Litedge: Towards Light-weight Edge Computing for Efficient Wireless Surveillance System

Yutong Liu[1], Linghe Kong[1], Muhammad Hassan[1], Long Cheng[2], Guangtao Xue[1], Guihai Chen[1]

[1] Shanghai Jiao Tong University, China

[2] Clemson University, USA

Wireless VS. Wired surveillance deployment

Choice!!!

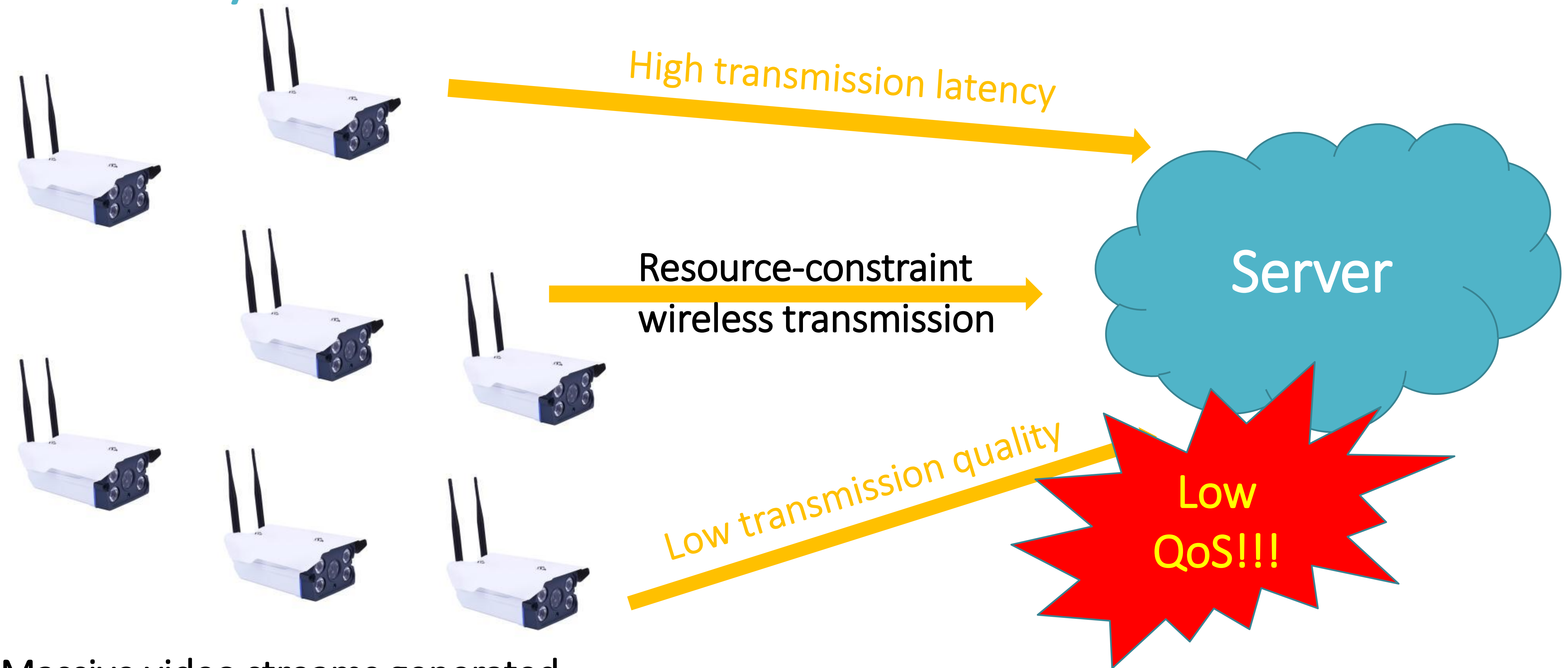
Wired surveillance system:

Massive cables,
inflexible deployment,
difficult and expensive maintenance.

Wireless surveillance sy

Much **less** cables,
flexible deployment,
easy and cheap maintenance.

