separate kernels) over an image, to act as an edge detector:



Picture I. Convolutions

B. Common CNN

A typical CNN architecture for a k class image classification can be split into two distinct parts. First a chain of repeating convolutional and pool layers (sometimes with more than one convolutional layer at once), and second followed by a few fully connected layers (taking each pixel of the computed images as an independent input), culminating in a k-way softmax layer, to which a cross-entropy loss is optimized. It is important to remember that after every convolutional or fully connected layer, an activation (e.g. ReLU) will be applied to all of the outputs.



Picture II. Example of Full Convolutional Neural Network

Note the effect of a single convolution and pooling pass through the image, it reduces height and width of the individual channels in favors of their number, i.e., depth.

A softmax layer's purpose is converting any vector of real numbers into a vector of *probabilities* (nonnegative real values that add up to 1). Within this context, the probabilities correspond to the likelihoods that an input image is a member of a particular class. Minimizing the cross-entropy loss has the effect of maximizing the model's confidence in the correct class, without being concerned for the probabilities for other classes. This makes it a more suitable choice for probabilistic tasks compared to, for example, the squared error loss.

C. Image Processing

Image classification is the task of taking an input image and outputting a class (cat, dog, etc.) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is comes naturally and effortlessly when adults. Without even thinking twice, we are able to quickly and seamlessly identify the environment we are in as well as the objects that surround us. When we see an image or just when we look at the world around us, most of the time we are able to immediately characterize the scene and give each object a label, all done automatically by the brain. That skill of being able to quickly recognize patterns, generalization from prior knowledge, and adapt to different image environments are ours and we not share with machines.

Picture III. What Human See vs. what computers see

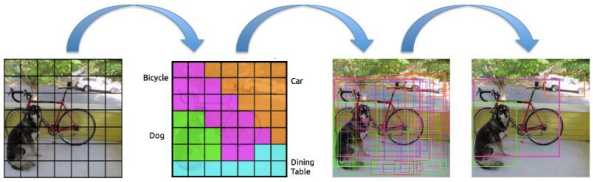
When a computer sees an image (takes an image as input), it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers (the 3 refers to RGB values). For exemplo, let us say we have a color image in JPG form and its size is 480 x 480. The representative array will be 480 x 480 x 3. Each of these numbers is given a value from 0 to 255, which describes the pixel intensity at that point. These numbers, while meaningless to us when we perform image classification, are the only inputs available to the computer. The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class (0.98 for GO and 0.02 for STOP).

Now that we know the problem, as well as the inputs and outputs, let us think about how to approach this. What we want the computer to do is to be able to differentiate between all the images that are given and figure out the specifiques features that make a dog a dog or that make a cat a cat. This process is done on in our minds subconsciously as well. When we look at a picture of a dog, we can classify it as such if the picture has identifiable features such as paws or four legs. In a similar way, the computer is able perform image classification by looking for low level features such as edges and curves, and then building up to more abstract concepts through a series of convolutional layers. This is a general overview of what a CNN does.

D. YOLO detection system

The YOLO (You only look once) is an object detection system that uses a different approach to most systems for this purpose: it applies a neural network to every image once. This network returns another picture with boxes around possible objects known to this already trained network. And the set of these boxes reinforces the certainty in the detection of these objects, because the greater the thickness of the box, the greater the precision in the detection. To use this system, Joseph Redmon created an implementation of this study, called darknet: easy to install, run, and written in C and CUDA.

The YOLOv3 predicts an objectivity score for each boundary box using logistic regression. This should be 1 if the bounding box before overlaps a true earth object by more than any other previous bounding box. If the previous bounding box is not the best, but it overlaps a real object of more than a few thresholds. Thus the system assigns only one bounding box before each basic truth object. If a previous bounding box is not assigned to an object of truth, it does not incur loss for coordinate or class predictions, only objectivity.



