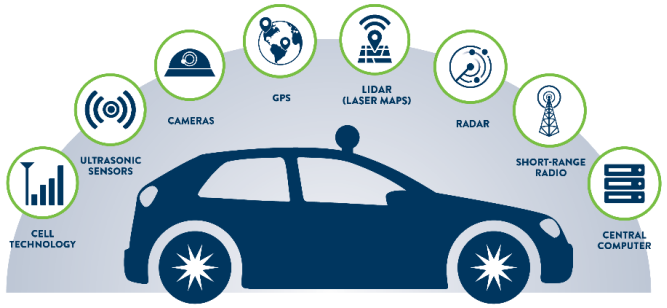
EdgeCompression: An Integrated Framework for Compressive Image Processing on CAVs

Sidi Lu, Xin Yuan, Weisong Shi

About the Authors

- Sidi Lu
  - Ph.D. Candidate Wayne State University
- Xin Yuan
  - Video analysis and coding lead researcher in Bell Labs
  - Received Ph.D. from Hong Kong Polytechnic University
- Weisong Shi
  - Professor of Computer Science from Wayne State University
  - Distinguished Faculty Fellow, IEEE Fellow, ACM Distinguished Scientist



What is a CAV?

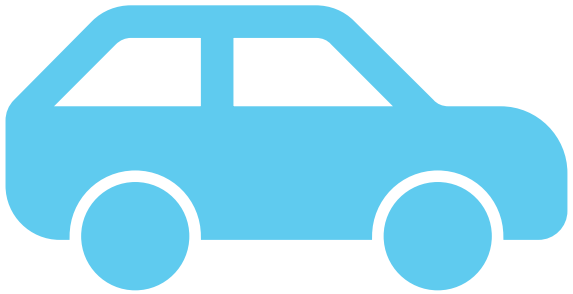
Connected Autonomous Vehicle

Key Word: **Connected**

Major assumption that all vehicles can maintain some form of connectivity through 4G/5G or other IP related connections

NOTE: In this paper they focus on the detection of other cars and NOT any other object or thing

Major Problem



- Autonomous Vehicles rely strongly on “machine vision” to determine information about the environment around it
- The state of the art in image recognition is relies on Deep Neural Networks
  - DNN require powerful compute to predict accurately and quickly
  - The “edge” or vehicles are not equipped with the compute as necessary

Previous Solutions

Reduced Frame Rate

- CON: Could miss certain environmental triggers with a reduced frame rate

Reduced Frame Size

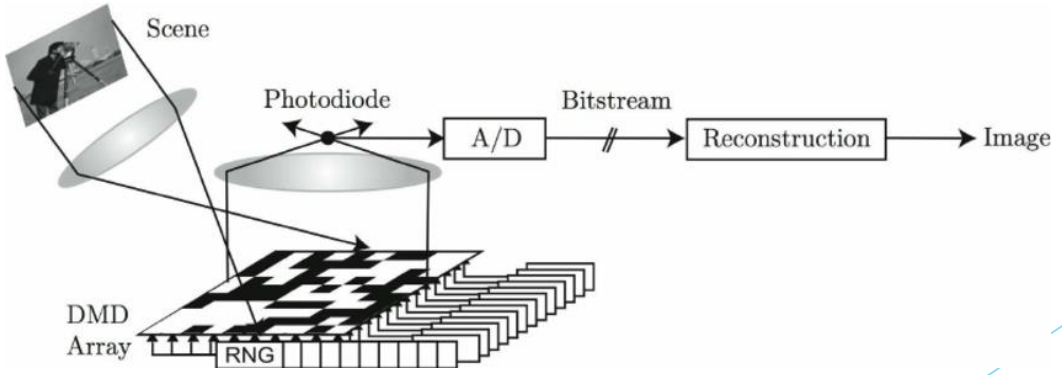
- CON: Lower quality images could result in missed objects or environmental triggers