Heavy wireless transmission burden

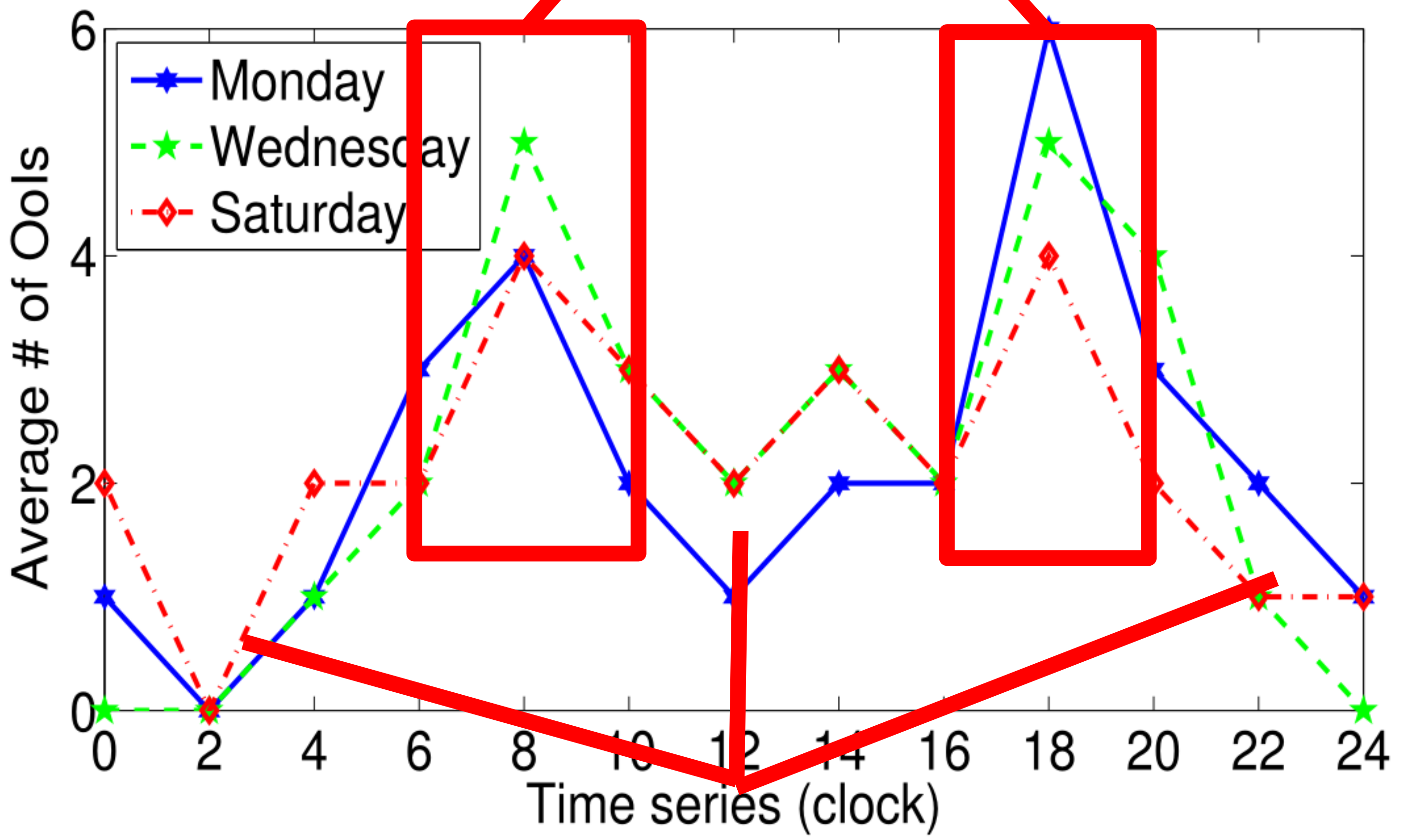


Massive video streams generated

Observations

Only 6 clocks have more than 1 object of interest (OoI) in videos.

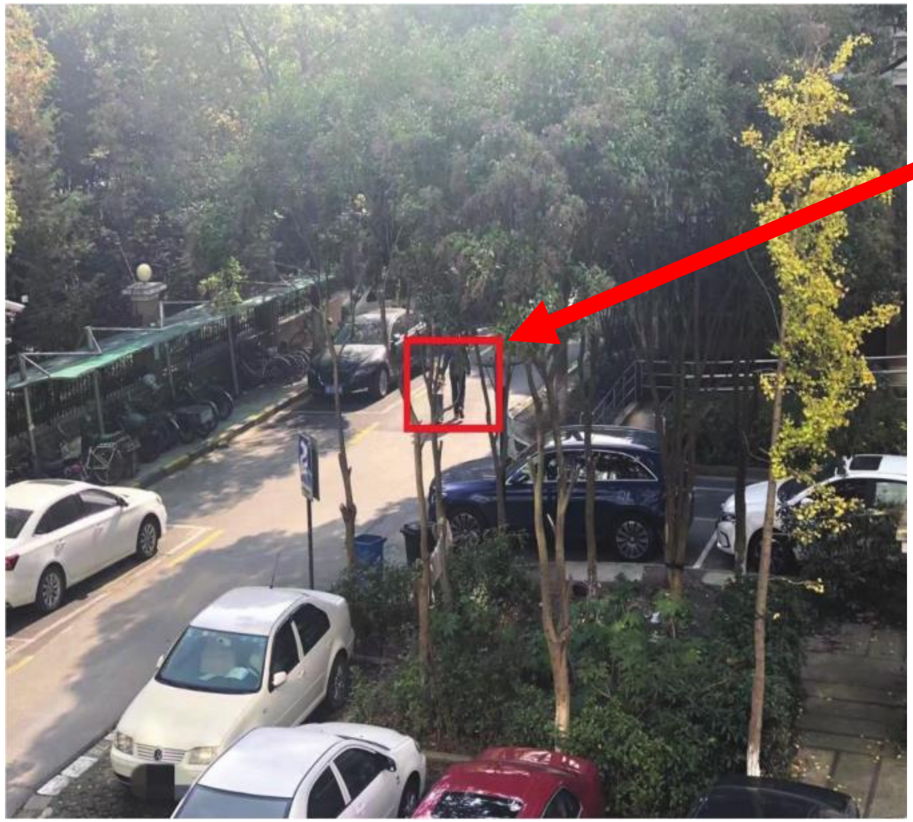
1. Large redundancy exists in collected surveillance videos.



Redundancy

Observations

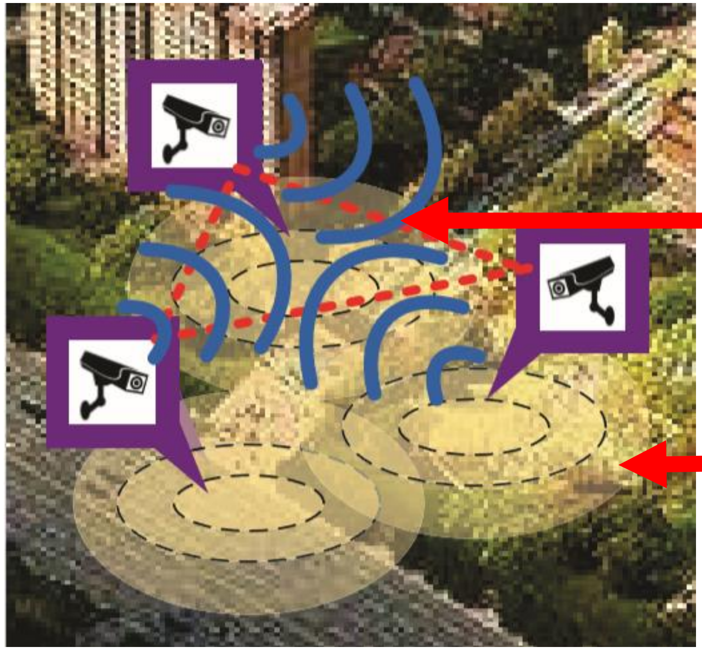
2. Environment shielding introduces error in frame analysis.



