Deep Learning-based Obstacle Detection and Depth Estimation

Yi-Yu Hsieh[1], Wei-Yu Lin[1], Dong-Lin Li[2], and Jen-Hui Chuang[3]

[1]Institute of Computer Science and Engineering, [2]Computer Vision Research Center, [3]Department of Computer Science

National Chiao Tung University, Taiwan

Sept. 24, 2019

Outline

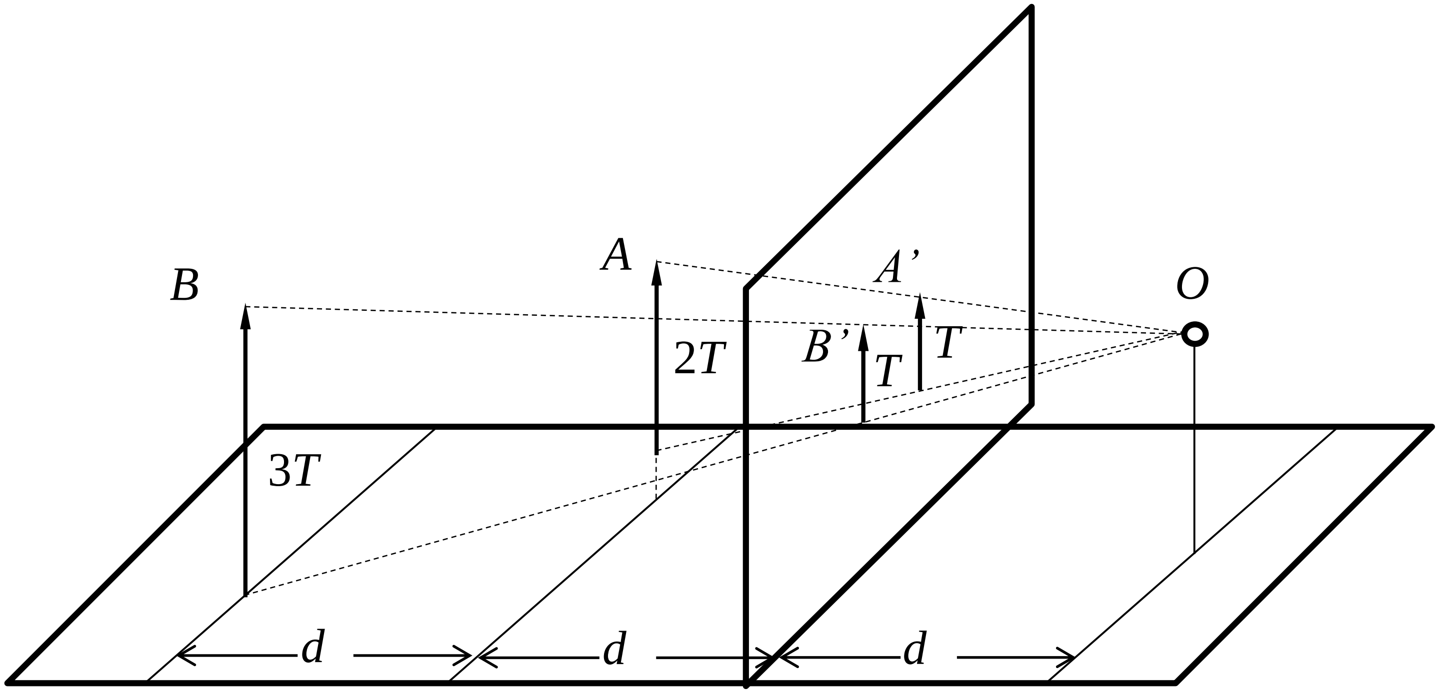
- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based On YOLO
- Experimental Results
- Conclusion

Outline

- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based On YOLO
- Experimental Results
- Conclusion

Introduction – Motivation

- Obstacle detection is a crucial issue in robotics and autonomous driving systems
- Because of perspective projection, obstacle depth information is lost



Introduction – Related Works

- To achieve obstacle avoidance, we need
  - Object detection
  - Depth prediction

- Object detection methods
  - Traditional methods
    - HOG + SVM
    - DPM
  - Deep learning-based methods
    - Fast / Faster R-CNN
    - SSD / R-FCN / FPN FRCN
    - ✓ YOLOv2 (2016) / YOLOv3 (2018)

- Depth prediction methods
  - Traditional methods
    - Stereo matching
  - Deep learning-based methods (monocular)
    - FCRN
    - Godard *et al*., CVPR, 2017
    - Kuznietsov *et al*., CVPR, 2017

  → Too slow (10fps↓ on TITAN)

➢ Self-designed real-time architecture using YOLOv2/3

Introduction – Related Works

Method/CNN model	COCO test mAP	Speed (fps)
Fast R-CNN	39.3	0.5
Faster R-CNN	41.5	7.0
R-FCN	51.9	12
RetinaNet	57.5	5.1
FPN FRCN	59.1	5.8
SSD 300x300	41.2	46
YOLOv2 416x416	44.0	67
SSD 500x500	46.5	19
YOLOv2 608x608	48.1	40
YOLOv3 416x416	55.3	35
YOLOv3 608x608	57.9	20

1st
2nd

Real-time

Outline

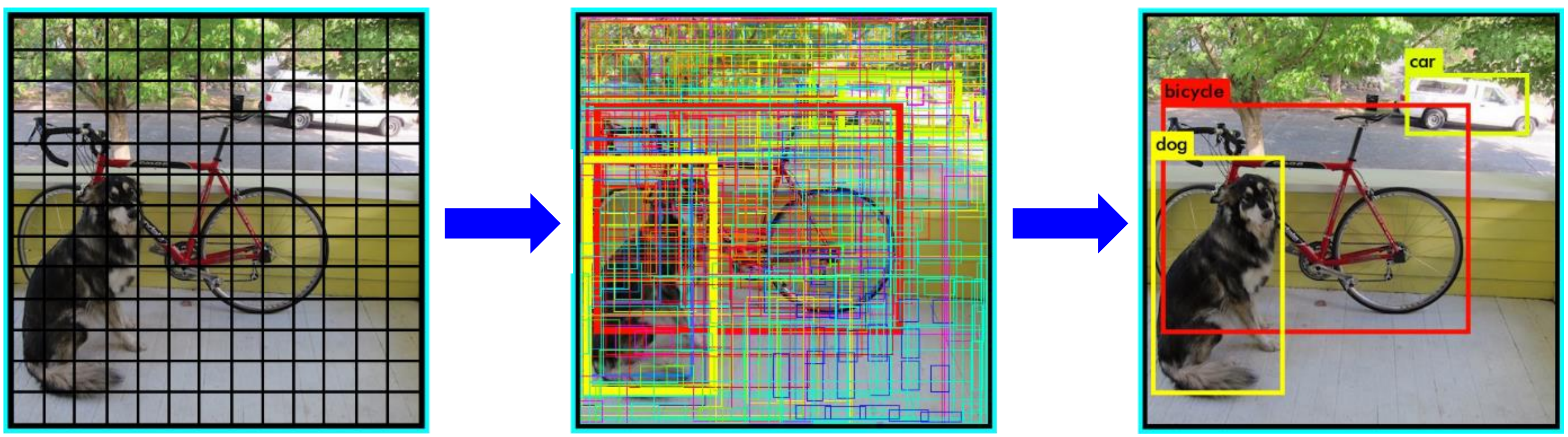
- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based On YOLO
- Experimental Results
- Conclusion

YOLO – a CNN for Deep Learning
– YOLO: Main Concept

A. Splits input image into 13x13 cells
B. Predicts 5 bounding boxes for each cell
C. Final detections → thresholding and NMS (Non-maximum Suppression)