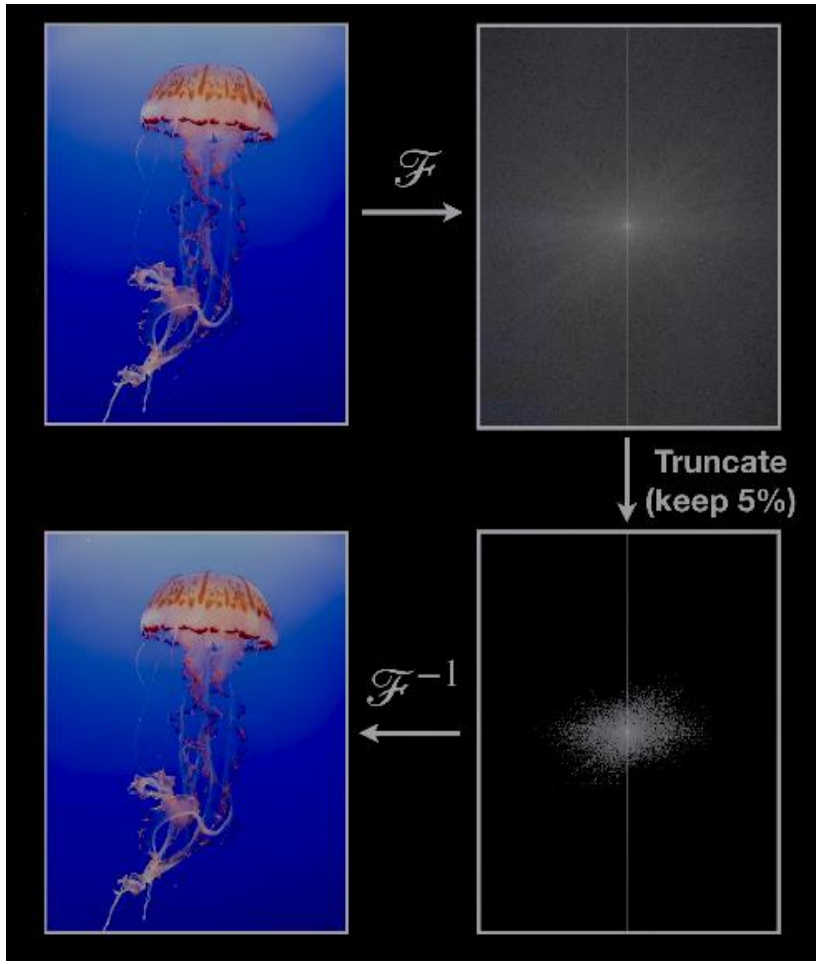
What can we do?

- What if we could reduce the image size without sacrificing important details?
  - Faster prediction time
  - Similar if not equal prediction accuracy
- Introducing: Compressive Imaging (CI) Cameras

Compressive Imaging

- The advent of the digital camera has made computer vision possible
- Traditional computer vision follows an “image first, process later” technique
- Compressive Imaging proposes “What if we can do some of that processing initially”



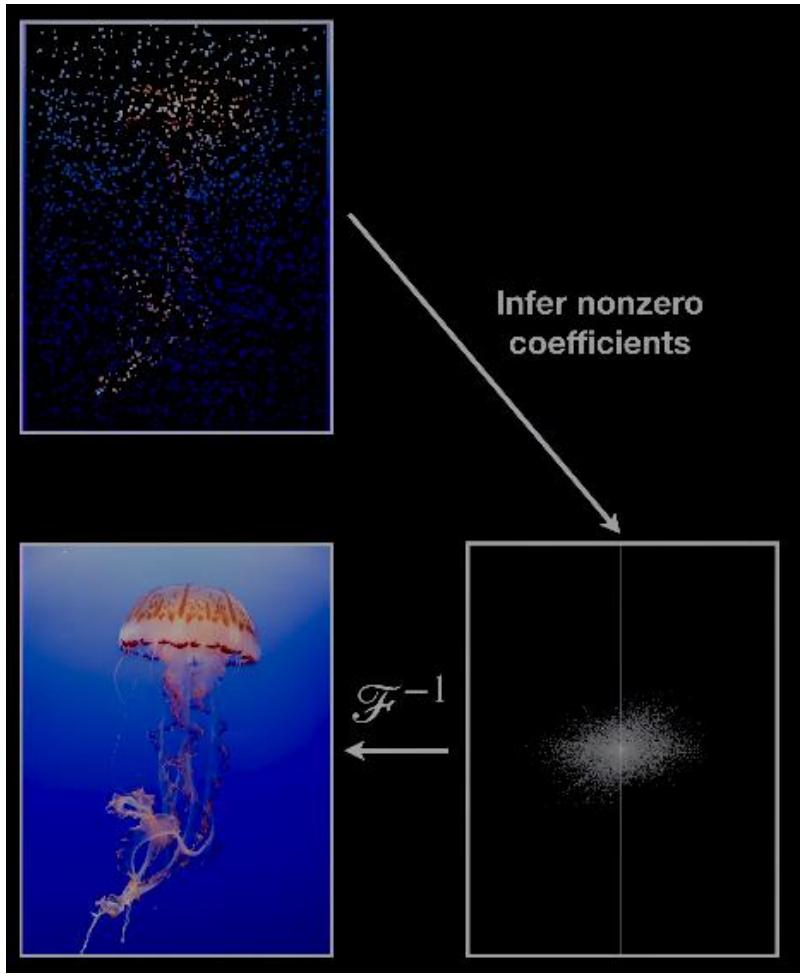


Video Credits: Steve Brunton (UW)

Compressive Imaging

Applying Fourier transform to the image allows for the ability to drop 95% of unnecessary pixels

Applying the inverse Fourier transform allows the image to be recovered



Video Credits: Steve Brunton (UW)

Compressive Imaging

What if we could instead sample the original image and apply some form of an assumption about the pixels around it?

