Traffic Sign Detection with YOLO Framework

Jaehyun Yoon, Renea Young, Andrew Tran

Background

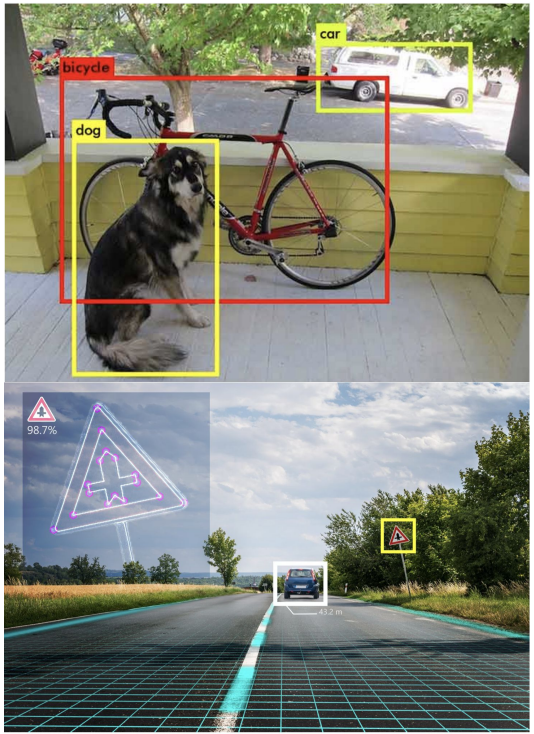
Object detection: is a computer vision technique that identifies and locates objects in an image or a video

Object detection can be applied in a number of ways, i.e. crowd counting, video surveillance, face recognition, self-driving cars, etc.

Traffic sign detection is a challenging task in which the algorithms have to cope with natural and complex environments, high accuracy demands, and real-time constraints.

- Real-world applications includes autonomous driving, traffic surveillance and driver safety and assistance

Many approaches for traffic sign detection have been proposed.



Current State of the Art

Deep neural network methods have become the state of the art approach to traffic sign detection.

- Faster Region-based CNN: similar to Fast Region-based CNN however it uses the Region Proposal Network.
- Region-based Fully Convolutional Network: is a region-based detector.
- Single Shot Detector: is a method for detecting objects in images using a single deep neural network
- You Only Look Once: process images in real time. Has been recognized for fast object detection

What is YOLO?

- “You Only Look Once” was developed by Joseph Redmon, first published in 2015
- is an object detector that uses features learned by deep convolutional neural network to detect objects
- YOLOv1, YOLOv2 and YOLOv3
- YOLOv3
    - Has 106 layers. Uses Darknet-53 as its backbone

The entire process of detection can be summed up as:

1. Intake an image and split it into an SxS grid
2. Each pixel in the image can be responsible for a finite number of bounding boxes predictions
3. The result is a large number of bounding boxes
4. Bounding boxes are consolidated into a final prediction by a post processing step.

Model Framework

- The most salient features of YOLOv3 is that it makes detections at three different scale.
- **YOLOv3 predicts boxes at 3 scales and 3 boxes at each scale**
  -> in total 9 boxes
- For the objectness score, we need to calculate the IoU (= Area of Overlap / Area of Union) of a bounding box * Pr(object)
- The tensor is N * N (B(=3) * (Box Co-ordinates + Objectness score + Class scores))
    - For example, the first feature map has the size of 13*13. The tensor is 13*13 (3 * (4 + 1 + 43(= the # of classes))).