13

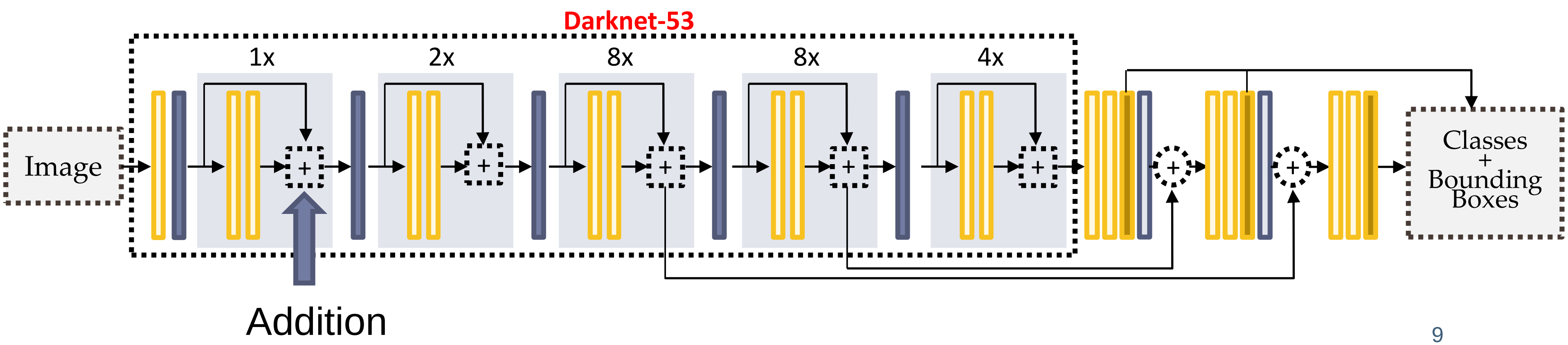
13



A B C

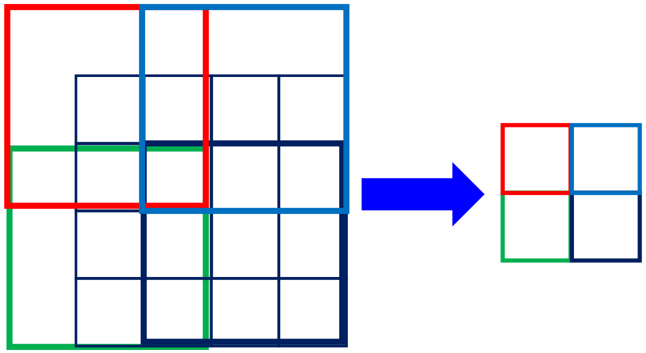
YOLO – a CNN for Deep Learning
– YOLOv3

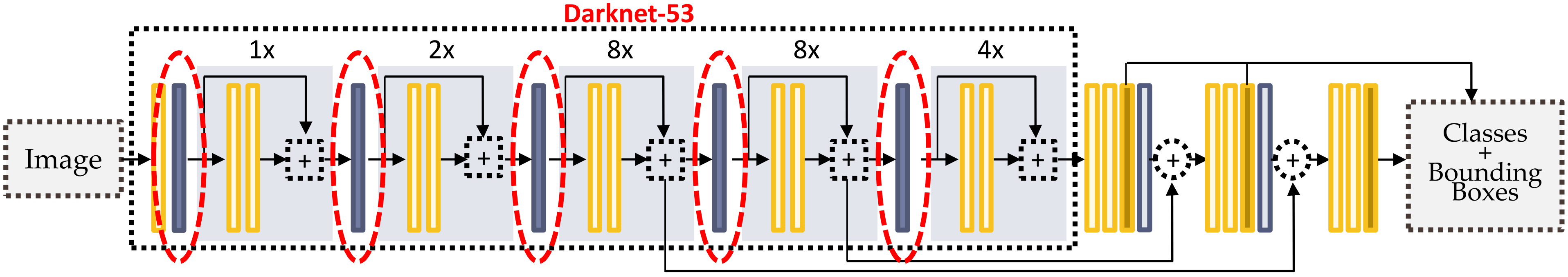
- Architecture design
  - Darknet-53 → learns better, computes faster
    - 53 convolution layers and 5 stride-2 convolution layers
  - No max-pooling → use stride-2 convolution
    - Preserve more information, each pixel is responsible for layer output
  - Up-sample layer → multi-scale prediction (3 scales)
    - To find objects at different sizes



YOLO – a CNN for Deep Learning
– YOLOv3: Stride-2 Convolution

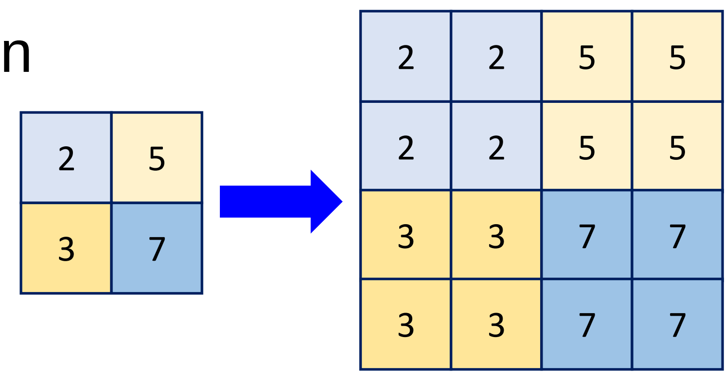
- Stride-2 convolution reduces the dimensionality of each feature map
  - Use convolutions to produce output features
    - ➢ Each feature has contribution to output features

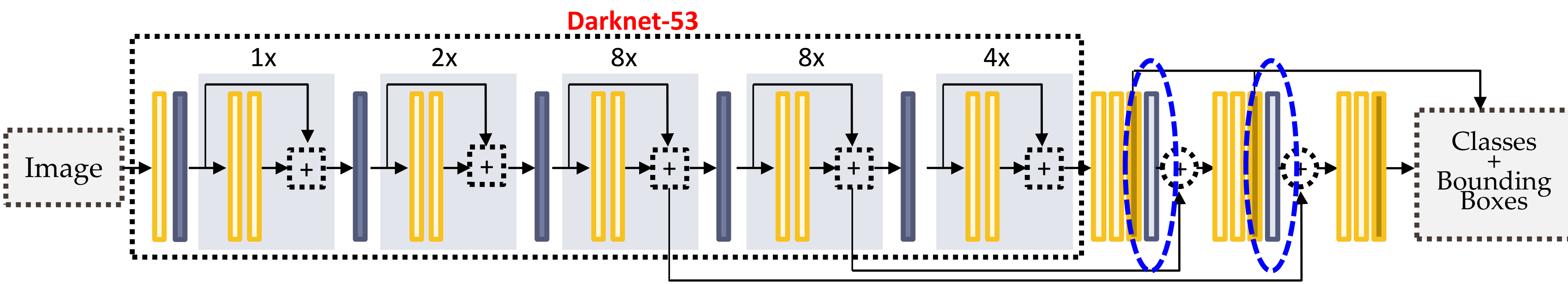




YOLO – a CNN for Deep Learning
– YOLOv3: Up-sample Layer

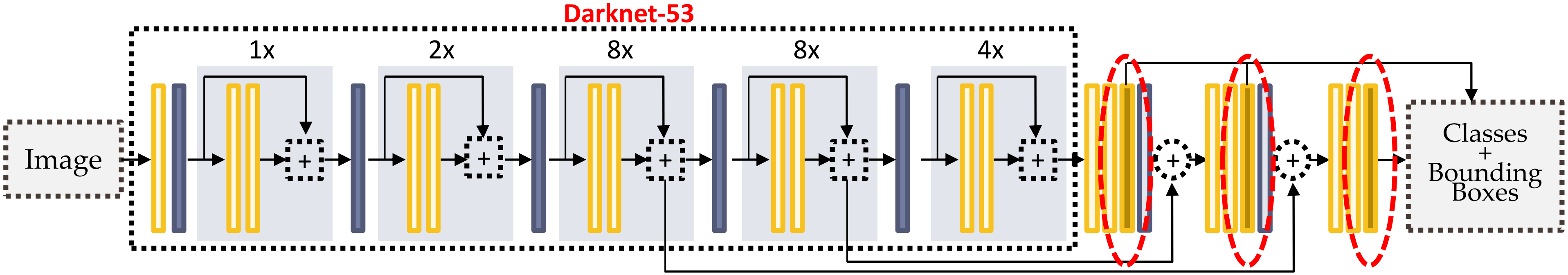
- Up-sample increases the dimensionality of each feature map
  - Larger feature map
    → detection of smaller objects
  - Concatenation of object information
    → better detection result