Picture IV. Identification of objects detected by Darknet

E. Autonomous Automobile IARA

An autonomous vehicle is a system composed of a hardware platform and a collection of software modules responsible for tasks that include: location, mapping, simultaneous localization and mapping (SLAM), planning paths, motion planning, light detection status and high-level decision making. There is a car developed at the Federal University of Espirito Santo (UFES) that is fully autonomous called Intelligent Autonomous Robotic Car (IARA). A Ford Escape Hybrid was used as a hardware platform and a complete set of software modules were designed and built to enable its autonomous operation in urban traffic.

There are several methods in the literature to address the problem of road movement planning for autonomous cars. Bautista shows a review of these methods. Among those that have been experimentally evaluated using autonomous cars from the real world, road planning methods mainly employ state structure, fast tree randomization (RRT), interpolation, optimization, and predictive modeling techniques.

According to Guidolini, in state-based methods, trajectories between the initial and desired states are sought in a state lattice adapted for motion planning and only states, a priori, that the solution is likely to be represented. The possible trajectories are evaluated by a cost-function that considers dynamics, environment and behavioral restrictions, among others. The disadvantage of these methods is that they are computationally heavy, due to the evaluation of each possible solution in the graph.

The motion planner most similar to IARA is that proposed by Ferguson. However, this work differs from that of Ferguson in three main respects. First, the planner is able to calculate more complex trajectories, since it uses an angle slot of an extra node point (four in total) to parameterize the control car, while Ferguson uses a three-bend node. Second, the IARA planner can generate trajectories optimized for winding roads with obstacles, since its cost function considers the desired road and obstacles, while Ferguson generates various alternative trajectories and can select a collision. Third, it presents the algorithm that was used to calculate the choices table.

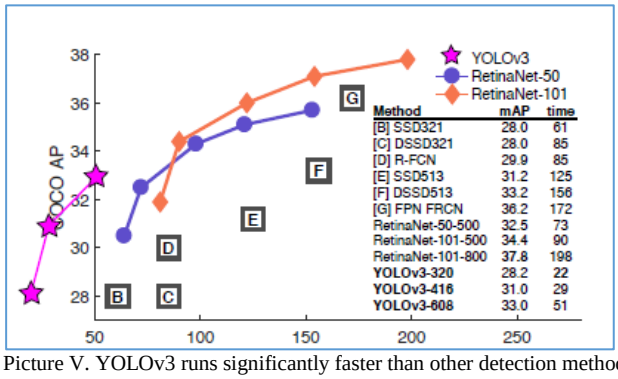
Berriel proposed a system that works on a temporal sequence of images applying a reverse perspective mapping and a particle filter for each image to track the track over time. The system generates a cubic spline for each of the right and left tracks (on the car track), in addition to each track class. Recently, Convolutional Neural Networks (CNNs) have outperformed image processing more accurately than other methods. Lee points out that he built a 20,000 data set with CNN-trained and labeled tracks to detect and track all of the tracks and road markings on the image in a pixel segmentation. The system generates a trust map that detects and sorts the tracks in the image.

These methods, however, are only used in the presence of road markings, do not produce a road map and have not been tested on a real standalone car. Moreover, the method presented by Lee is strongly parameter dependent.

III. METHODS

A. YOLOv3 Net

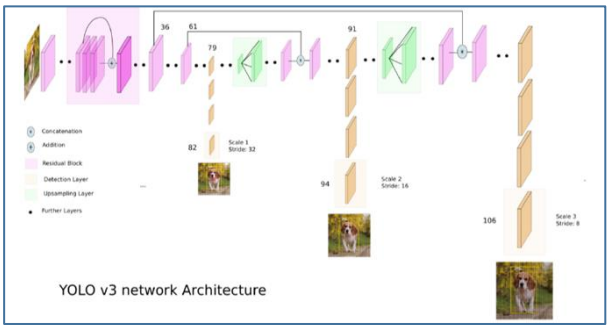
YOLOv3 is extremely fast and accurate. In MAP measured at 0.5 IOU, YOLOv3 is at the same level as Focal Loss, but about 4 times faster. Also, you can easily switch between speed and accuracy simply by changing the model size, without training.