If we could accurately solve for this “assumption” in the form of a sparse coefficient matrix, we can recover the original image

But what about for video?

- Instead of down sampling a single image, down sampling many frames at once and compress into a single measurement
- How?
  - Apply a form of modulation to each frame (in the form of abit mask)
    - Modulation allows for some randomization between each frame to capture all information
  - Allow the CI Camera to sum each of the frames by integrating the ligh in the image system
- DNN's have proved valuable in learning the "sparse coefficient matrix" for decoding the compressed images

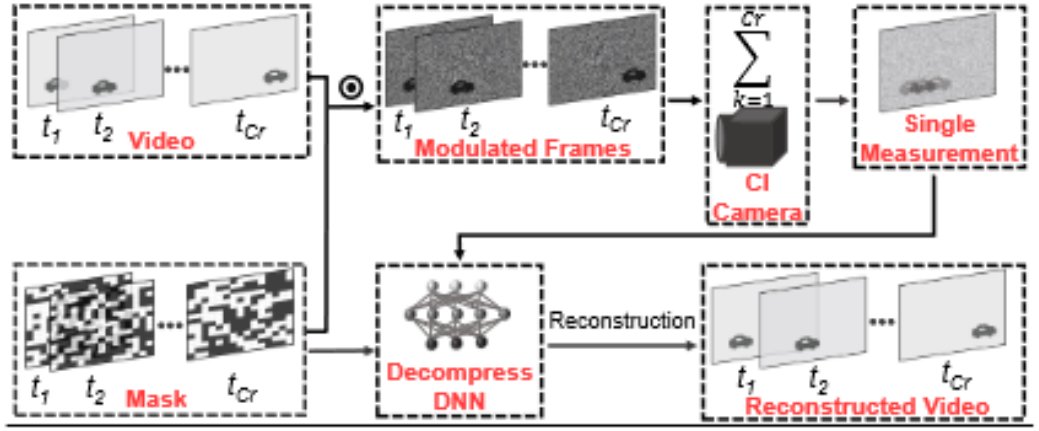
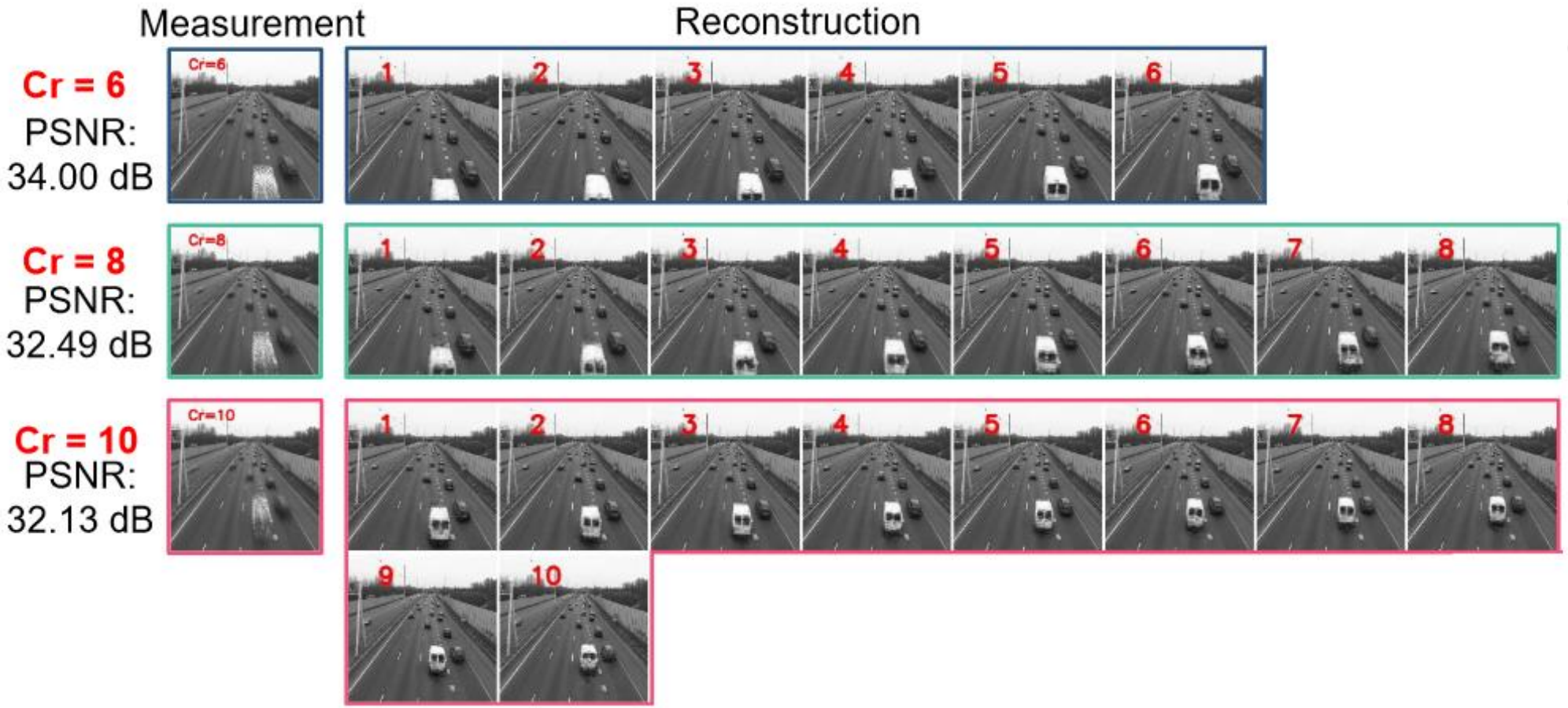


