Reinforcement Learning for Adaptive Video Compressive Sensing

SIDI LU[*], Department of Computer Science, Wayne State University, USA
XIN YUAN, Westlake University, China
AGGELOS K KATSAGGELOS, Department of Electrical and Computer Engineering, Northwestern University, USA
WEISONG SHI, Department of Computer and Information Sciences, University of Delaware, USA

We apply reinforcement learning to video compressive sensing to adapt the compression ratio. Specifically, video snapshot compressive imaging (SCI), which captures high-speed video using a low-speed camera is considered in this work, in which multiple (B) video frames can be reconstructed from a snapshot measurement. One research gap in previous studies is how to adapt B in the video SCI system for different scenes. In this paper, we fill this gap utilizing reinforcement learning (RL). An RL model, as well as various convolutional neural networks for reconstruction, are learned to achieve adaptive sensing of video SCI systems. Furthermore, the performance of an object detection network using directly the video SCI measurements *without reconstruction* is also used to perform RL-based adaptive video compressive sensing. Our proposed adaptive SCI method can thus be implemented in low cost and real time. Our work takes the technology one step further towards real applications of video SCI.

CCS Concepts: • **Computing methodologies → Image compression**; **Computer vision**; *Reinforcement learning*.

Additional Key Words and Phrases: Image processing, compressive sensing, reinforcement learning

1 INTRODUCTION

Video compressive sensing is a promising technique inspired by compressive sensing (CS) [4, 7], where multiple temporal video frames are mapped into a single measurement (i.e., a small number of linear projections of the original video image data). We consider the snapshot compressive imaging (SCI) [13, 19, 49], which uses a two-dimensional (2D) detector to sample the high-dimensional data (such as high-speed video [21] and hyperspectral images [26]) and output measurements). The underlying principle of video SCI is to modulate the high-speed video with a higher frequency than the sampling rate of the camera [10, 21, 32]. In this manner, video SCI can utilize a low-speed camera to capture high-speed videos.

Most recently, by using deep learning (DL) algorithms [5, 6, 42] for real-time reconstruction, end-to-end sampling and reconstruction video SCI systems have been built [31]. In the meanwhile, recent work has demonstrated the effectiveness of SCI cameras in real-world applications. For example, [23] demonstrated that the detection accuracy of the measurements (*i.e.*, compressed

Authors' addresses: Sidi Lu, lu.sidi@wayne.edu, Department of Computer Science, Wayne State University, Detroit, Michigan, USA, 48202; Xin Yuan, Westlake University, Hangzhou, Zhejiang, China, xyuan@westlake.edu.cn; Aggelos K Katsaggelos, Department of Electrical and Computer Engineering, Northwestern University, Evanston, Illinois, USA, aggk@eecs.northwestern.edu; Weisong Shi, Department of Computer and Information Sciences, University of Delaware, Newark, Delaware, USA, weisong@udel.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Association for Computing Machinery.

0004-5411/2023/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

video images) generated by the SCI camera could achieve a satisfactory accuracy level (*i.e.*, close to the accuracy on reconstructed videos and comparable to the true value), which paves the way of applying SCI in connected and autonomous vehicles (CAVs), *e.g.*, conducting measurement-based object detection with the objective of *i*) detection speed acceleration and *ii*) bandwidth reduction by reducing the transmission volume of detection results between CAVs and roadside-units.

However, as to the CAVs, since the driving scenes being captured are dynamic and vehicle speed varies over time, it is imperative to realize *adaptive video SCI* for real-world applications, *i.e.*, *automatically determining the optimal B under different application environment*. Take the measurement-based object detection as an example, when the vehicle is driving under a slow-motion scenario, increasing B (B refers to the compression ratio) can further accelerate inference speed while still guaranteeing a high measurement-based detection accuracy. In contrast, if the driving environment changes rapidly, *e.g.*, a red traffic light or an accident suddenly halt all vehicles, B should be decreased to avoid missing high-speed information. Therefore, it is the right time to take the developments one step further and make SCI systems suitable for real applications.