Picture V. YOLOv3 runs significantly faster than other detection methods

Pre-detection systems redirect classifiers or locators to perform the detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered to be detections. YOLOv3 applies a totally different approach. Uses a single neural network to the complete image. This network divides the image into regions and provides bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

The model has several advantages over classifier-based systems. It looks at the entire image in test time so that its predictions are informed by the overall context in the image. It also makes predictions with a single network assessment, unlike systems like R-CNN, which require thousands for a single image. This makes it extremely fast, more than 1000 times faster than R-CNN and 100 times faster than Fast R-CNN.



Picture VI. Network Architecture YOLOv3

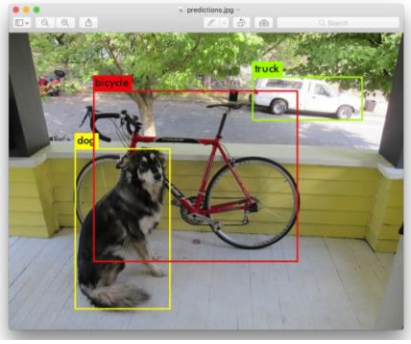
YOLOv3 uses some tricks to improve training and increase performance, including: multi-level predictions, a better backbone classifier among other things.

B. Detection Using A Pre-Trained Model

We run these commands on the Linux terminal:
git clone https://github.com/pjreddie/darknet
cd darknet
make

Then we load the weights:
wget https://pjreddie.com/media/files/yolov3.weights
They were in the subdirectory of Darkenet cfg/

Then we executed the darknet and we tried to find the dog in the image, using yolov3
./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg



Picture VII. Dog image detection

The above result found by YOLOv3 was: dog: 99%
truck: 93% and bicycle: 99%.

We tested YOLOv3 using the web cam, running the command:
./darknet detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights

We don't use the VOC part.
Then we went to the Training YOLO on COCO (http://cocodataset.org/#overview)
COCO is a dataset of images marked with acronym for Common Objects in Context. In order to train YOLO you need the images and the labels (markings), by executing these commands you can load the images and the markings of the COCO:
cp scripts/get_coco_dataset.sh data
cd data
bash get_coco_dataset.sh

After loading the COCO base we changed a file cfg/coco.data.
We created a file called coco_cps.data (cps is cone, stop and follow). We did not change the COCO files, we actually created others to use in training later. This is what is in the coco_cps.data file:

classes= 3
train=/home/edufrigini/Desktop/darknet/data/coco/ trainvalno5k _cps.txt
valid = /home/edufrigini/Desktop/darknet/data/coco/5k_cps.txt
eval = /home/edufrigini/Desktop/darknet/data/coco/5k_cps.txt
names = data/coco_cps.names
backup = backup

There are 3 classes and for training points to the file trainvalno5k_cps.txt where you have the list of names of all the images for training.
The 5k_cps.txt file is all images for testing.
The file coco_cps.names are the names of the three classes
Cone
Stop
Go
The focus of this work was to identify the STOP and GO.

Then we copied the yolov3.cfg file and yolov3_cps.cfg and changed some parameters.
We put the batch for 128 and subdivisions = 16
We changed these parameters
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky
mask = 6,7,8
anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

Then we changed the filters from 255 to 24.
After that we train the network using the GPU with images of Cones, Stop and GO boards, using this command
./darknet detector train cfg/coco_cps.data cfg/yolov3_cps.cfg darknet53.conv.74 -gpus 0

After 5 days of training the network had done 80 thousand iterations and we used it for the tests.