Fig. 2. The framework of video compressive imaging (CI). Here, ⊙ denotes the element-wise product.

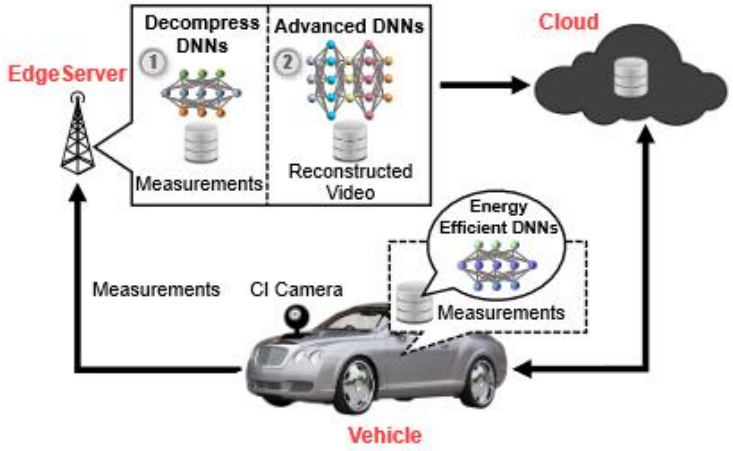

Measurement
Reconstruction
Cr = 6
PSNR:
34.00 dB
Cr=6
1
2
3
4
5
6
Cr = 8
PSNR:
32.49 dB
Cr=8
1
2
3
4
5
6
7
8
Cr = 10
PSNR:
32.13 dB
Cr=10
1
2
3
4
5
6
7
8
9
10

Their Solution => EdgeCompression

- A Vehicle-EdgeServer-Cloud Closed-Loop framework with the goal of:
  1. Accelerate video analysis
  2. Decrease Energy consumption