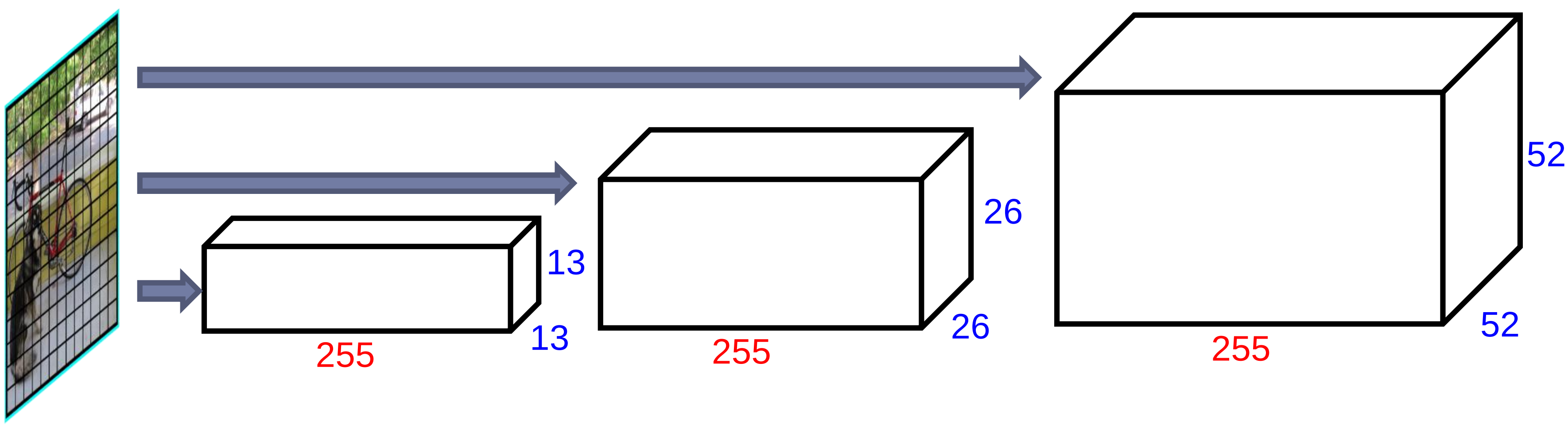
YOLO – a CNN for Deep Learning
– YOLOv3: Output Layers

- Output layer feature map size: 13×13, 26×26, 52×52
- For each scale
  - Each cell predicts 3 bounding boxes
  - Each bounding box needs 85 parameters
    - $x, y, w, h, confidence$
    - $class1, class2, \cdots, class80$ (COCO has 80 classes)
  - The depth of output layer is 3×85 = 255



YOLO – a CNN for Deep Learning
– YOLOv3: Output Layers

- Output layer feature map size: 13×13, 26×26, 52×52
- For each scale
  - Each cell predicts 3 bounding boxes
  - Each bounding box needs 85 parameters
    - $x, y, w, h, confidence$
    - $class1, class2, \cdots, class80$ (COCO has 80 classes)
  - The depth of output layer is 3×85 = 255



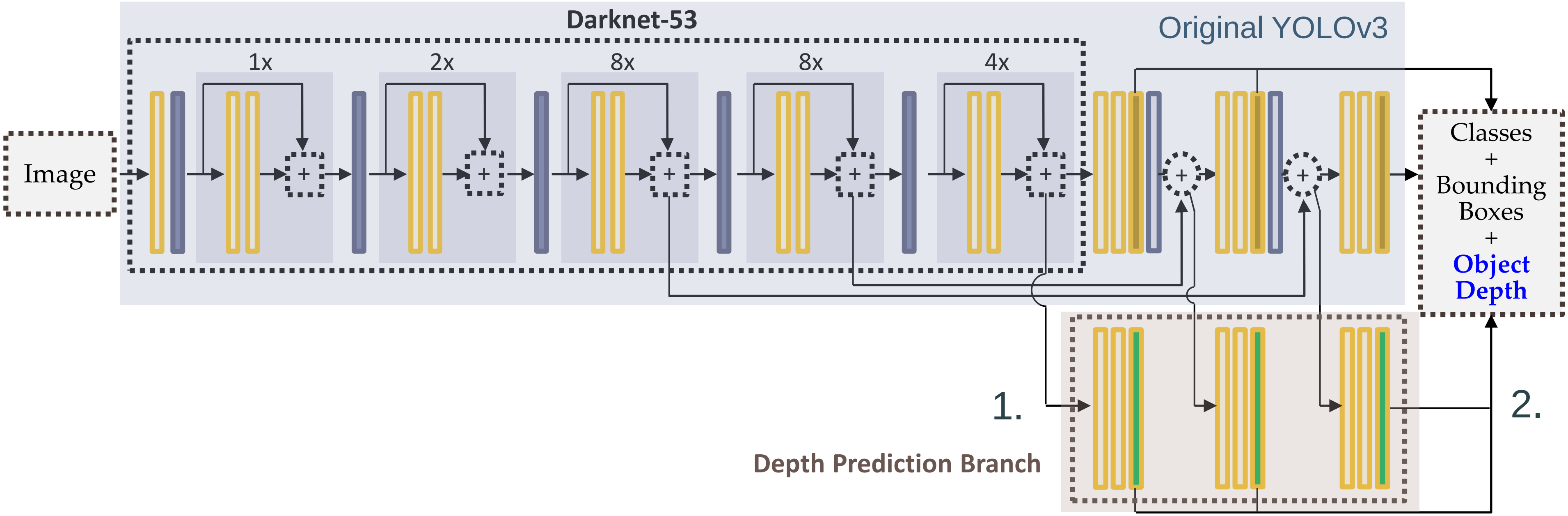
Layer Type	Feature Map Number	Filter Size	Filter Stride	Output Size
Convolutional	512	3×3	1	13×13
Convolutional	1024	3×3	1	13×13
Convolutional	512	3×3	1	13×13
Convolutional	1024	3×3	1	13×13
Convolutional (A1)	512	3×3	1	13×13
convolutional	1024	3×3	1	13×13
Prediction 1 (scale 1)	**255**	1×1	1	$\mathbf{13 \times 13}$
A1	256	3×3	1	13×13
Up-sample	256	$2 \times$	2	26×26
Convolutional+A2	256	3×3	1	26×26
Convolutional	512	3×3	1	26×26
Convolutional	256	3×3	1	26×26
Convolutional	512	3×3	1	26×26
Convolutional (B1)	256	3×3	1	26×26
Convolutional	512	3×3	1	26×26
Prediction 2 (scale 2)	**255**	1×1	1	$\mathbf{26 \times 26}$
B1	128	3×3	1	26×26
Up-sample	128	$2 \times$	2	52×52
Convolutional+B2	128	3×3	1	52×52
Convolutional	256	3×3	1	52×52
Convolutional	128	3×3	1	52×52
Convolutional	256	3×3	1	52×52
Convolutional	128	3×3	1	52×52
Convolutional	256	3×3	1	52×52
Prediction 3 (scale 3)	**255**	1×1	1	$\mathbf{52 \times 52}$

Outline

- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based On YOLO
- Experimental Results
- Conclusion

The Proposed Depth Prediction
– YOLOv3-based Architecture

- Two modifications
  1. Multiple depth prediction branches
  2. Modify the output layer

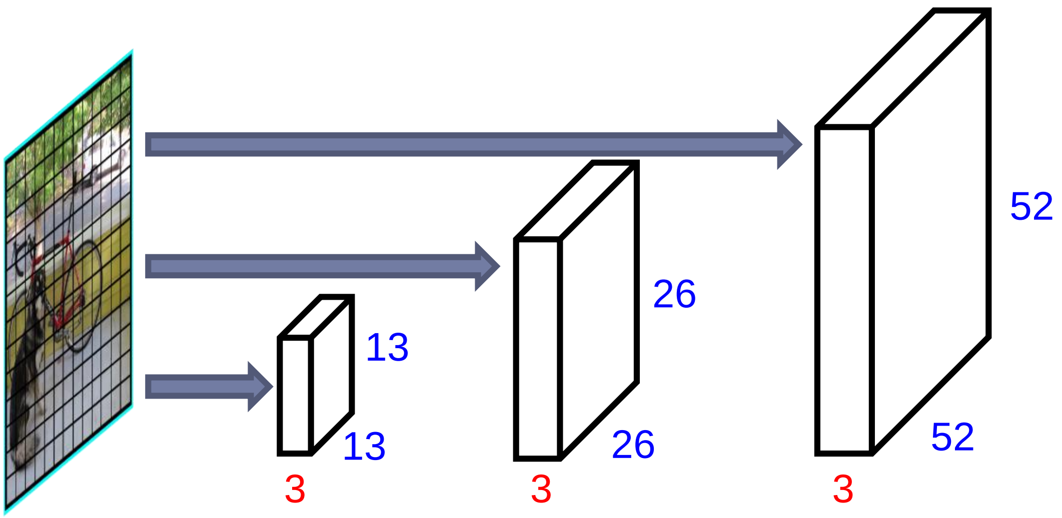


The Proposed Depth Prediction
– YOLOv3-based Architecture: Multiple depth branches