C. Applying image classification

We use the same file coco_cps.data. We made a copy of yolov3_cps.cfg for yolov3_cps_teste.cfg and changed the beginning of the file. We just put the batch equal to 1 and the subdivisions for 1 also, as the script below:
[net]
# Testing
batch=1
subdivisions=1
# Training
# batch=128
# subdivisions=16

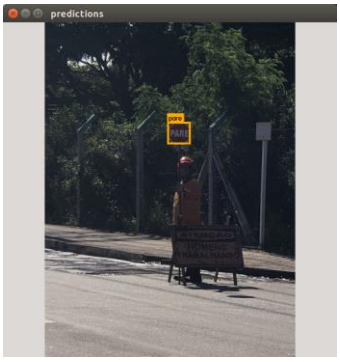
```
width=320
height=320
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
```

To test the network:
./darknet detector test cfg/coco_cps.data cfg/yolov3_cps_teste.cfg yolov3_cps_80000.weights

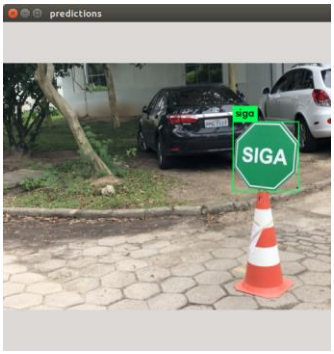
Command result

```
99 conv  128  1 x 1 / 1  40 x  40 x 384 >  40 x  40 x 128  0.157 BFLOPs
  100 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256 0.944 BFLOPs
  101 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128 0.105 BFLOPs
  102 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256 0.944 BFLOPs
  103 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128 0.105 BFLOPs
  104 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256 0.944 BFLOPs
  105 conv     24  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x  24  0.020 BFLOPs
  106 yolo
Loading weights from yolov3_cps_80000.weights...Done!
Enter Image Path:
/home/edufrigini/Desktop/darknet/data/coco/images/teste_cps/IMG_1681.JPG
```

D. Result of the test done in the network trained with the images of STOP and GO



Picture VIII. STOP image detection



Picture IV. GO image detection

To mark the photos we made a script in C.

IV. CONCLUSION AND FUTURE WORK

The work focused on developing a neural convolutional network that allows the autonomous vehicle IARA to detect STOP and GO operations during a route on a route. For this, the YOLOv3 network was used to train and validate the network.

The result was positive as we were able to verify that the network is able to detect STOP and GO boards perfectly, causing the target to be reached.

As a future work, we can create a dataset with more STOP and SIGA images in different scenarios. So we can train and validate the network more comprehensively.

V. REFERENCES

[1] HINTON, G. E., Osindero, S., and The, Y. A fast learning algorithm for deep belief nets. Neural Computation, 18:1527{1554, 2006.

[2] HINTON, G.E., Srivastave, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012.

[3] KRIZHEVSKY, A., Sutskever, I., and Hinton, G.E. Imagenet classication with deep convolutional neural networks. In NIPS, 2012.

[4] LECUN, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. Neural Comput., 1(4):541{551, 1989.

[5] SOHN, K., Jung, D., Lee, H., and Hero III, A. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In ICCV, 2011.

[6] D. G. Bautista, J. Pérez, V. Milanés and F. Nashashibi, “A Review of Motion Planning Techniques for Automated Vehicles”, IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135 – 1145, 2015.

[7] D. Ferguson, T. Howard and M. Likhachev, “Motion Planning in Urban Environments”, Journal of Field Robotics, vol. 25, no. 11–12, pp. 939–960, 2008.

[8] R. F. Berriel, F. S. Rossi, A. F. de Souza, and T. Oliveira-Santos, “Automatic large-scale data acquisition via crowdsourcing for crosswalk classification: A deep learning approach,” Computers & Graphics, vol. 68, pp. 32–42, 2017.

[9] S. Lee et al., “VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition,” In IEEE International Conference on Computer Vision, pp. 1965-1973, 2017.

[10] R. Guidolini, C. Badue, M. Berger, A. F. De Souza, "A Simple Yet Effective Obstacle Avoider For The IARA Autonomous Car", 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), 2016.

[11] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv, 2018.