Bearing this concern in mind, this paper considers the video SCI system from the perspective of adaptive sensing [51]. This is motivated by real applications, as scenes are dynamic, of various backgrounds and speeds and thus different compression ratios should be used. Moreover, the compression ratio should be *adaptively* adjusted for different scenes or as the contents in the scene change. In this paper, we address this challenge by *reinforcement learning* (RL) [40]. Specifically, we treat the video SCI system as an *agent* and the scene being captured as the *environment*. By developing the *policy* and *reward*, we build an end-to-end RL-based adaptive video SCI system.

1.1 Video Compressive Sensing

As depicted in Fig. 1 (top-middle), for a high-speed video with B frames $\mathbf{X} \in \mathbb{R}^{N_x \times N_y \times B}$, a different mask (coding pattern) $\mathbf{C} \in \mathbb{R}^{N_x \times N_y \times B}$ is imposed on each of them, and then these modulated frames are summed into a single measurement $\mathbf{Y} \in \mathbb{R}^{N_x \times N_y}$, and we define B as the compression ratio. Here, the coding pattern is a random matrix which consists of zeros and ones. This process can be recognized as a *hardware encoder* and the key ingredient is the high-speed modulation. Different approaches have been proposed in the literature, such as a shifting mask [15, 21] or a digital micromirror device [32, 39], to achieve this modulation.

The other important part of video SCI is the *software decoder*, or the inverse algorithms, to reconstruct the high-speed video from the compressed measurement given the masks [49]. For a long time, the reconstruction algorithm was the bottleneck precluding the wide applications of video SCI. In the literature, diverse optimization methods developed for CS have been used [2, 44–46, 48]. It is only in the last few years that the quality of the reconstructed videos has been significantly improved and they can be used in our daily life [19]. One common drawback of these model-based optimization methods is the slow reconstruction speed. Most recently, this drawback has been ameliorated by DL neural networks [6, 12, 25, 31], and has led to high-speed high-quality reconstructions. In short, the hardware encoder and DL based software decoder have now paved the way of end-to-end video SCI systems to be used in our daily life [24].

1.2 Temporal Adaptive Sensing in Video SCI

From the application perspective, to deploy video SCI systems into our daily life, different settings are required for different scenes. Taking video surveillance as an example, video SCI cameras can significantly reduce memory and transmission bandwidth costs, while, recovering the high-speed video if needed. However, a fixed compression ratio (B:1) is clearly not optimal in this case, since when there are no moving objects in the scene, a large B can be used, while when a high-speed object exists in the scene, a small B is desired for maintaining high quality reconstruction (Fig. 1).