The tree shielding on two monitored persons.

Observations

3. Surveillance cameras have neighboring deployment and allow information sharing.



Collaborative communication

Overlapping monitoring area

Intuitions

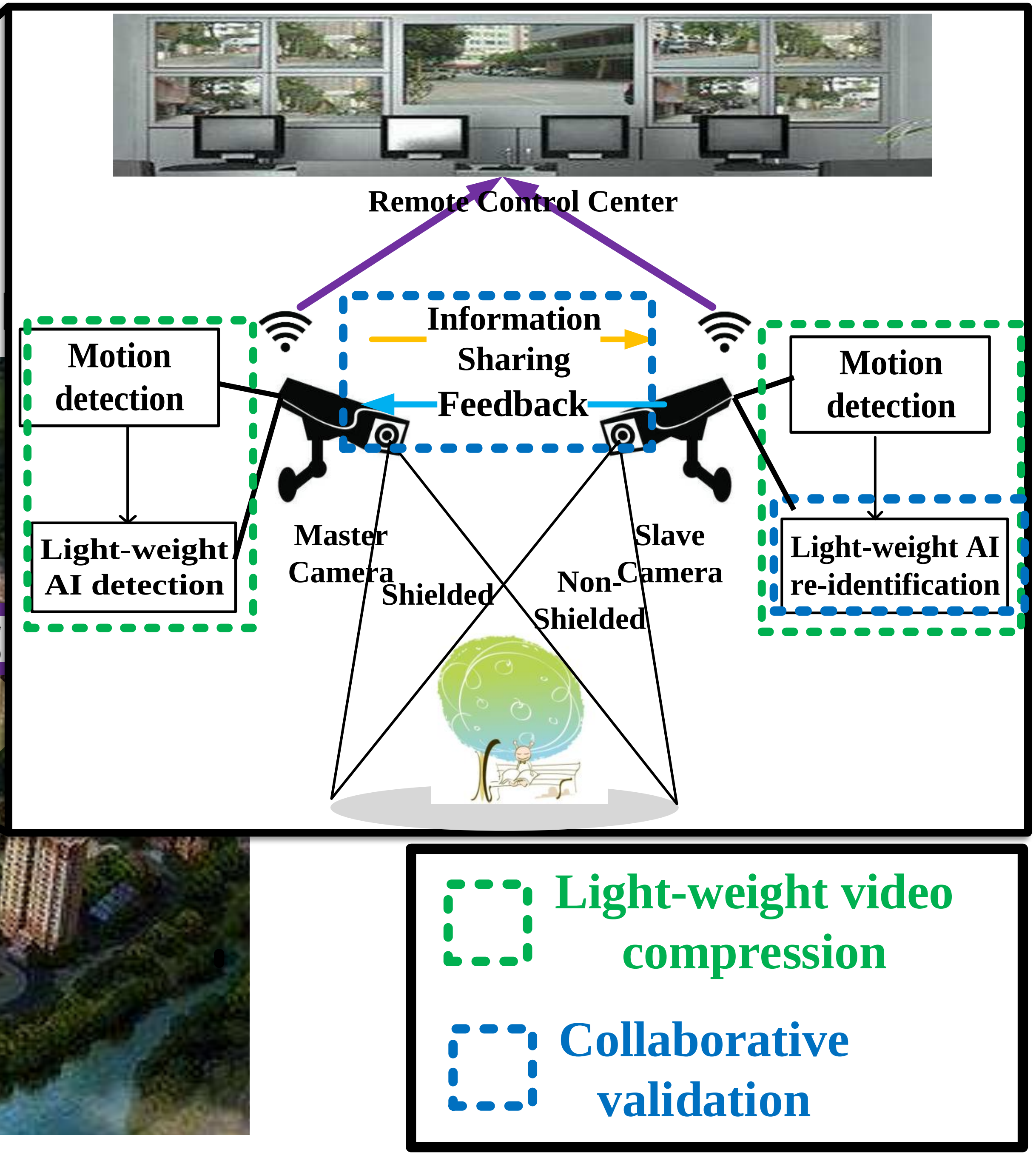
Local-processing on cameras:

Type	Method	Advantage	Disadvantage
1. Frame filtering	Wu et al. [1]	Fast	Coarse filtering
	Zhang et al. [2]	Accurate	Long processing latency
2. Frame compression	Mitchell et al. [3]	Standard	Relatively less redundancy reduction
3. Rate deduction	Zhang et al. [4]	Simple	Too naive
4. Collaborative computation among cameras	Collins et al. [5]	First	Not utilized for error compensation
	Natarajan et al. [6]	Scheme proposal	A generalized inspiration
OURS	Light-weight frame filtering & collaborative error compensation	Fast and accurate	Future work