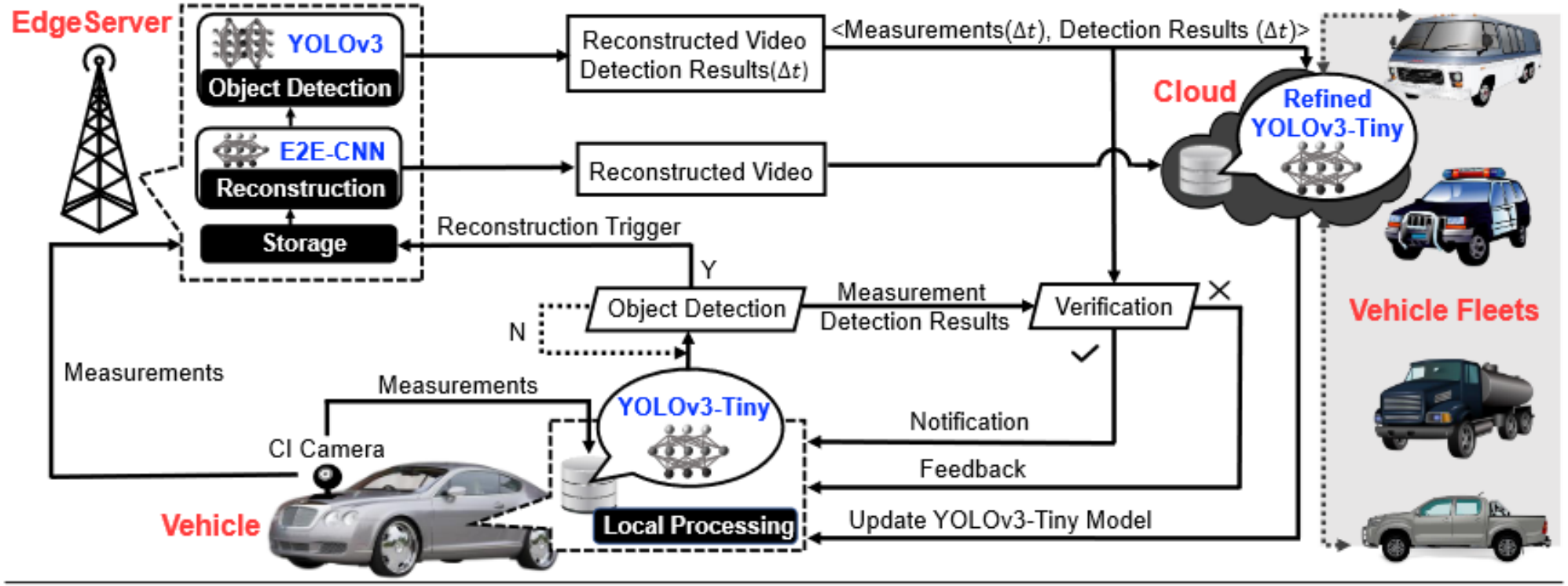
- Address the three core challenges:
  1. High speed cameras produce tons of data, require high bandwidth
  2. DNN's are traditionally slow to predict, but accurate
  3. Edge devices are resource constrained

Closed Loop System – 10,000 ft view



Compressive Image Data Generated on Car

1. **On-Device** low power DNN predicts quickly with CI Data for real time DNN Prediction
2. **On-Edge** after certain triggers, CI data sent to edge, edge uses reconstruction DNN to reconstruct image, high power DNN predicts and re-informs device of accuracy
3. **On-Cloud** aggregation of raw data and prediction results to:
    1. Improve future compressed DNN's
    2. Provide less time sensitive predictions like traffic control or path prediction


EdgeServer
YOLOv3
Object Detection
E2E-CNN
Reconstruction
Storage
Reconstructed Video
Detection Results(Δt)
Reconstructed Video
<Measurements(Δt), Detection Results (Δt)>
Cloud
Refined
YOLOv3-Tiny
Vehicle Fleets
Reconstruction Trigger
Y
N
Object Detection
Measurement
Detection Results
Verification
Measurements
Measurements
YOLOv3-Tiny
Notification
Feedback
CI Camera
Vehicle
Local Processing
Update YOLOv3-Tiny Model

Data Sets Used for Experimentation

- AAU Rain Snow Dataset
  - 22 x 5-minute videos of traffic intersections
- BDD100K Dataset
  - Berkeley DeepDrive has 1,100 hours od driving experience
- PDTV Dataset
  - Public dataset of traffic videos
- DynTex Dataset:
  - Collection of dynamic texture videos

Hardware Setup

- Intel FRD (Fog Reference Design)
  - Deployed on vehicles
- NIVIDIA GPU Workstation
  - Deployed as edge server

	Intel FRD	NVIDIA GPU Workstation
CPU	Intel Xeon E3-1275 v5	Intel Xeon E5-2690 v4
GPU	NONE	4 × 11 GB GeForce RTX 2080 Ti
Frequency	3.6 GHz	2.6 GHz
Core	4	14
Memory	32 GB	64 GB
OS	Ubuntu 16.04.6 LTS	Ubuntu 16.04.6 LTS

(a)

(b)

Experiment

The ground truth video feed from each data set to simulate the output of a CI Camera

- Note: They compress the images to grayscale prior to CI Simulation compression

They fed the compressed image to the YOLOv3-Tiny running on the Intel machines for quick predictions

They decompressed the image on the Edge-Server GPU cluster using E2E-CNN decompression Architecture

They fed the decompressed image through the YOLOv3 running on the Edge-Server for higher accuracy predictions