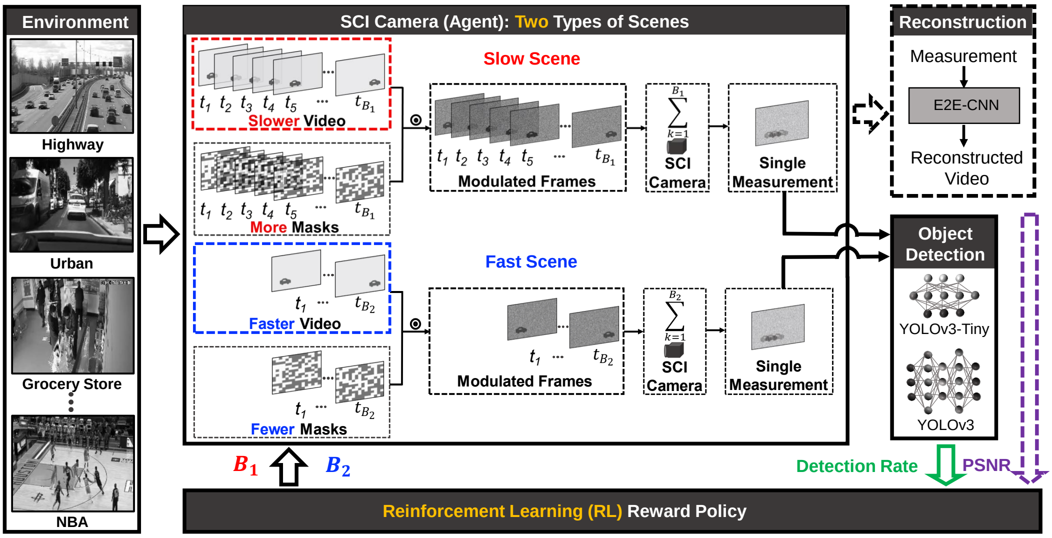


Fig. 1. The framework of video Snapshot Compressive Imaging (SCI) and reinforcement learning (RL) for temporal adaptive sensing. SCI cameras (middle-top) are used to capture and thus sense the environment (left) with an adaptive compression ratio (B) determined by the RL policy (bottom-middle). In SCI, every B frames are *compressed* to a single measurement, which is sent to the *object detection* module (bottom-right, YOLOv3 [34] and YOLOv3-Tiny [43] are used here) directly; optionally, the measurement can also be sent to the *reconstruction* module (top-right) to perform video recovery. The end-to-end convolutional neural network, named E2E-CNN [31], is used to reconstruct high-speed video frames from a single measurement. The detection rate and optionally the PSNR of the reconstructed video (available during training) are sent to the RL module to adjust B for different scenes. Here, $\odot$ denotes the element-wise product. B_1 is large for a slow motion scenes while B_2 is small for a high-speed motion scenes. Note that only one B (B_1 or B_2 as in the two examples) is the output of the RL module at each time step. Only *a single* SCI camera is used, instead of two camera agents, to capture slower and faster scenes. The goal of distinguishing slower and faster parts of a scene in this plot is to highlight that our work can *adapt the compression ratio (B)* by the proposed RL policy.

Moreover, we expect that the video SCI system can adjust this B value automatically. This is what we refer to as *temporal adaptive sensing*[1] and we aim to address it by RL (Fig. 1 bottom) in this paper.

1.3 Related Work

Although the idea of adaptive CS has been proposed for a long time, in most cases it applies to spatial CS, *i.e.*, following the single pixel camera architecture [8]. By contrast, for adaptive sensing in video CS considered in this paper, only a few papers exist and the one closely related to ours is [51], which considers the same problem but by using a motion estimation method to adapt B. However, both the reconstruction algorithm and the adaptive sensing framework developed therein produce low quality results. During the past eight years, the reconstruction algorithms of video CS have been improved significantly, especially the ones based on DL [6, 31, 50]. Moreover, the look-up table used in [51] only connected adaptive temporal sensing with motion estimation and

[1] The other proposal to adaptive sensing is to adjust the compression ratio spatially as a function of content (different places on the image plane). However, this will pose a significant challenge for the hardware design and thus we do not consider it here.

did not consider the scene complexity and object detection rate, which are important factors for the adaptive framework developed in this paper.

1.4 Reinforcement Learning

Reinforcement learning [14, 28, 40, 52] is an online algorithm designed to optimize behavioral strategies in sequential decision problems [18], wherein agents continuously interact with unknown environments and seek behavioral policies to maximize the expected cumulative reward. Many challenging benchmark tasks can be performed in this framework, such as robotics [16, 17], high-dimensional continuous control simulations [17, 36], the game of Go [38], Atari [27], and competitive video games [37, 41].

An RL agent uses a policy to control its behavior, where the policy is a mapping from obtained inputs to actions. One main difference between RL and supervised learning is that the RL agent is never told the optimal action, instead, it receives an evaluation signal indicating the goodness of the selected action. This matches well with an adaptive video CS considered in this work, where the SCI camera usually does not know the environment and the objects in the scene being captured are dynamic and their speed can vary over time. Recently, Zhang *etal.* designed an online RL-based real-time video telephony system named OnRL to optimize video streaming applications [52]. A model-based RL for microservice resource allocation over scientific workflows was proposed in [47], and [1] leverage RL approach to high QoE video streaming over wireless networks. These works are designed to schedule the right clients for prioritization in a high-load scenario to outperform the status quo. While our paper mainly focuses on using RL to adapt video compression rate for the temporal compressive sensing, which can automatically compress video for the model inference and therefore reduce video transmission costs and accelerate video inference speed.