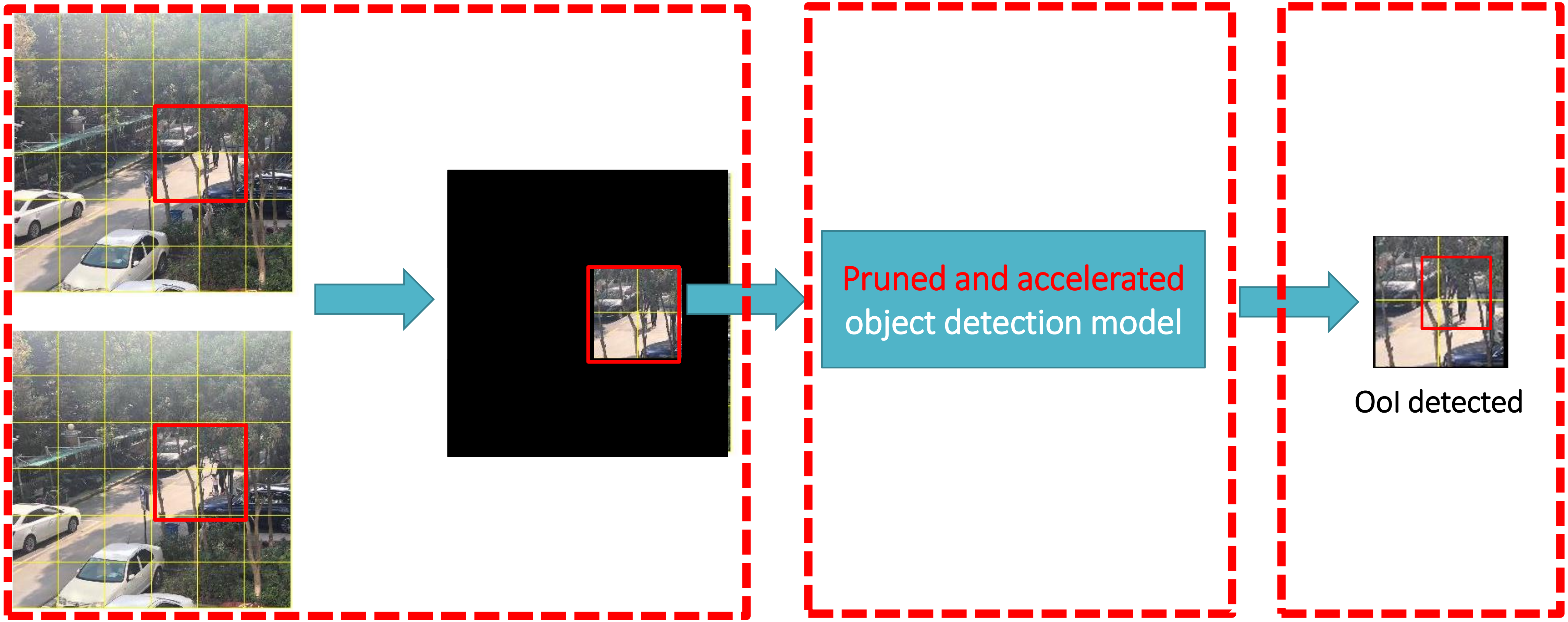
Litedge’s Architecture

A **light-weight edge** computing scheme in wire

Surveillance cameras distribution diagram



Light-weight video compression

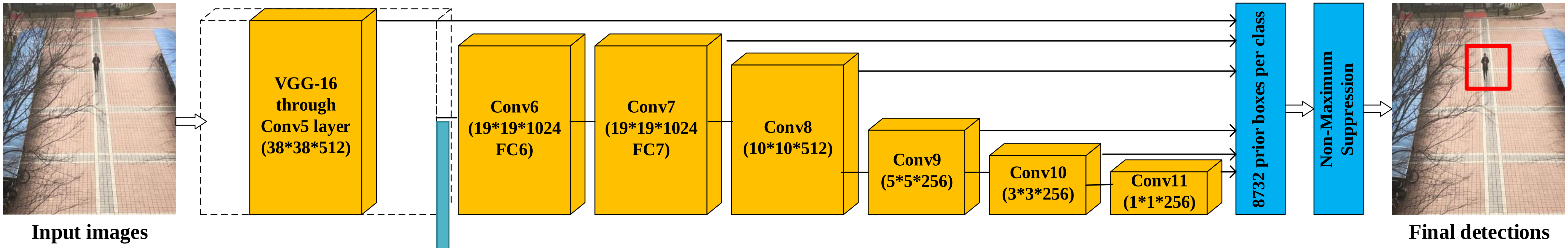


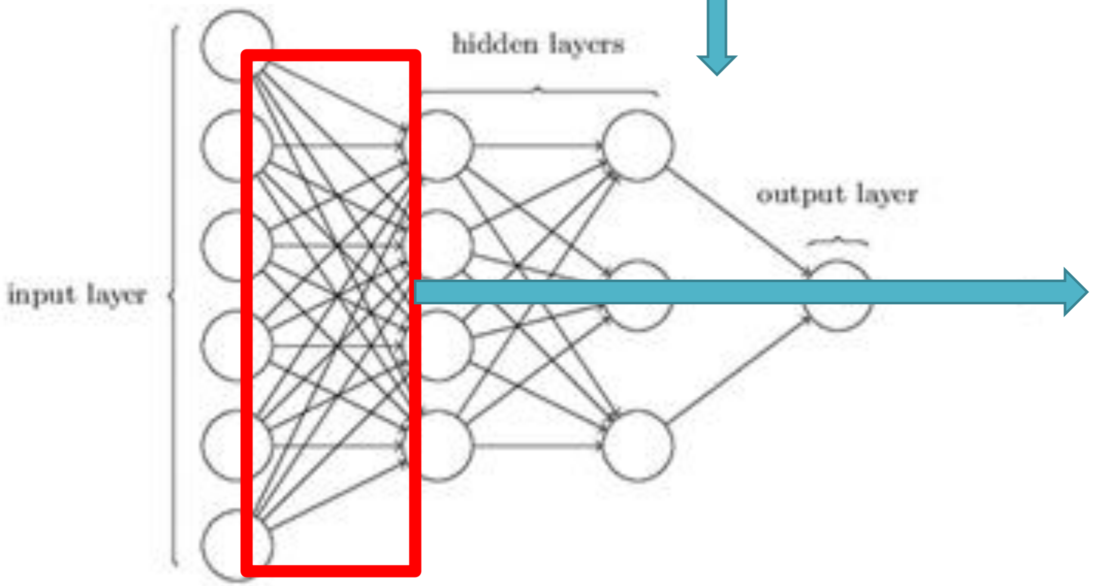
Input: Frame pruned

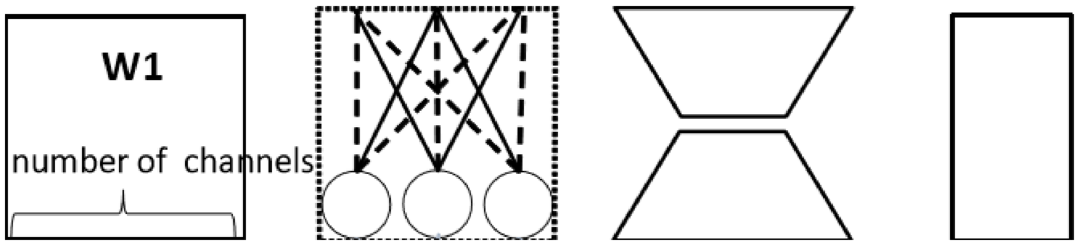
Model: model pruning and convolution acceleration