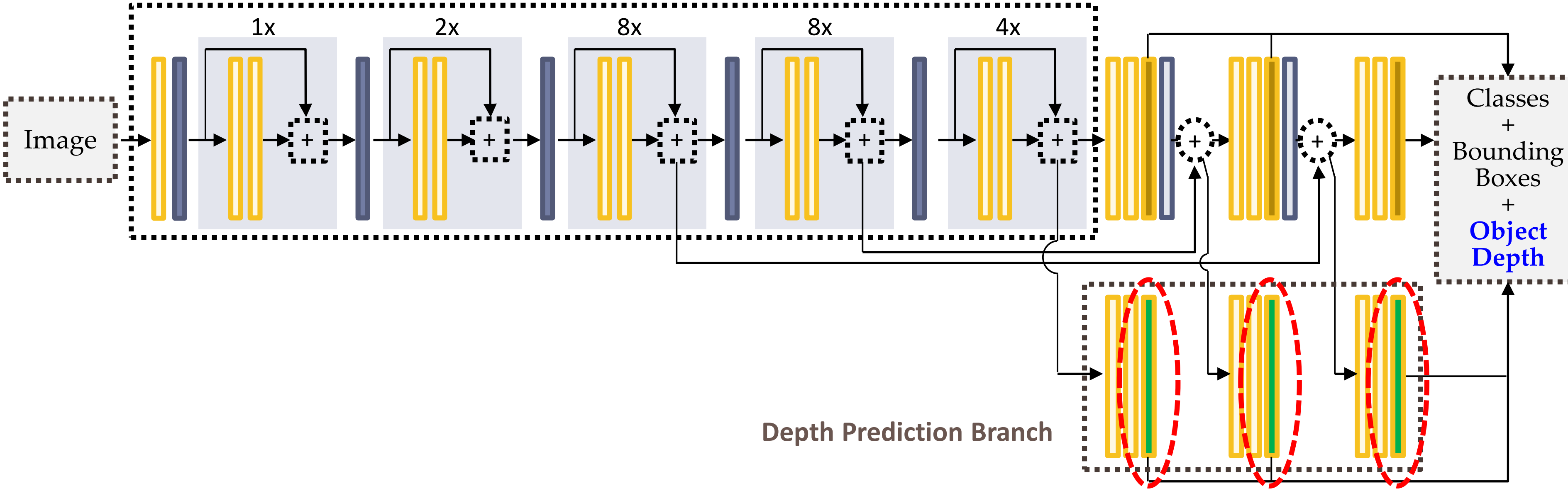
- 3 prediction layers in original YOLOv3

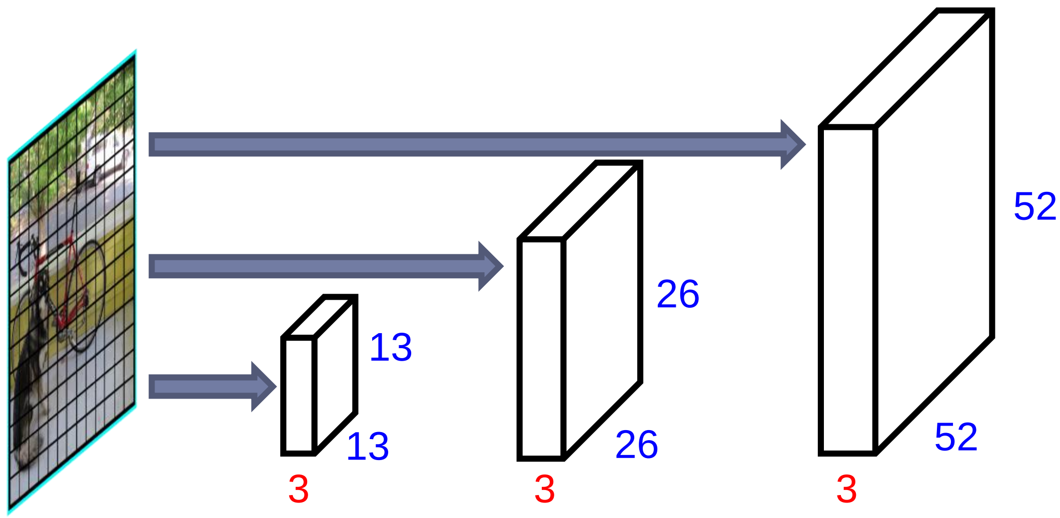
  → 3 depth prediction branches
  - Output layer feature map sizes: 13×13, 26×26, 52×52
  - 3 boxes per cell (for each scale)
  - One depth prediction for each box

➢ The sizes of output layer: 13×13×3, 26×26×3, 52×52×3



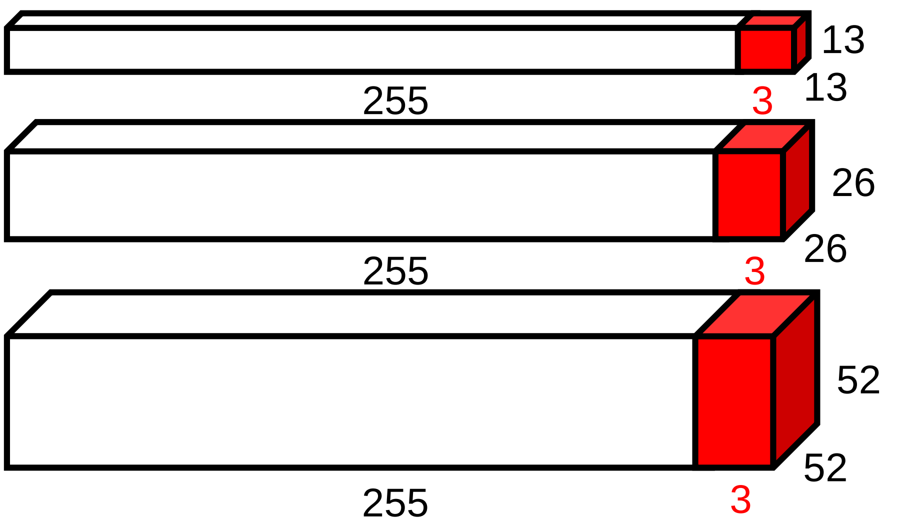
The Proposed Depth Prediction
– YOLOv3-based Architecture: Multiple depth branches





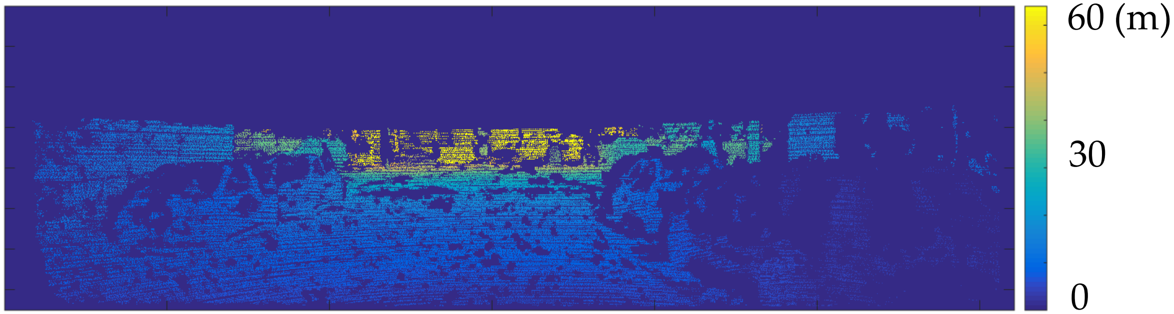
The Proposed Depth Prediction
– YOLOv3-based Architecture: Output Layer

- Each bounding box now needs 85+1 parameters
  - $x, y, w, h, confidence, depth$
  - $class1, class2, \cdots, class80$
- Each cell predicts 3 bounding boxes
- The depth of output layer is 3×(85+1) = 255+3