1.5 Contributions of This Paper

In this work, we revisit the temporal adaptive sensing problem in video CS by using three new modules: *i*) end-to-end convolutional neural network (E2E-CNN) [31] based reconstruction, *ii*) RL for adaptive sensing control, and *iii*) edge compression based applications [24] by conducting object detection directly on the video SCI measurements without reconstruction. Our new regime brings video SCI closer to real applications, such as, connected and autonomous vehicles.

Remarkably, previous work [24] proved that the object detection accuracy utilizing the compressed measurements (without reconstructing the high-speed video) was close to the one obtained using the original video. Therefore, the advantage of using SCI is clear since it accelerates inference by performing *measurement-based object detection*. However, there is a non-negligible trade-off between detection accuracy, reconstruction quality, and compression ratio, which hinders the real applications of measurement-based object detection across diverse fields significantly. In this context, the core innovation of our study is to provide actionable insights into solving a real application challenge by automatically determining the optimal compression ratio using RL, which can accelerate the deployment of SCI cameras and measurement-based object detection for time-sensitive applications.

The rest of this paper is organized as follows. Section 2 describes the proposed RL model for adaptive video CS. Extensive results are presented in Section 3 and Section 4 concludes the paper.

2 PROPOSED RL MODEL FOR ADAPTIVE VIDEO CS

In this section, we first describe the mathematical model of video SCI and briefly introduce the state-of-the-art deep learning based reconstruction approaches. The proposed RL based adaptive sensing is detailed in Sec. 2.3.

2.1 Mathematical Model of Video SCI

Following Fig. 1 (top-middle), a B-frame dynamic scene $\mathbf{X} \in \mathbb{R}^{N_x \times N_y \times B}$ is modulated by B fast updated masks $\mathbf{C} \in \mathbb{R}^{N_x \times N_y \times B}$, and then the modulated video frames are integrated into a single measurement frame $\mathbf{Y} \in \mathbb{R}^{N_x \times N_y}$ by a camera sensor with the exposure time of these B frames. This process can be expressed as

$$\mathbf{Y} = \sum_{b=1}^{B} \mathbf{C}_b \odot \mathbf{X}_b + \mathbf{Z}, \tag{1}$$

where $\mathbf{Z} \in \mathbb{R}^{N_x \times N_y}$ denotes noise, $\mathbf{C}_b = \mathbf{C}(:,:,b)$ and $\mathbf{X}_b = \mathbf{X}(:,:,b) \in \mathbb{R}^{N_x \times N_y}$ the b-th mask and the corresponding video frame, and $\odot$ the Hadamard (element-wise) product. Using a vectoring operator, define $\boldsymbol{y} = \text{Vec}(\mathbf{Y}) \in \mathbb{R}^{N_x N_y}$ and $\boldsymbol{z} = \text{Vec}(\mathbf{Z}) \in \mathbb{R}^{N_x N_y}$. Similarly, define $\boldsymbol{x} \in \mathbb{R}^{N_x \times N_y \times B}$ as

$$\boldsymbol{x} = \text{Vec}(\mathbf{X}) = [\text{Vec}(\mathbf{X}_1)^\top, ..., \text{Vec}(\mathbf{X}_B)^\top]^\top. \tag{2}$$

The measurement process in (1) can thus be expressed as

$$\boldsymbol{y} = [\mathbf{D}_1, ..., \mathbf{D}_B]\boldsymbol{x} + \boldsymbol{z}, \tag{3}$$

where, $\mathbf{D}_b = \text{diag}(\text{Vec}(\mathbf{C}_b)) \in \mathbb{R}^{N \times N}$, for $b = 1, \ldots B$ and $N = N_x N_y$. The sensing matrix $\mathbf{H} = [\mathbf{D}_1, ..., \mathbf{D}_B] \in \mathbb{R}^{N \times NB}$ in video SCI is highly structured and sparse. It has been shown in [13] that, if the signal is structured enough, there exist SCI recovery algorithms with bounded reconstruction error for $B > 1$.