Output: class deduction

Model pruning

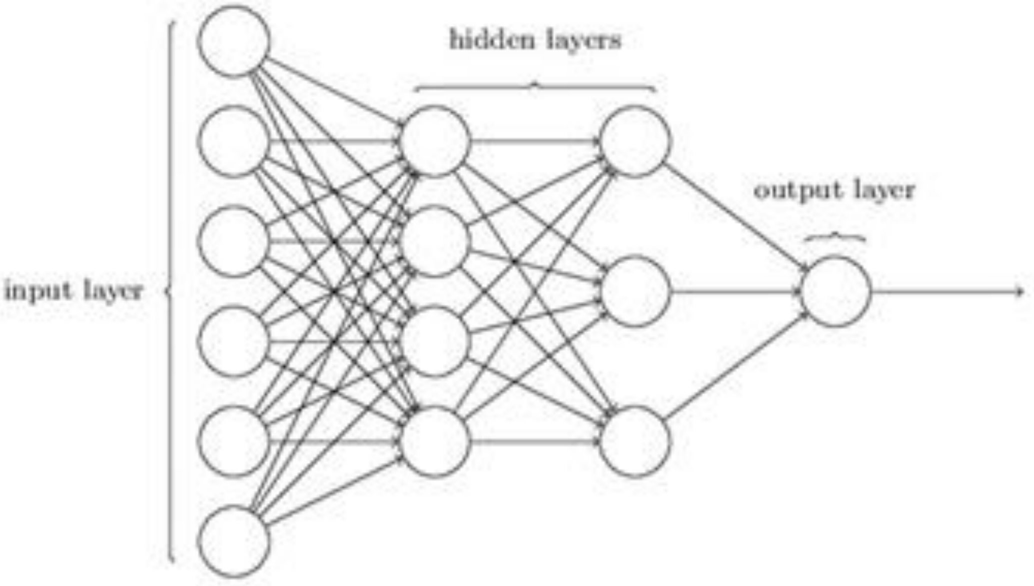






Original → Channel selection → Channel reconstruction

Channel selection



To keep the reconstruction error as small as possible, select the most representative channels:

$$\arg\min_{\beta, W} \frac{1}{2N} \left\| Y' - \sum_{i=1}^{c} \beta_i X_i W_i^T \right\|_F^2$$
$$\text{subject to } \|\beta\|_0 \leq c'.$$

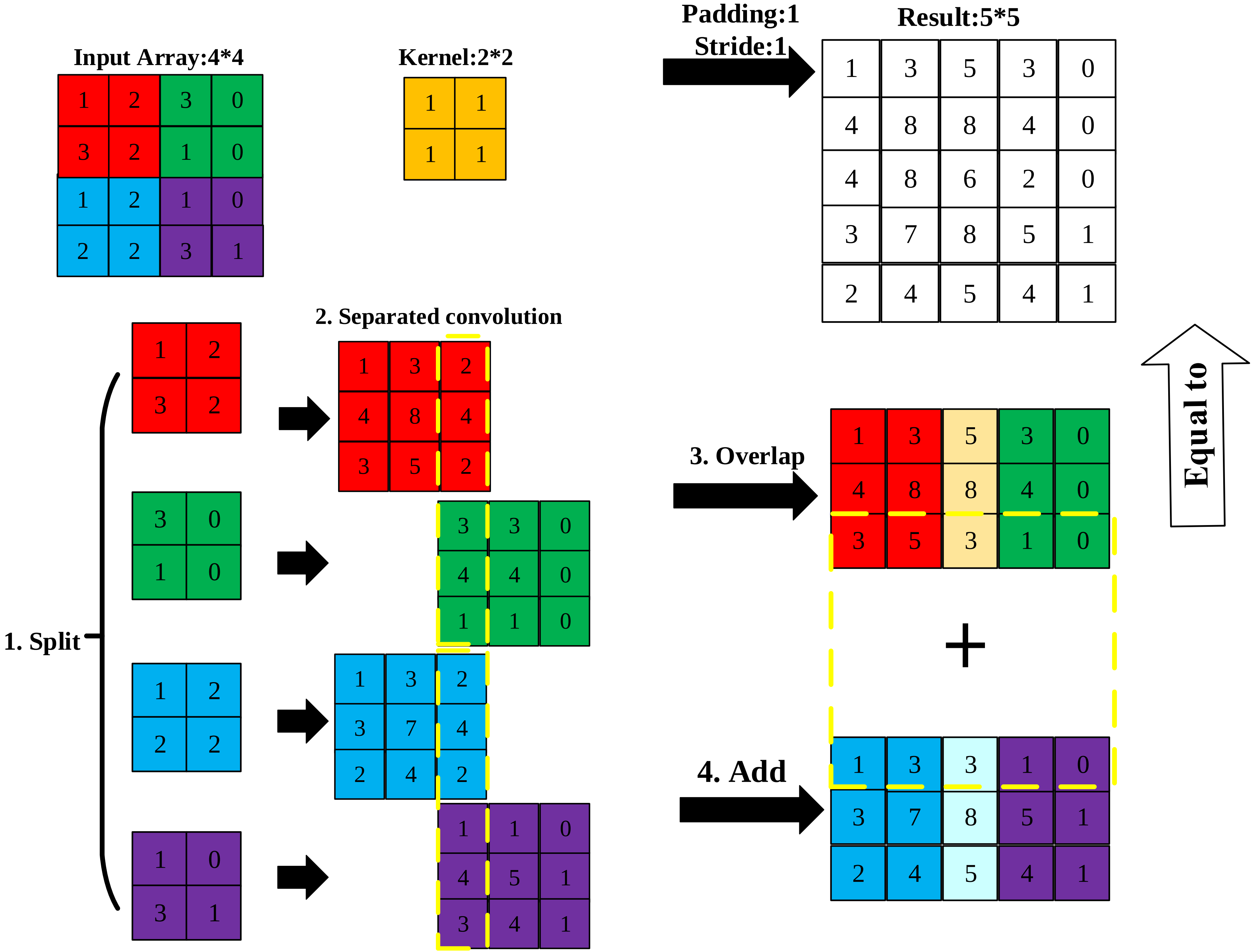
Channels: c
Convolutional filter W: n × c × k_h × k_w
Input volumes X: N × c × k_h × k_w
Output matrix Y: N × n