The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

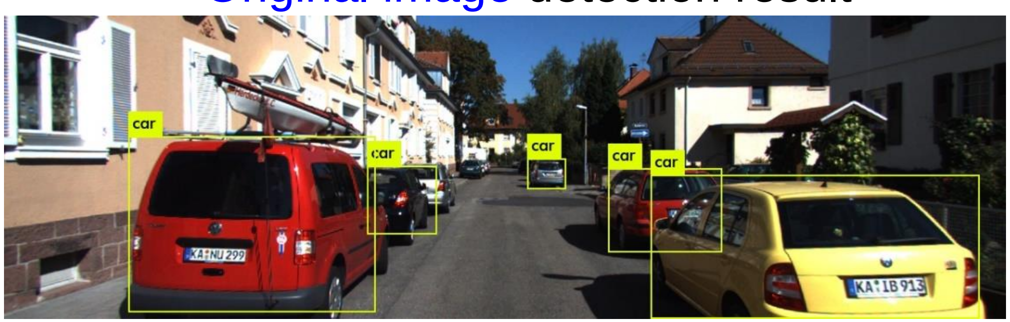
- KITTI has RGB image and corresponding depth image
- To train our model: use ground truth of object depth
  - Use RGB images to locate objects
  - Use depth images to calculate ground truth of object depth



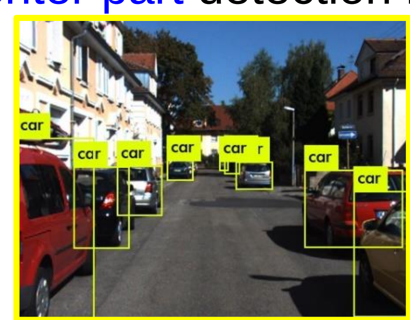
The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

1. Use original YOLOv3 to locate objects
   - The input of original YOLOv3 is square(1:1), and may cause object distortion and feature loss

Original image detection result



Center part detection result



The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

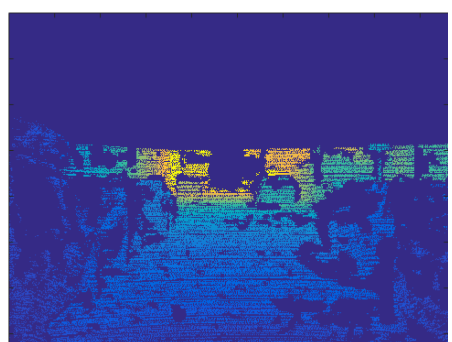
2. Split images to near square
   - Original: 1242×375 (3.3:1)
   - Split: 480×375 (1.2:1)

The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

3. Refine object location using Mask-RCNN
   - Object bounding box is not accurate enough
     - ➢ object depth may be erroneous

The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

3. Refine object location using Mask-RCNN
   - Object bounding box is not accurate enough
     - object depth may be erroneous



The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

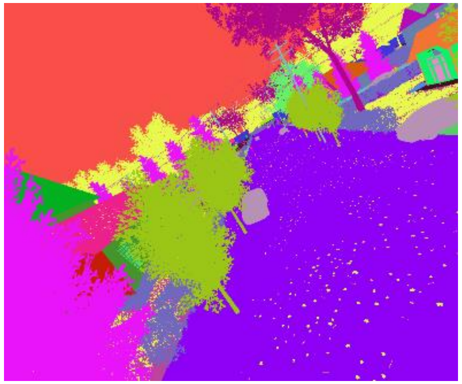
4. Define object depth (for obstacle detection)
   - Use average depth of the nearest 20% object points
   - KITTI dataset: 60K training images & 130K objects/depths

The Proposed Depth Prediction
– Build a Dataset Using AirSim

- AirSim – a program to generate training data
  - Load different scenes – different data domains
  - Generate different types of ground truth
    - RGB images / depth images / segmentation images
  - Use different vehicles
    - Car
    - ✓ Drone

RGB

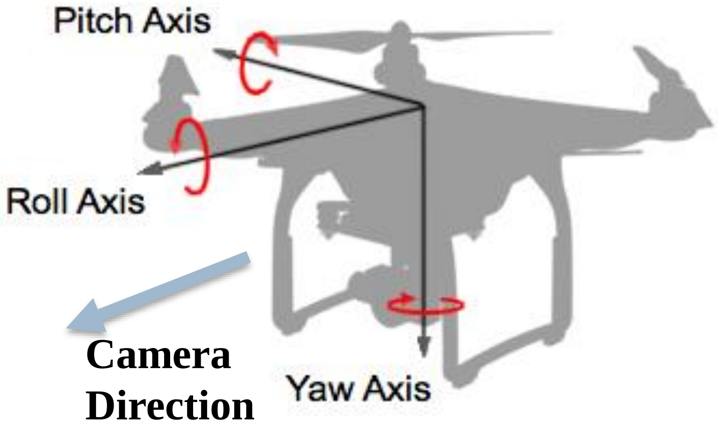
Depth



Segmentation

The Proposed Depth Prediction
– Build a Dataset Using AirSim — Data Collection

- Camera position
  - Equally spaced samples along red lines: 1m spacing
  - Height: 1, 2 … 10m
- Camera direction
  - Random samples from normal distribution
  - Yaw: μ=0, σ=30; Pitch: μ=0, σ=15; Roll: μ=0, σ=15



The Proposed Depth Prediction
– Build a Dataset Using AirSim — Generate GT

- Bounding box ground truth

Input segmentation masks contours bounding boxes

- Object depth ground truth
  - Nearest 20% depth average in the mask
- Dataset detail
  - Number of training images: 32,800
  - Number of objects: 60,000

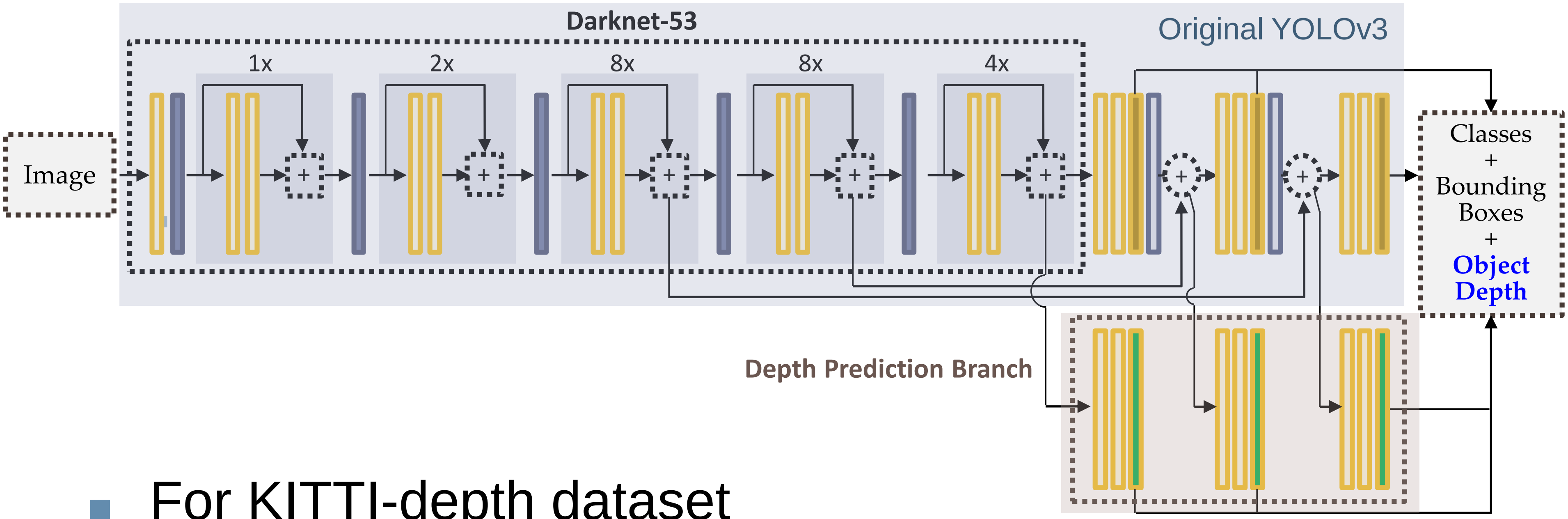
The Proposed Depth Prediction
– Training Details