Data Variation

Paper does a good job of experimenting on a well varied data set

1. Data Set Variation
    1. They use 4 different state of the art data sets for training and testing
2. Compression Ration (CR) Variation
    1. They vary the number of frames compressed into one frame
3. Training Variation
    1. They train the YOLOv3-Tiny Algorithm (on CAV) three different ways
        1. Gray-Scale → Base images just w/ a gray scale
        2. Measurement → Compressed images as output of CI Camera
        3. Reconstructed → Reconstructed images from E2E-CNN reconstruction DNN

Intuition: Higher CR means longer reconstruction

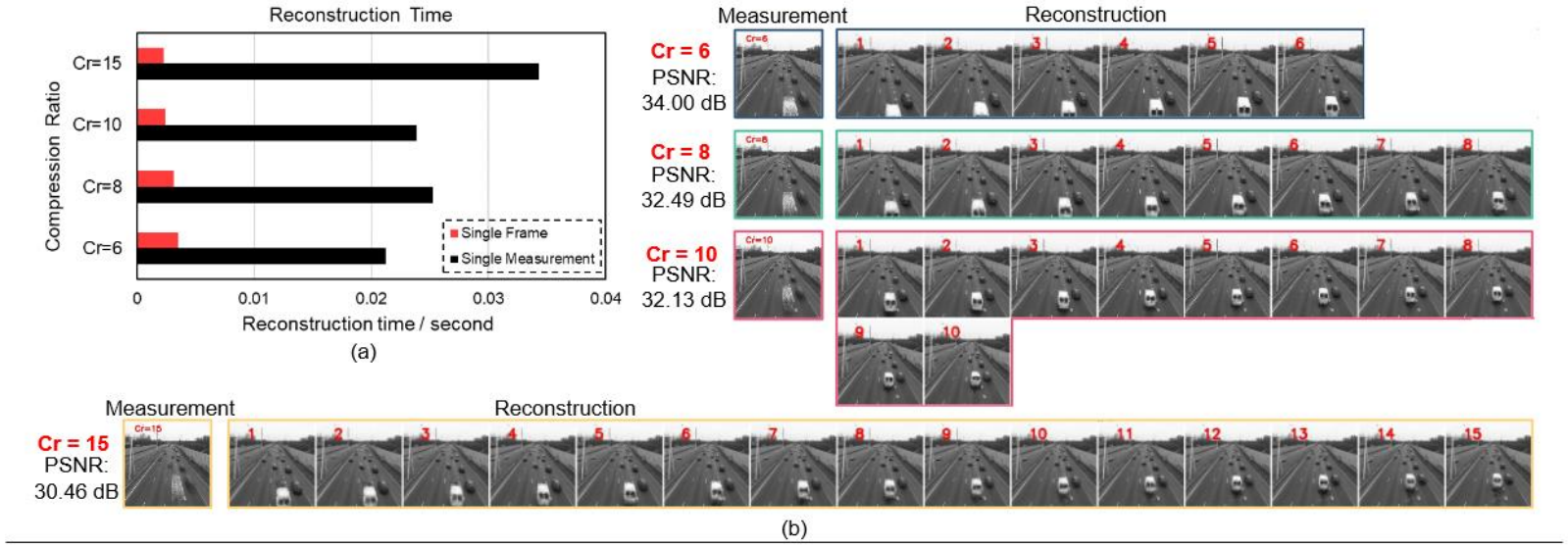


Fig. 7. (a): Reconstructed time of a single measurement and a single frame, (b): Reconstruction results with different compression ratio for the same scene.