2.2 Deep Learning for Reconstruction and Detection

Reconstruction aims to recover high quality videos from the compressed measurement Y captured by the SCI camera. Significant efforts have been made to develop new reconstruction algorithms in the past decade since high quality videos were recognized as the main output of a SCI camera. Recently, with the aid of DL, this challenge has been addressed using deep convolutional neural networks (CNN) and recurrent neural networks (RNN) [6, 31]. Most recently, motivated by the demanding application of connected and autonomous vehicles, an SCI-vehicle-edge-cloud framework has been proposed [24]. This leads us to think deeper about the main objective of an SCI camera. In addition to the high quality videos, which is of course very important for the subsequent processing, we also need *fast detection and real-time control*, from the raw measurements if possible. Studies in [24] have proved that this dual objective is feasible and thus demonstrated the promising applications of SCI.

E2E-CNN Model Background and Descriptions. Qiao *et al.* proposed the E2E-CNN, a well-trained end-to-end convolutional neural network, in their recent work [31]. This model has led to significant improvements over existing algorithms, providing millisecond-level reconstruction for CI problems. Unlike conventional iteration-based algorithms such as those proposed by Liu *et al.* [20], which require iteration and computation for each measurement, the E2E-CNN optimizes only during the training phase and efficiently recovers images during the inference phase. The E2E-CNN can provide video-rate high-quality reconstruction. In this study, we adopt the E2E-CNN algorithm as our reconstruction model to verify its applicability for vehicle detection on the measurements. We trained the E2E-CNN model using four selected video datasets.

To elaborate, the E2E-CNN model takes measurements and masks as input and produces a reconstructed video as output. As shown in Figure 2(a), E2E-CNN involves a convolutional encoder and decoder with Res-block connection [9]. The encoder and decoder parts each contain five residual blocks, which are connected by two convolutional layers. The first convolutional layer performs multi-dimensional feature extraction from the input data. After each convolution operation, ReLU activation and batch normalization are applied, as illustrated in Figure 2(b). Furthermore, the

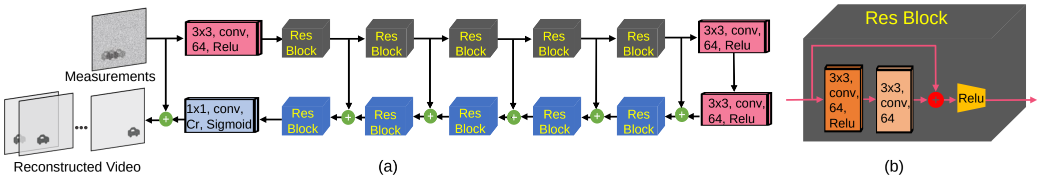


Fig. 2. E2E-CNN architecture [23].

output of an encoder residual block is added to the input of the corresponding decoder residual block, denoted by ⊕. Furthermore, the E2E-CNN model synthesizes the network input into the final reconstruction by utilizing a large-span residual connection. Upsampling and pooling are not employed in the network to prevent loss of image details, and sigmoid is employed as the activation function to ensure the final output has the desired scale.

Taking one step further, it is not optimal to use a fixed compression ratio (B:1) in SCI cameras due to the dynamic nature of the scene. This dictates the research on adaptive sensing, and in this paper, we fill this gap by RL since an SCI camera itself is an agent to sense (and thus capture) the environment.

2.3 RL for Adaptive Sensing

In RL, the goal of the agent is formalized with respect to a specific signal passing from the environment to the agent. This signal is referred to as the reward (r), which is a simple number at each time step (t), *i.e.*, $r_t \in \mathbb{R}$. To be specific, the goal of this work is to maximize the cumulative reward that the agent (SCI camera) receives.