- Pre-trained COCO dataset
- Use data augmentation
  - Flip, rotate, random crop, adjust hue, saturation, exposure
- Add depth prediction loss (L_1 distance)

$$\sum_{i}^{N} |depth_i - depth_i^*|$$

- For KITTI-depth dataset
  - Detection result of original YOLOv3 is good
    - Train depth prediction branch only
- For AirSim Dataset
  - Detection result of original YOLOv3 is no good
    - Train full architecture

The Proposed Depth Prediction
– Training Details



- For KITTI-depth dataset
  - Detection result of original YOLOv3 is good
    - ➢ Train depth prediction branch only
- For AirSim Dataset
  - Detection result of original YOLOv3 is no good
    - ➢ Train full architecture

The Proposed Depth Prediction
– Evaluation Metrics

- For object detection

$$Precision = \frac{\#\ of\ correct\ detections}{\#\ of\ total\ detecitons}, \quad Recall = \frac{\#\ of\ correct\ detections}{\#\ of\ ground\ truths}$$

- For depth prediction
  - Absolute relative difference (ARD): $\frac{1}{N}\sum_i^N \frac{|y_i - y_i^*|}{y_i^*}$
  - Root mean square error (RMSE): $\sqrt{\frac{1}{N}\sum_i^N (y_i - y_i^*)^2}$
  - Threshold: percentage of y_i such that

$$\delta = \max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) < thr$$

y_i: predicted depth
y_i^*: ground truth depth
N: total object number

Outline

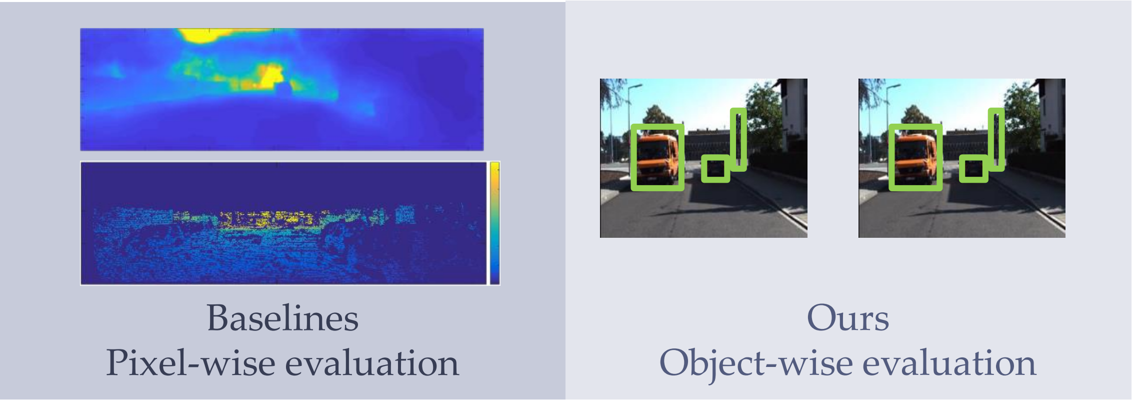
- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based on YOLO
- Experimental Results
- Conclusion

Experimental Results
– Testing Dataset Detail

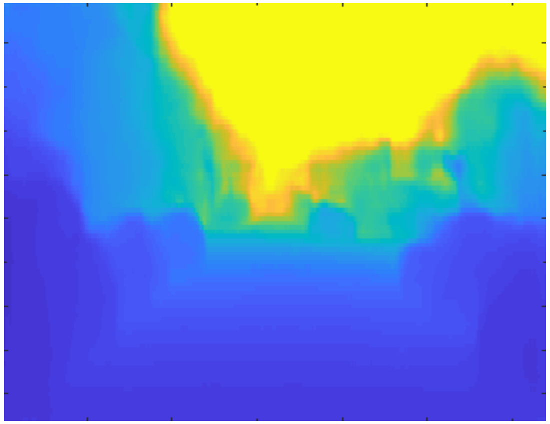
- KITTI-depth
  - Number of testing images: 5,200
  - Number of objects: 14,200
- AirSim
  - Number of testing images: 3,400
  - Number of objects: 5,100

Experimental Results
– From Depth Image to Object Depth

- Evaluation
  - Baselines
    - Depth per pixel
  - Our method
    - Depth per object



- To make fair comparison
  - ➢ Transform the result of baseline



Experimental Results
– Comparison with other methods

- Observation – KITTI-depth dataset
  - YOLOv3-based model compares favorably with other methods (and better than YOLOv2-based model)

Model	RMSE (meter)			ARD	Threshold (No Cap)			FPS
	No Cap	*Cap 50	Cap 30		$\delta < 1.25$	$\delta < 1.56$	$\delta < 1.95$	
Godard *et al*. (CVPR, 2017)	6.011	4.939	2.853	0.207	0.676	0.845	0.925	-
Kuznietsov *et al*. (CVPR, 2017)	4.958	**3.483**	**1.903**	**0.131**	0.832	0.950	**0.987**	-
Ours (YOLOv2-based)	**4.373**	3.908	2.887	0.159	**0.841**	**0.953**	0.978	**42.1**
Ours (YOLOv3-based)	**2.927**	**2.655**	**1.899**	**0.086**	**0.936**	**0.991**	**0.997**	33.9