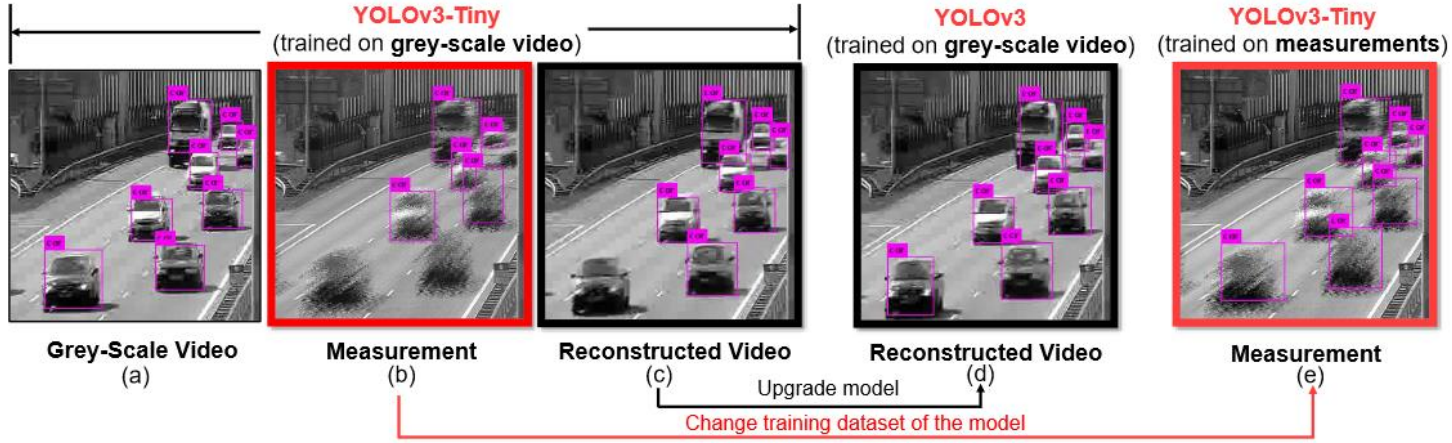
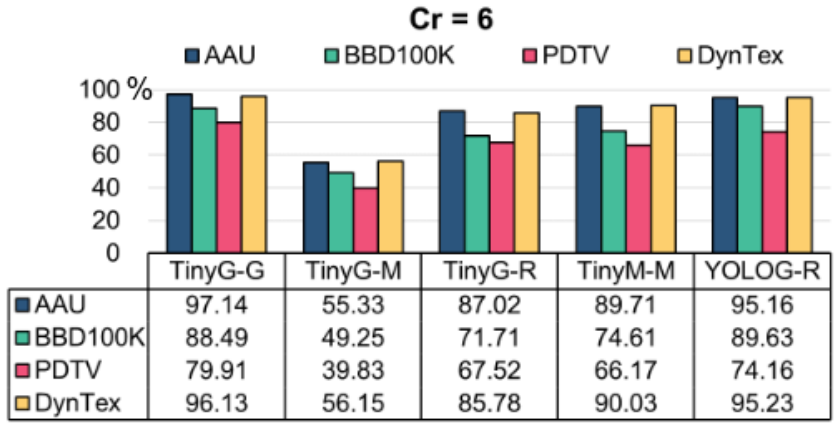


Fig. 8. Vehicle detection results on grey-scale video, measurements, and reconstructed video.

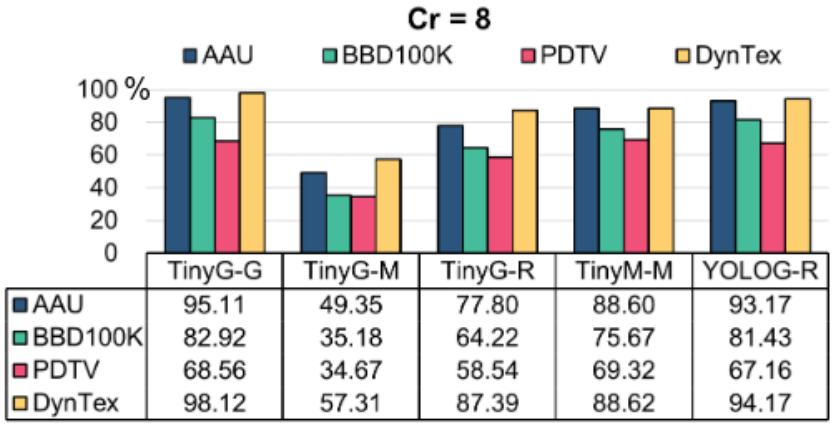
Intuition: Tiny does better trained on what it is prediction & YOLO > YOLO tiny

Intuition:

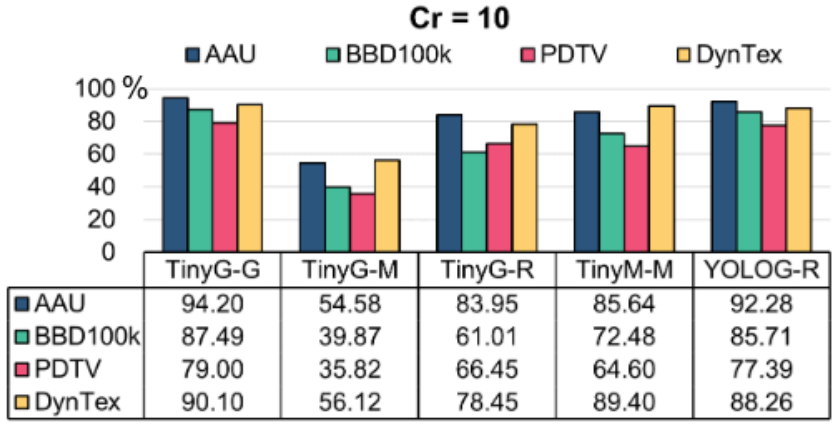
- Lower CR, better prediction
- Model does better when predicting on what it was trained on
- Highest average accuracy is tiny model trained on original data predicting original data… whoops!