2.3.1 States and Transition Graph. To make the SCI camera learn to automatically determine the optimal B, we have provided a reward at each time step corresponding to the SCI camera's forward action a including increasing B, keeping the current value of B, or decreasing B. More specifically, in this work, we assume that six reconstruction models (E2E-CNN) with different values of B, *i.e.*, $B = \{6, 8, 10, 12, 15, 20\}$ have been trained for real-world applications, comprising a `state set` $\mathcal{S} = \{6, 8, 10, 12, 15, 20\}$. These values are heuristically selected by extensive experiments on various videos to be able to obtain decent reconstructions.

At each state, the SCI camera can decide whether to i) actively increase B, ii) keep the current value of B, or iii) decrease B. Note that "increase" and "decrease" can skip intermediate values of B; for example, our policy allows changing $B = 15$ to $B = 6$ as in real life applications, when a red traffic light or an accident can suddenly halt all cars (a large B can be used) while all cars will speed up (a small B is required) when the traffic light turns green. We use a to represent the `action set` and $a = \{decrease, keep, increase\}$, which is predicted by the RL model. S' indicates the updated state after conducting each a. As to each action step, RL provides the corresponding `reward` $r\,(S, a, S')$, which is related to the corresponding environment.

Table 1 summarizes the dynamics of the transition table for a simple example. For the sake of conciseness and concreteness, Table 1 only considers three states, *i.e.*, $\mathcal{S} = \{6, 10, 15\}$. In this example, a period of search that begins with $S = 6$ cannot leave for the new state S' with $a = decrease$ since 6 is already the minimum value of B; therefore, the corresponding conditional probability $p\,(S' \mid S, a) = 0$ and no related reward r exists. However, with the action of increase, *i.e.*, $S = 6$ and $a = increase$, S could be increased to 10 or 15 (*i.e.*, $S' = 10$ or 15) with probability α and $1 - \alpha$, respectively, where $\alpha \in [0, 1]$. Similarly, a period of searching undertaken when $S = 15$ and $a = decrease$ ends at $S' = 6$ with probability β and $S' = 10$ with probability $1 - \beta$, with $\beta \in [0, 1]$. The corresponding state transition graph is shown in Fig. 3.

Table 1. State transition table of the proposed RL for adaptive video CS by only considering three states, *i.e.*, $S = 6, 10, 15$. *increase*^{2} denotes the increase of B skips from 6 to 15 and similarly *decrease*^{2} denotes the decrease from 15 to 6 with $\alpha, \beta \in [0, 1]$.

S	a	S'	$p(S' \mid S, a)$	$r(S, a, S')$
6	decrease	6	0	–
6	keep	6	1	r_{keep}
6	increase	10	α	$r_{increase}$
6	*increase*^{2}	15	$1 - \alpha$	$r_{increase}$
10	decrease	6	1	$r_{decrease}$
10	keep	10	1	r_{keep}
10	increase	15	1	$r_{increase}$
15	*decrease*^{2}	6	β	$r_{decrease}$
15	decrease	10	$1 - \beta$	$r_{decrease}$
15	keep	15	1	r_{keep}
15	increase	15	0	–

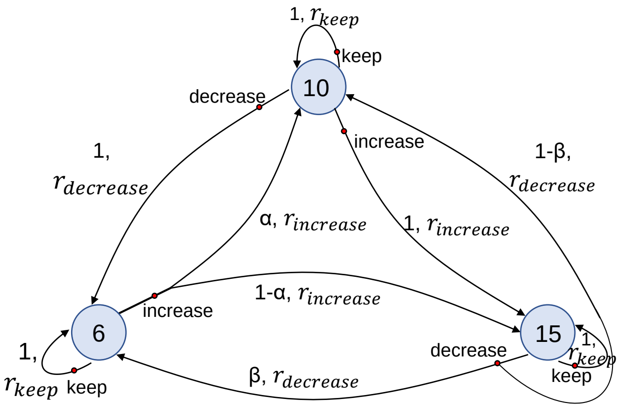


Fig. 3. State transition graph of Table 1.