*Cap 50: only objects within 50m are calculated

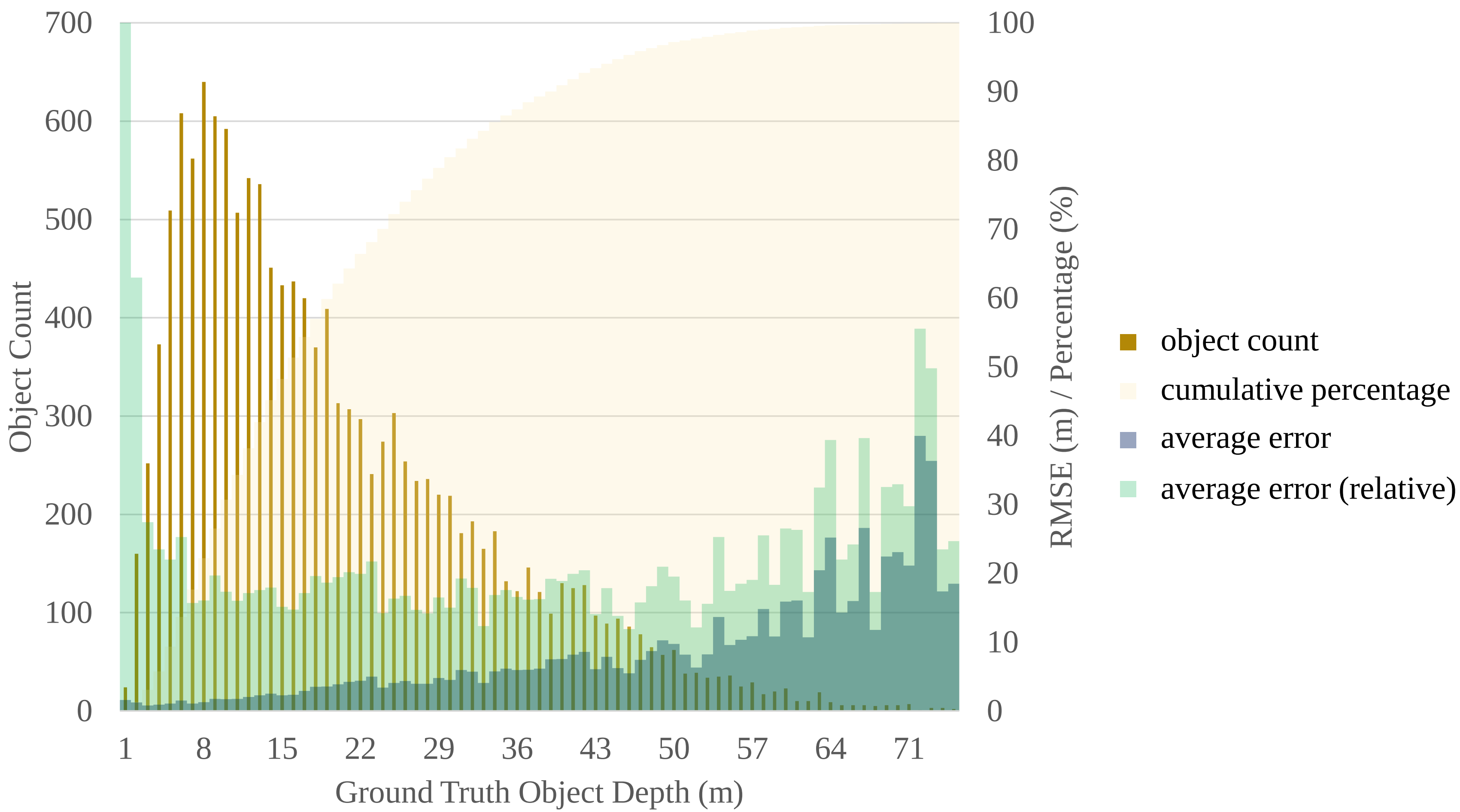
*Tested on GTX1080

Lower is better

Higher is better

Experimental Results
– Comparisons between YOLOv2 and YOLOv3-based Model

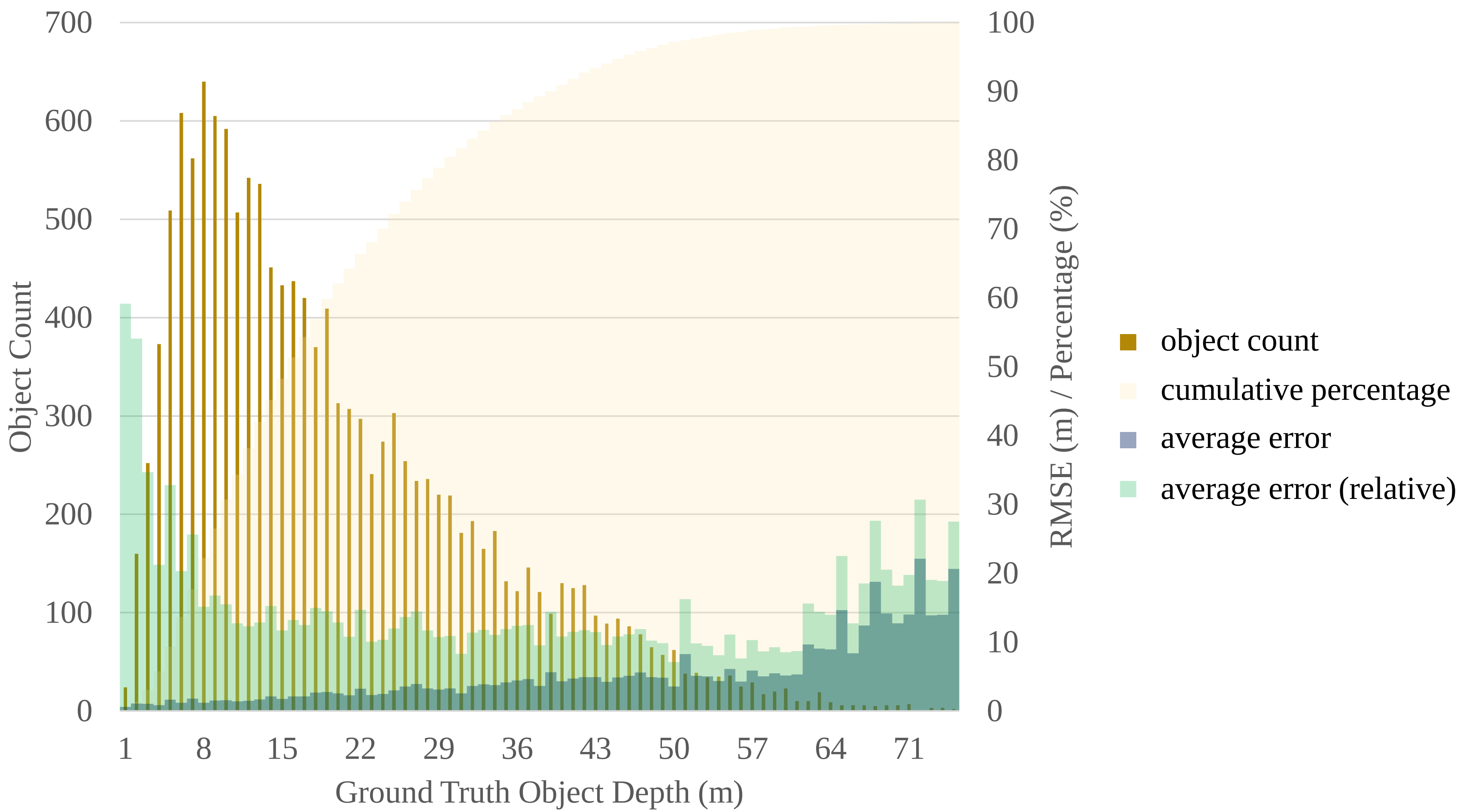
- YOLOv2-based model testing result on KITTI-depth dataset



Experimental Results

– Comparisons between YOLOv2 and YOLOv3-based Model

- YOLOv3-based model testing result on KITTI-depth dataset

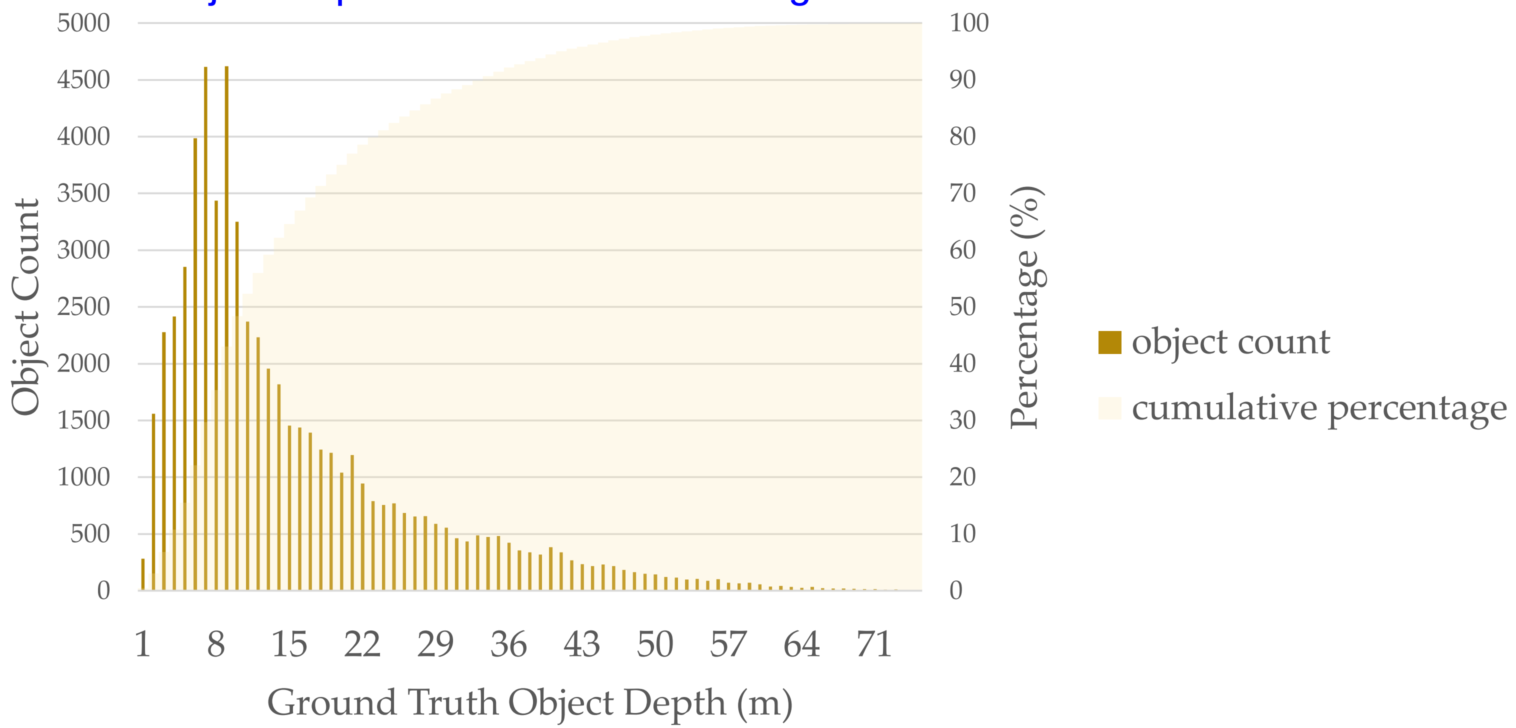


Experimental Results

– Comparisons between YOLOv2-based and YOLOv3-based Model

- Observations
  - YOLOv3-based model is better than YOLOv2-based model
  - Fewer training data → larger relative error