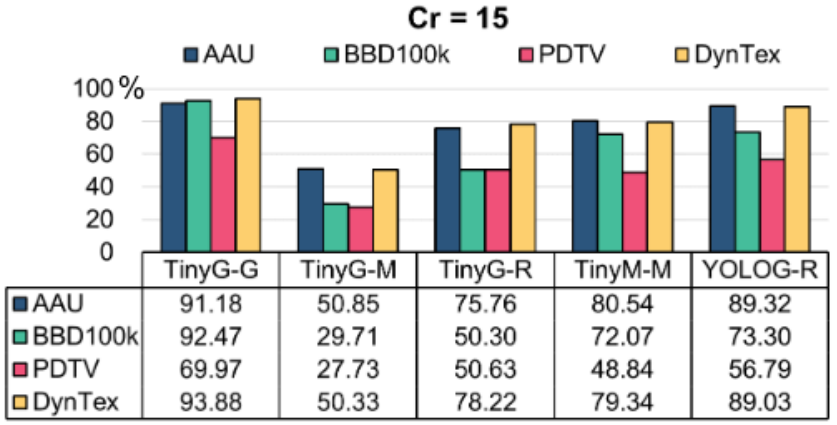
	TinyG-G	TinyG-M	TinyG-R	TinyM-M	YOLOG-R
AAU	97.14	55.33	87.02	89.71	95.16
BBD100K	88.49	49.25	71.71	74.61	89.63
PDTV	79.91	39.83	67.52	66.17	74.16
DynTex	96.13	56.15	85.78	90.03	95.23



	TinyG-G	TinyG-M	TinyG-R	TinyM-M	YOLOG-R
AAU	95.11	49.35	77.80	88.60	93.17
BBD100K	82.92	35.18	64.22	75.67	81.43
PDTV	68.56	34.67	58.54	69.32	67.16
DynTex	98.12	57.31	87.39	88.62	94.17



	TinyG-G	TinyG-M	TinyG-R	TinyM-M	YOLOG-R
AAU	94.20	54.58	83.95	85.64	92.28
BBD100k	87.49	39.87	61.01	72.48	85.71
PDTV	79.00	35.82	66.45	64.60	77.39
DynTex	90.10	56.12	78.45	89.40	88.26



	TinyG-G	TinyG-M	TinyG-R	TinyM-M	YOLOG-R
AAU	91.18	50.85	75.76	80.54	89.32
BBD100k	92.47	29.71	50.30	72.07	73.30
PDTV	69.97	27.73	50.63	48.84	56.79
DynTex	93.88	50.33	78.22	79.34	89.03

Author Observations

1. YOLO should be trained on what it will be predicting
    1. E.g. Train on compressed images if detecting on them
2. TinyM-M predicted just as well as TinyM-R, thus there is NOT a need to decompress the image before prediction
3. There exists a real tradeoff between CR and accuracy
4. PDTV predicts the worst because lowest quality cameras and cars are far away
5. AAU data set predicted well, they claim this means that they can predict in inclement weather
6. The BBDK100 data set suffered accuracy because it is harder to predict cars from the perspective of a car vs. static traffic camera
    1. Buttttt their whole project was for this to be deployed on a car…

Where is the energy data?

TABLE III
FPS OF THREE MODELS ON TWO HETEROGENEOUS HARDWARE PLATFORMS.

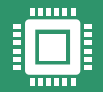
	YOLOG	TinyG	TinyM
NVIDIA GPU Workstation	67.55	336.93	334.11
Intel FRD	2.15	13.89	13.47

- They had two goals… reduce the time to prediction and decrease energy consumption
- This image shows that the prediction time is faster for the light weight YOLOG tiny, but it also shows that TinyG is the same as TinyM basically
  - Meaning if they just deployed the YOLOGv3-Tiny on the edge trained by the gray scale images they would have achieved better accuracy and the same prediction time
- So maybe their argument is for energy savings?
  - Need figures for this energy saving!!!!
  - There could be an argument for the bandwidth reduction for compressed sending to the edge, but they showed that YOLOv3-TINY predicted on the CAV as well as YOLOv3 on the edge

Questions

What was the real purpose of sending prediction results from edge-server back to the CAV when the cloud was updating the YOLOv3 algorithm?

Why did they not test this on a real CI Camera? Is the technology not widely available?



By not simulating driving conditions with network connectivity related issues are they missing a key research question on all of this?

Where is the energy data?!?!?!?!?!??!