2.3.2 *Reward Policy.* In real-world applications, the reward policy design of RL is highly correlated with the involved deep learning models and the specific scenes. As shown in Fig. 1, the SCI captured measurements are sent to two modules *i.e.*, the *detection* module and *reconstruction* module, to perform object detection and optionally the video reconstruction, respectively. Therefore, we consider the detection rate and PSNR of the reconstructed video as the key performance metrics for the RL module to adjust B for different scenes.

Here, PSNR [30] refers to the peak-signal-to-noise ratio between two images, and we use it to evaluate the performance of the reconstruction model (E2E-CNN) [31]. More specifically, let $\mathbf{X}^* \in \mathbb{R}^{N_x \times N_y \times B \times G}$ denote the ground truth video group, where G denotes the number of measurements being used, and $\hat{\mathbf{X}}$ be the reconstructed video by the E2E-CNN with the same size as $\mathbf{X}^*$. The average PSNR of the video group is given by:

$$\mathrm{PSNR} = \frac{1}{BG}\left[-10\log\frac{\sum_{n_x=1}^{N_x}\sum_{n_y=1}^{N_y}(\hat{x}_{n_x,n_y,b,g} - x^*_{n_x,n_y,b,g})^2}{N_xN_y}\right] \tag{4}$$

Algorithm 1 RL for Adaptive video CS

```
Require: H, detection model (and reconstruction models).
Initial $B$, $drth$ as the threshold of acceptable detection rate, and optionally $psnrth$ as the threshold of acceptable PSNR.
while Capturing do
    Capture measurement of Y.
    Perform detection on the measurement and output the detection rate. Optionally conduct the reconstruction and calculate PSNR during training.
    RL policy update by detection rate (and PSNR).

    if $detect\_rate<drth$ then
        if $a$=decrease OR ($a$=keep AND $B$=$B_{min}$) then
            $r \leftarrow r_1$
        else
            $r \leftarrow r_2$
        end if
    else
        if $a$=increase OR ($a$=keep AND $B$=$B_{max}$) then
            $r \leftarrow r_1$
        else
            $r \leftarrow r_2$
        end if
    end if

    if PSNR provided then
        if PSNR > $psnrth$ then
            if $r > 0$ then
                $r \leftarrow r * \lambda_1$
            else
                $r \leftarrow r * \lambda_2$
            end if
        else
            if $r > 0$ then
                $r \leftarrow r * \lambda_2$
            else
                $r \leftarrow r * \lambda_1$
            end if
        end if
    end if

    Output $B$, $r$.
end while
```

where $\hat{x}_{n_x,n_y,b,g}$ and $x^*_{n_x,n_y,b,g}$ denote the (n_x, n_y)-th pixel in the b-th frame of the g-th measurement in the estimated video and ground truth video, respectively. Usually, the lower the value of B, the higher the PSNR (smaller error), and the better the quality of the reconstructed image.

In this work, the goal of video CS is to conduct object detection on the *measurements* (compressed data captured by SCI cameras) with an adaptive compression ratio (B). Therefore, apart from PSNR, the detection rate is a good objective metric for this task to assist the adjustment of B, *i.e.*, it is also sent to the RL module to adjust B for different scenes. Other metrics can also be used in the future for the same or different tasks.

Algorithm 1 presents the RL reward mechanism for adaptive temporal video CS. As depicted in it, after defining the sensing matrix $\mathbf{H}$ and the initial B, the RL module will predict the action (*i.e.*, increase the value of B, keep the current value, or decrease it) based on the captured measurement of $\mathbf{Y}$, and update B accordingly. We will then perform object detection through YOLOv3-Tiny on the measurements and calculate the detection rate. Here, YOLOv3-Tiny [11] is a light-weight DL algorithm designed for resource-constrained devices, with superior advantages on fast object detection due to the significantly reduced parameters. Optionally, the measurements can also be sent to the reconstruction module for video recovery, and the PSNR of the reconstructed video (available during training) will be sent to the RL module to adjust B for different scenes.