β_i represents the status of channel i (i.e., selected or not selected),
X_i represents new input which prunes channel i from X,
and W_i represents new filters which prunes channel i from W
Channel selection: fix W
Reconstruction: fix β

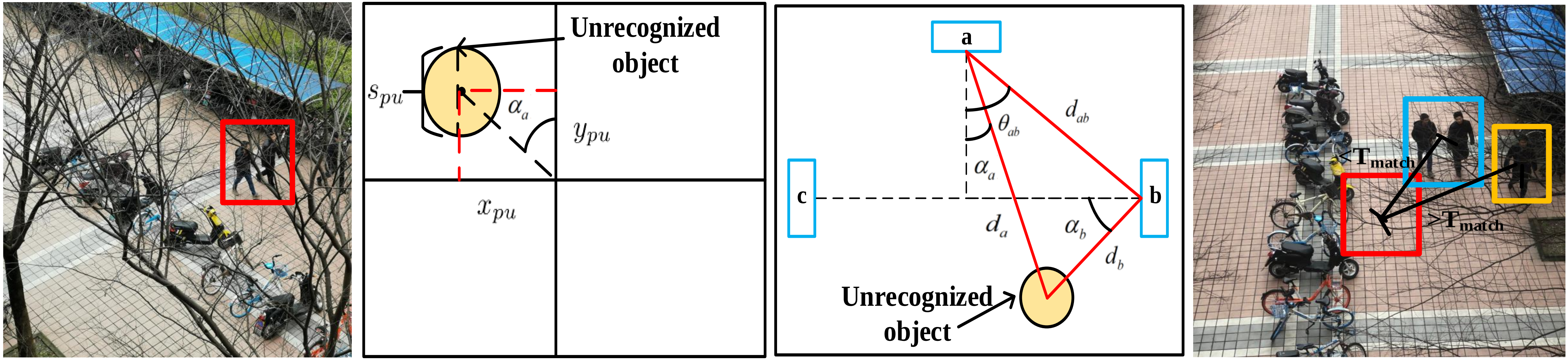
Convolutional acceleration

Convolutional method	Computational complexity
Simple dot product	$O(M^2m^2)$
FFT	$O(M^2\log_2 M)$
Overlap-and-Add (OaA)	$O(M^2\log_2 m)$
Note: Input size M*M, kernel size m*m	

Choice!!!


Input Array:4*4
Kernel:2*2
Padding:1
Stride:1
Result:5*5
1. Split
2. Separated convolution
3. Overlap
4. Add
+
Equal to