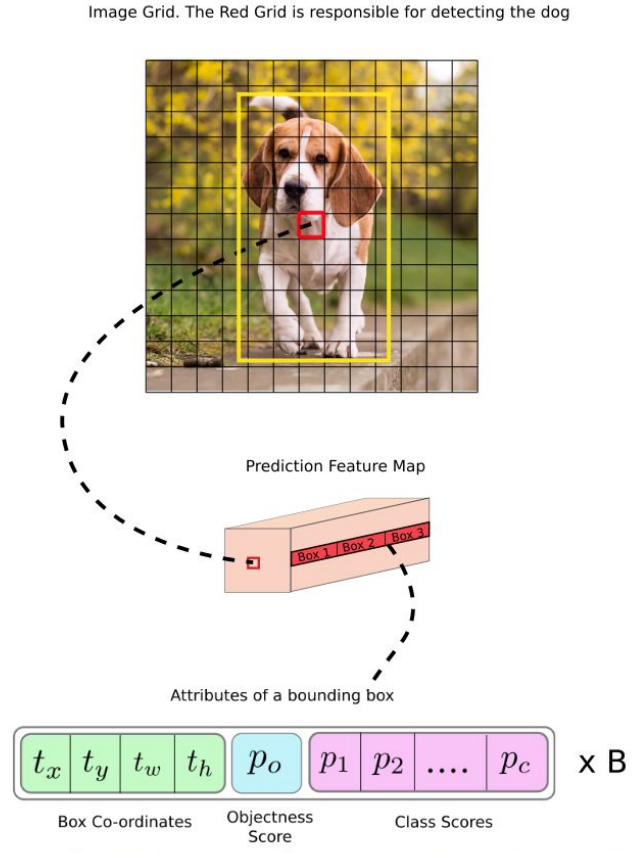
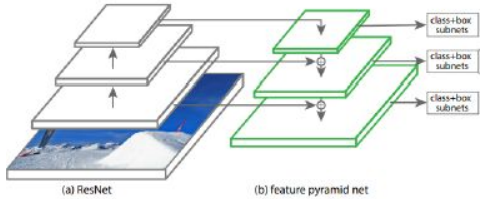
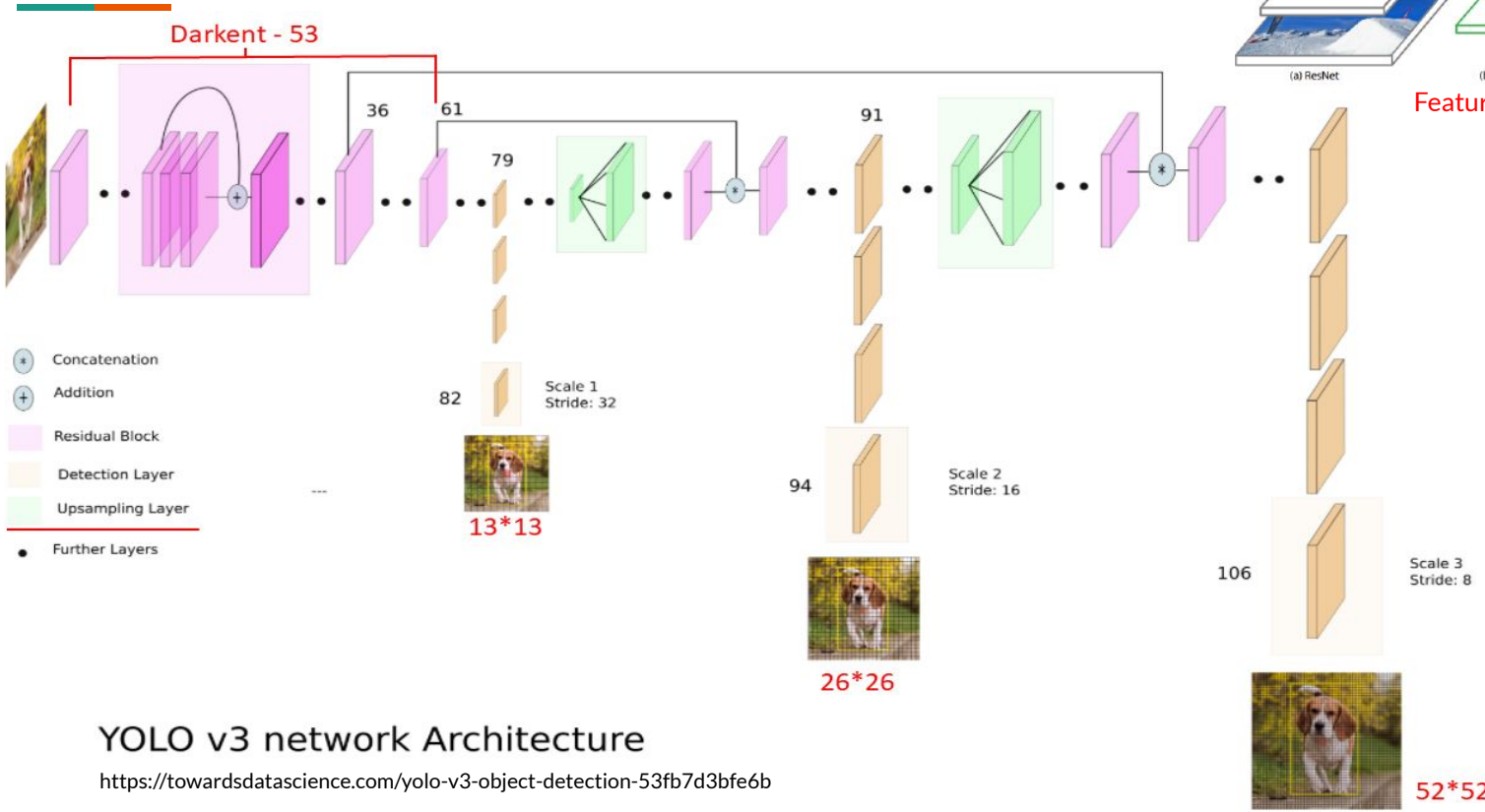


Image credits: https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

Model Framework



Feature Pyramid network



YOLO v3 network Architecture

https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

Feature Extraction & Multi labels

- Feature extraction - Darknet-53
  - “Darknet-53 is better than ResNet-101 and 1.5x faster. **Darknet-53 has similar performance to ResNet-152 and is 2x faster**.” (YOLOv3: An Incremental Improvement)
  - “Darknet-53 achieves **the highest measured floating point operations per second**. This means the network structure better utilizes the GPU, making it more efficient to evaluate and thus faster.” (YOLOv3: An Incremental Improvement)
- Multi labels prediction
  - YOLOv3 uses the multi-label prediction of individual objects as a sigmoid-based logistic classifier instead of Softmax.