Object depth distribution of training set



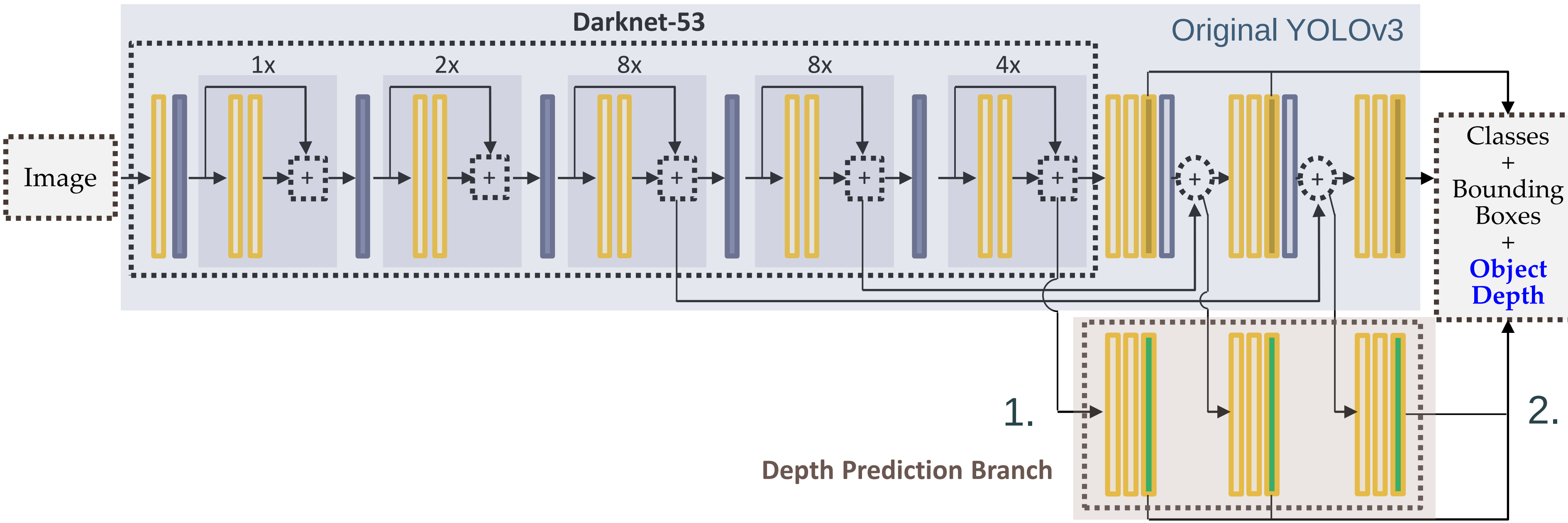
Experimental Results
– Comparisons between different input sizes

- Observations – KITTI-depth dataset
  - Increasing input size decreases performance
  - Larger input size → lower FPS

Model	Input Size	RMSE (meter)			ARD	Threshold (No Cap)			FPS
		No Cap	Cap 50	Cap 30		$\delta < 1.25$	$\delta < 1.56$	$\delta < 1.95$	
Ours (YOLOv3)	416×416	**2.927**	**2.655**	1.899	**0.086**	0.936	**0.991**	**0.997**	**33.9**
	480×480	2.981	2.671	**1.871**	0.092	**0.937**	**0.991**	**0.997**	28.0
	544×544	2.983	2.695	1.909	0.093	0.936	**0.991**	**0.997**	22.6

Lower is better

Higher is better



Model	Input Size	RMSE (meter)			ARD	Threshold (No Cap)			FPS
		No Cap	Cap 50	Cap 30		$\delta < 1.25$	$\delta < 1.56$	$\delta < 1.95$	
Ours (YOLOv3)	416 × 416	**2.927**	**2.655**	1.899	**0.086**	0.936	**0.991**	**0.997**	**33.9**
	480 × 480	2.981	2.671	**1.871**	0.092	**0.937**	**0.991**	**0.997**	28.0
	544 × 544	2.983	2.695	1.909	0.093	0.936	**0.991**	**0.997**	22.6

Lower is better

Higher is better

Experimental Results
– Comparisons between different input sizes

- Observation – AirSim dataset
  - Larger input size → higher recall rate
  - Larger input size → higher RMSE error

Model	Input Size	RMSE (meter)			ARD	Threshold (No Cap)			Precision (%)	Recall (%)	FPS
		No Cap	Cap 50	Cap 30		$\delta < 1.25$	$\delta < 1.56$	$\delta < 1.95$			
Ours Fix (YOLOv3)	416×416	6.473	5.376	3.257	0.195	0.401	0.513	0.597	22.6	43.6	**33.9**
Ours Full (YOLOv3)	416×416	**3.323**	**2.815**	1.719	**0.075**	0.868	0.922	0.938	**82.2**	85.2	**33.9**
	480×480	**3.773**	**2.910**	**1.342**	**0.085**	**0.929**	**0.961**	**0.972**	81.6	**90.2**	28.0
	544×544	3.935	2.963	**1.431**	0.092	**0.928**	**0.962**	**0.975**	**82.2**	**90.7**	22.6

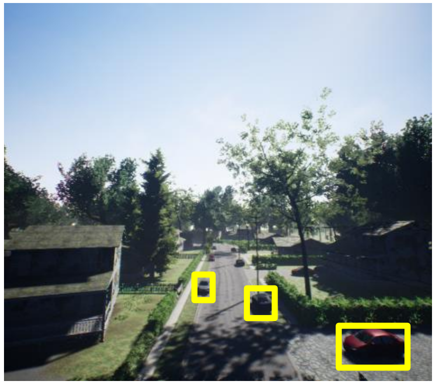
Lower is better

Higher is better

Experimental Results
– Comparisons between different input sizes

$$\delta = \max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right)$$

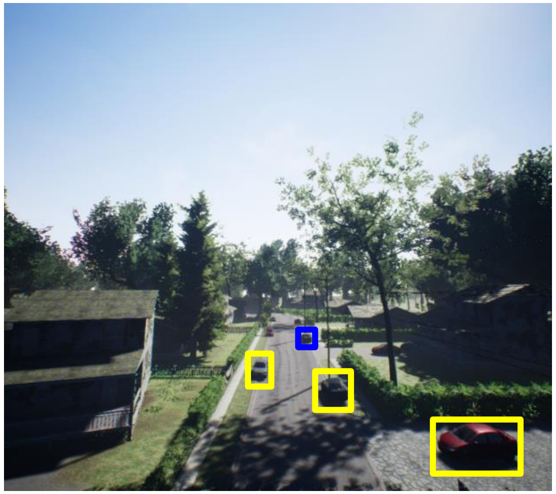
416x416



	Predicted Depth	Ground Truth	δ
Object 1	18	15	1.2
Object 2	22	25	1.13
Object 3	26	30	1.15

RMSE: 3.36

544x544



	Predicted Depth	Ground Truth	δ
Object 1	18	15	1.2
Object 2	22	25	1.13
Object 3	26	30	1.15
Object 4	55	65	1.18

RMSE: 5.78

Experimental Results
– Interactions between AirSim and YOLOv3-based Model

- Observations
  - Training can improve precision and recall rates
  - Depth architecture helps detector learn better
  - Detector learns better → depth predicts better

Detector fixed

Depth metric | Detection metric

Model	RMSE (meter)			Precision (%)	Recall (%)
	No Cap	Cap 50	Cap 30		
Original YOLOv3 (Not trained)	-	-	-	22.60	43.67
Original YOLOv3 (Trained)	-	-	-	67.95	71.71
Ours Fix	6.473	5.376	3.257	22.60	43.67
Ours Full	**3.323**	**2.815**	**1.719**	**82.20**	**85.25**

Lower is better

Higher is better

Experimental Results
– Qualitative Results

[Go to demo video]

Outline

- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Using YOLO
- Experimental Results
- Conclusion

Conclusion

- KITTI dataset is adapted and have ground truth object depth
- The original YOLOv2 and YOLOv3 are modified to incorporate depth prediction
- The proposed architecture compares favorably on other depth prediction methods (KITTI)
- Extra depth prediction architecture can enhance the performance of object detection (AirSim)

Thank you