Collaborative validation

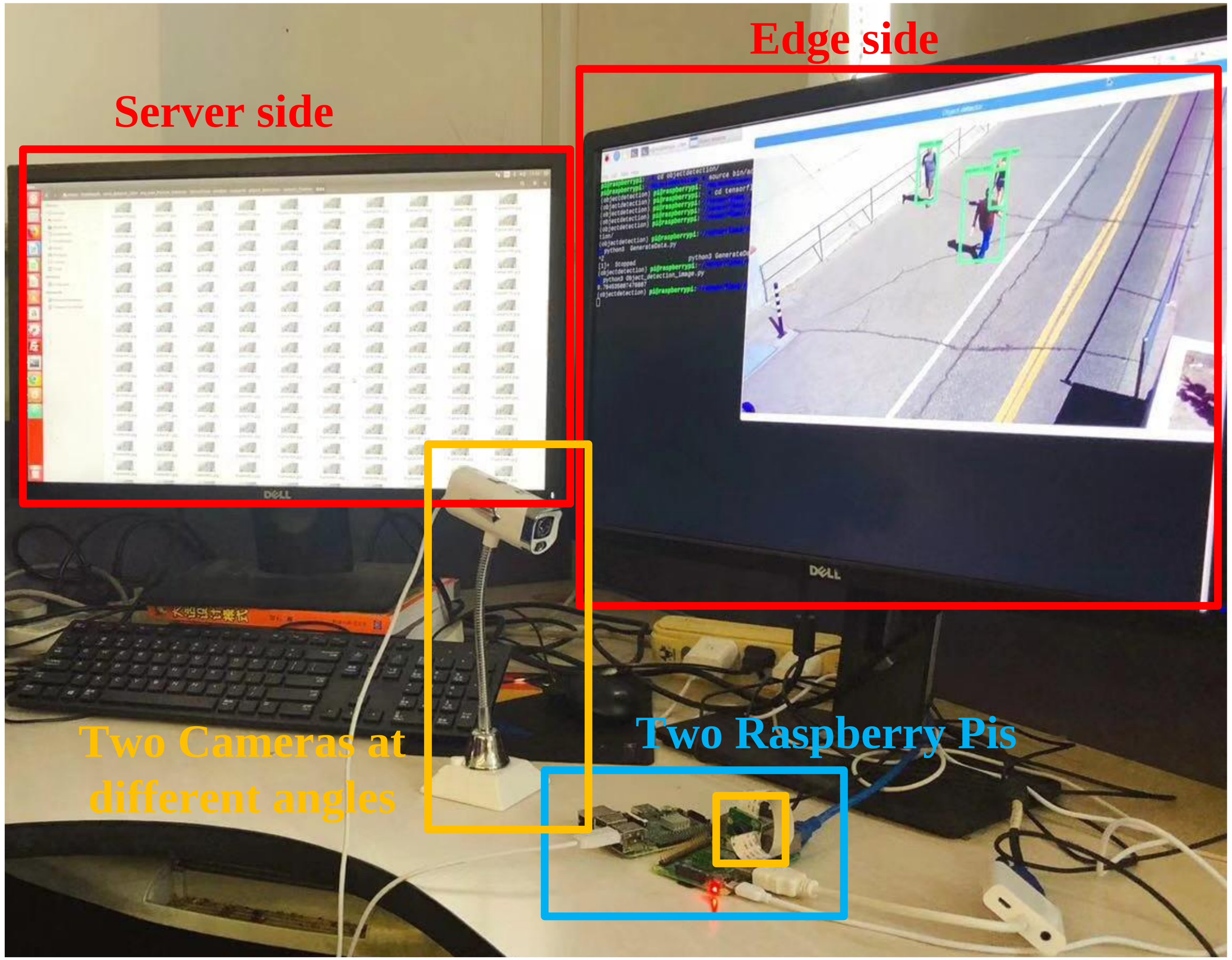


$$d_a = \frac{h_u \times f}{s_{pu} + h_a}$$

$$d_b = \sqrt{(d_a \sin(\theta_{ab} - \alpha_a))^2 + (d_{ab} - d_a \cos(\theta_{ab} - \alpha_a))^2}$$

$$\alpha_b = \arccos \frac{d_{ab} - d_a \cos(\theta_{ab} - \alpha_a)}{d_b}$$

Implementation



- **Surveillance cameras:**

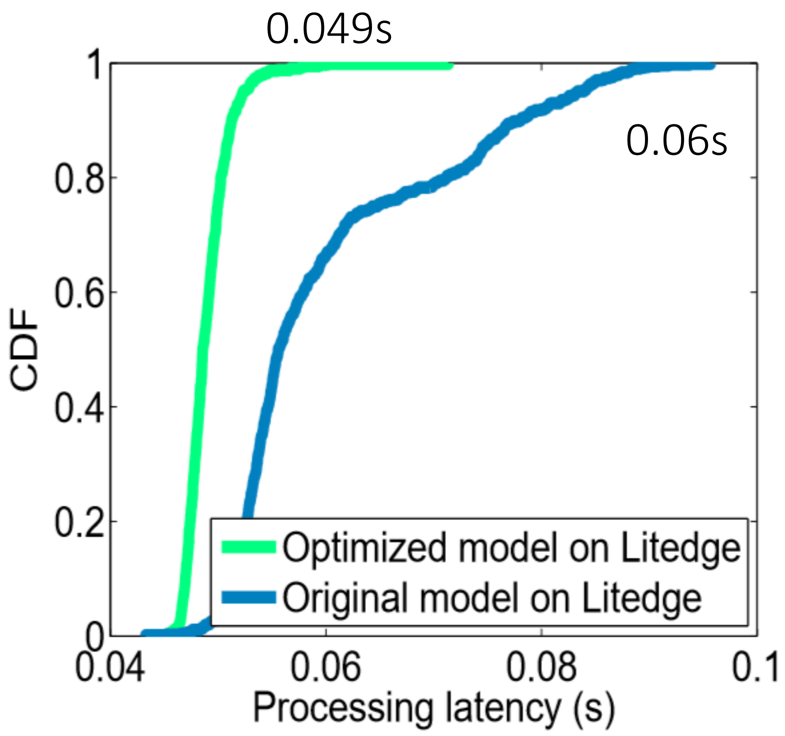
Raspberry Pi 3b embedded with ARM Cortex-A53 and no GPU. Supported by 802.11n WiFi module and external camera module v2

- **Server:**

64-bit Ubuntu 14.04 LTS version, 8 core 3.6GHz Intel Core i7 CPU and Kabylake GT2 770 GPU.

QoS evaluations

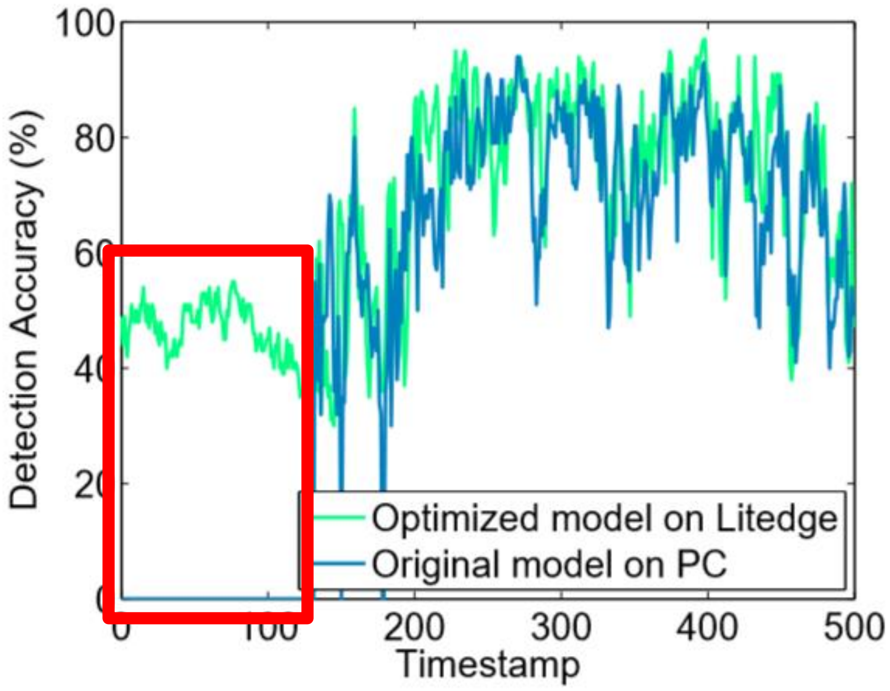
1. Latency evaluation



Method	Original	MPEG-4	Litedge
Video size	325M	133M	96.82M
Transmission Latency	676s	296.64s	121.62s
Reduction Ratio	-	56.1%	82%