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

The Data

- German Traffic Sign Detection Benchmark (“GTSDB”) and German Traffic Sign Recognition Benchmark (“GTSRB”) datasets
- 43 classes of traffic signs
- 900 images from the GTSDB set and over 50,000 images from the GTSRB set
- GTSDB set works well for detection experiments, while GTSRB set works well for classification experiments

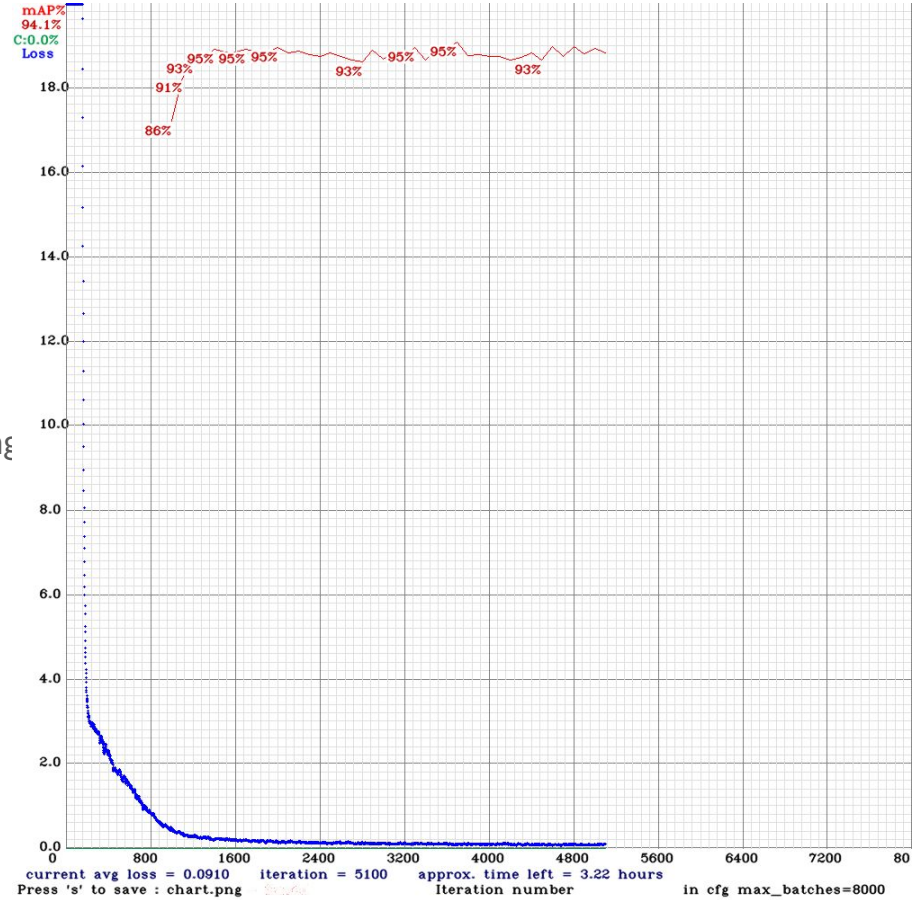
https://benchmark.ini.rub.de/index.html

YOLO Implementation

- Data preparation
  - YOLO format = class id, center X, center Y , height, and width
  - Configuration file
    - optimize the number of batch, max batch, the number of classes, and filters
    - Darknet recommends setting the max_batches = (# of classes) * 2000 for the number of epochs
  - Darknet
- Backup
  - To save the best weights and the weights for every 1000 epoch in case of getting an error and being stopped during the training
- Train
  - Feed all information with the pre-trained weights from Darknet to train model

Results

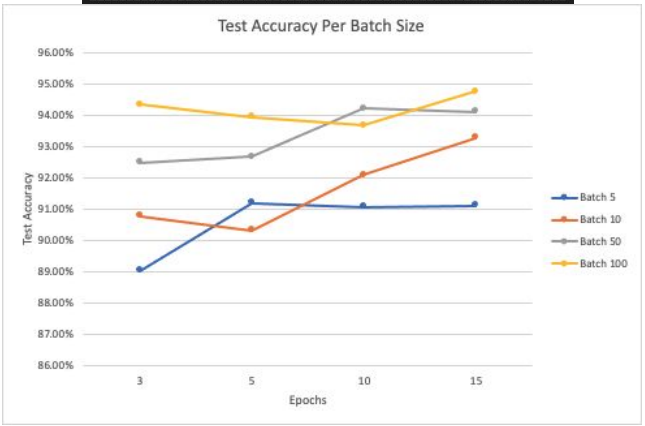
- The loss value quickly drop at about 400 epoch
- Stop at 5000 epoch, no reason to keep training
- Find an early stopping point to avoid overfitting



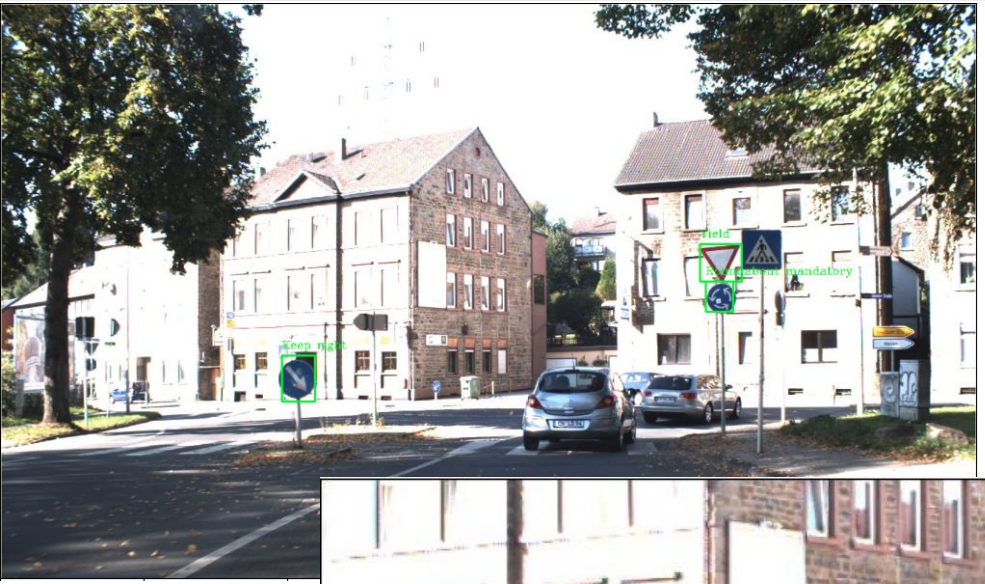
Classification Implementation

- Preprocessing training data
  - Converting source files from .pickle and redimensioning
  - Scaling values tensor values to range of 0-255 for pixels
- Began with AlexNet but gradually reduced the number of layers to improve accuracy
- Trained and tested on GTSRB dataset with 43 classes
- Tested different batch and epochs to find maximum accuracy
- Predicted classes on bounding box slices found from YOLO implementation

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 32, 32, 32)        2432
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 96)        76896
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 96)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 6144)              0
_________________________________________________________________
dense_9 (Dense)              (None, 1000)              6145000
_________________________________________________________________
dropout_5 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_10 (Dense)             (None, 43)                43043
=================================================================
Total params: 6,267,371
Trainable params: 6,267,371
Non-trainable params: 0
_________________________________________________________________
```



Results - Image Detection


Field
Roundabout mandatory
Keep right


Field
Roundabout mandatory
Keep right

Results - Video Detection

Novelty

- YOLO framework is constantly improved for speed and accuracy finding bounding boxes and classification
- Speed can decrease dramatically with more classes
- Using a separate classification model can reduce overall training time
- Better fitting the data for classification by slicing original image

References

- J. Redmon and A. Farhadi. “YOLOv3: An incremental improvement.” Volume 4, *arXiv*, 2018.
- Gupta, Mehul. “All about YOLO Object Detection and Its 3 Versions (Paper Summary and Codes!!).” *Medium*, Data Science in Your Pocket, 20 Apr. 2020, medium.com/data-science-in-your-pocket/all-about-yolo-object-detection-and-its-3-versions-paper-summary-and-codes-2742d24f56e.
- V. Meel. "YOLOv3: Real-Time Object Detection Algorithm (What's New?) | viso.ai." *Computer Vision Application Platform | viso.ai.* Viso.ai, 25 Feb 2021. Web. 28 Jul 2021. <http://viso.ai/deep-learning/yolov3-overview/>.
- Real-Time Computer Vision. “Welcome to the INI Benchmark Website.” German Traffic Sign Benchmarks, 16 Sep. 2010, https://benchmark.ini.rub.de/.
- Brownlee, Jason. “How to Perform Object Detection with YOLOv3 in Keras.” Machine Learning Mastery, 27, May 2019, https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/.