QoS evaluations

2. Accuracy evaluations

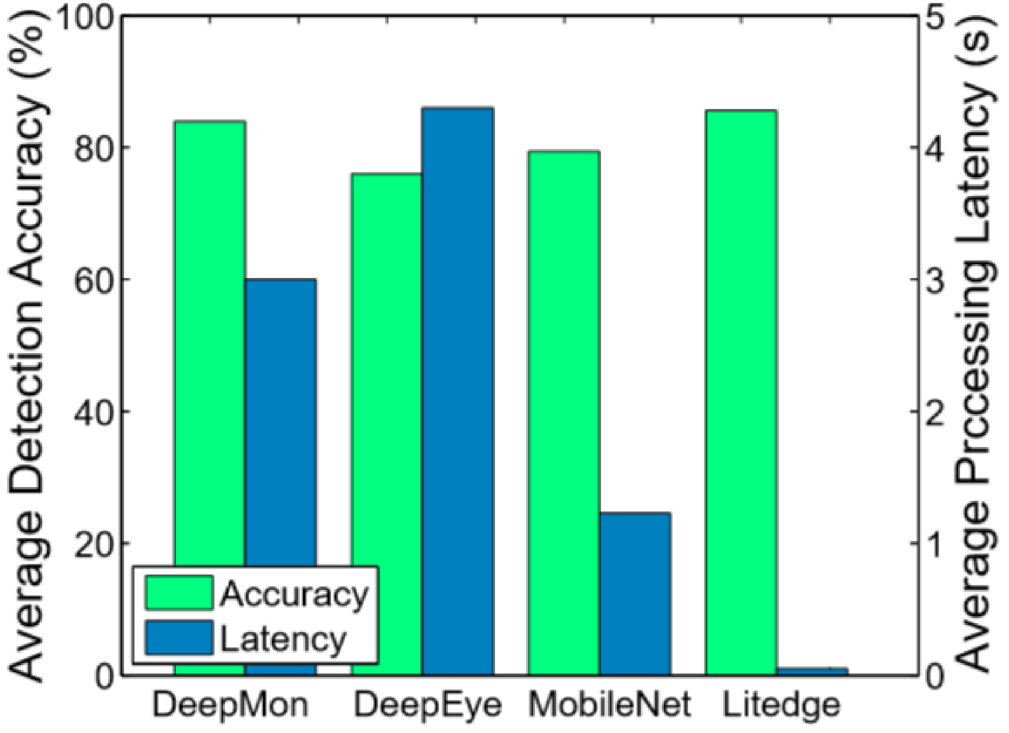


$$Acc_{Litedge} = \frac{f_{AI}}{f_{real}} + R_{com} = \frac{f_{AI} + f_{val}}{f_{real}}$$

QoS evaluation

3. Overall evaluation

- DeepMon [7]: cache-based convolutional optimization and tensor decomposition
- DeepEye [8]: memory caching and SVD-based model compression
- MobileNet [9]: depth-wise separable convolution, combined by a single-filter derived convolution and 1*1 pointwise convolution



Method	Original Transmission (Ground Truth)		Compression Only (Control experiment)		Litedge (Our design)		Zhang et al. [5] (Related method)	
Performances	Accuracy	Latency	Accuracy	Latency	Accuracy	Latency	Accuracy	Latency
Results	93%	676s	85.6%	247.62s	91.28%	262.62s	90%	329.88s
Balance ratio	0.138		0.346		**0.348**		0.273	

Conclusion

- ➢Light-weight video compression
- ➢Collaborative validation on cameras
- ➢Surveillance system implementation and evaluation

Possible future directions:

- ➢Low illumination and bad weather effects should be eliminated.
- ➢Distortion rate should be further decreased by proper video compression.
- ➢Monitoring scenarios and camera types can be extended.

Thanks for listening!

Yutong Liu
isabelleliu@sjtu.edu.cn

Reference

[1] D. Wu, C. Song, H. Luo, Y. Ye, H. Wang, Video surveillance over wireless sensor and actuator networks using active cameras, IEEE Transactions on Automatic Control 56 (10) (2011) 2467–2472.

[2] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, S. Banerjee, The design and implementation of a wireless video surveillance system, in: ACM/IEEE Mobicom, Paris, France, 2015, pp. 426–438.

[3] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, D. J. Legall, Mpeg video compression standard 99 (5) (1996) 1666–1675.

[4] Y. Zhang, H. Wang, D. Zhao, Up-sampling dependent frame rate reduction for low bit-rate video coding, in: DCC, Snowbird, Utah, USA, 2011, p. 489.

[5] R. T. Collins, A. J. Lipton, H. Fujiyoshi, T. Kanade, Algorithms for cooperative multisensor surveillance, Proceedings of the IEEE 89 (10) (2001) 1456–1477.

[6]P. Natarajan, P. K. Atrey, M. S. Kankanhalli, Multi-camera coordination and control in surveillance systems: A survey, TOMCCAP 11 (4) (2015) 57:1–57:30.

[7] H. N. Loc, Y. Lee, R. K. Balan, Deepmon: Mobile gpu-based deep learning framework for continuous vision applications, in: MobiSys, Niagara Falls, NY, USA, 2017, pp. 82–95.

[8]A. Mathur, N. D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, F. Kawsar, Deepeye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware, in: MobiSys, Niagara Falls, NY, USA, 2017, pp. 68–81.

[9]A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, CoRR abs/1704.04861.