Lines 6-11: The RL module first defines the thresholds (lower bounds) of the acceptable detection rate and PSNR as $drth$ and $psnrth$, respectively. The higher the values of $drth$ and $psnrth$ the smaller the value of B. Consider a round of capturing as an example; if the calculated detection rate is smaller than the threshold, *i.e.*, $detect\_rate < drth$, it reveals that the current B is larger than the optimal value, so we expect the RL module to output a smaller B. In this context, if *i*) the corresponding action a indicates to decrease B, or *ii*) the action a is to keep the current B when B already achieves its minimum value, then the RL module will assign a positive reward r_1 as encouragement; otherwise, it will assign a negative reward r_2 as penalty.

Lines 12-18: Similarly, if $detect\_rate > drth$, it reveals that the current B is smaller than the optimal value, so we expect the RL module to output a larger B. In this context, if *i*) the corresponding action a indicates to increase B, or *ii*) the action a is to keep the current B and B already achieves its maximum value, then the RL module will assign a positive reward r_1 as encouragement; otherwise, it will assign a negative reward r_2 as penalty.

Lines 19-33: Optionally, if reconstruction is conducted and the corresponding PSNR is provided, the reward mechanism will take it into account: *i*) when PSNR > $psnrth$ (*i.e.*, revealing that the RL module should increase B), if the current cumulative reward r is positive, the RL module will update the reward by $r \cdot \lambda_1$ ($\lambda_1 \in (1, 2)$, we let $\lambda_1 = 1.1$) to increase the related reward; otherwise, the reward will be updated by $r \cdot \lambda_2$ ($\lambda_2 \in (0, 1)$, we let $\lambda_2 = 0.8$) to weaken the reward. *ii*) When PSNR < $psnrth$ (*i.e.*, revealing that the RL module should decrease B), if the current cumulative reward r is positive, the RL module will update the reward by $r \cdot \lambda_2$ to weaken the related reward; otherwise, the reward will be updated by $r \cdot \lambda_1$ to increase the reward. Finally, Algorithm 1 will output B and the cumulative reward r.

Specifically, in our experiments, during training when PSNR is available, we consider three scenarios: *i*) PSNR<24, *ii*) 24 $\leqslant$ PSNR$\leqslant$28, and *iii*) PSNR > 28. The range 24 $\leqslant$PSNR$\leqslant$28 indicates a good performance of the reconstruction model. Since we expect to obtain a relatively higher B, we set the corresponding reward to $r = |\text{PSNR} - 24| \cdot B$; this way, a higher B will provide a higher reward, encouraging the agent to figure out a higher B while guaranteeing the reconstruction quality. When PSNR<24, which denotes a poor quality reconstruction, we should reduce B; therefore, the reward r is negative as a punishment. Similarly, if PSNR>28 in the current time step, we could further improve the value of B, so the reward r is positive to encourage a higher B. Although the specific positive and negative rewards depend on the specific scene, the basic idea is the same.

2.3.3 RL Agent. We adopt model-based reinforcement learning method, and build an LSTM model as an agent to make inferences with historical information. As for the input and output of the LSTM model, the input is the most latest100 decision values and the corresponding reward value, as well as the array composed of the detection rate and PSNR, and the output is the predicted action.

3 EVALUATION RESULTS

3.1 Datasets and Experiment Setting

We choose four case studies to show how the proposed RL module can automatically adjust B for different scenes, including urban, highway, grocery store, and NBA scenes. For each case study, we select a specific dataset to train and test the RL module.

Urban Dataset: We selected the public dataset of traffic video (PDTV) [35] which provides traffic videos at three intersections with annotations for real transportation applications, such as tracking road users and detection of pedestrian infractions. The video dataset was collected at three sites of Belarus and Canada with a resolution of 640 × 480 pixels at 30 frames per second (fps), and the traffic scenes cross diverse traffic, lighting, and weather conditions.

Highway Dataset: The DynTex dataset [29] is the first collection of high-quality dynamic texture videos that are structured by videos' underlying physical processes such as waving motion and discrete units, with the goal of serving as a standard database for dynamic texture research. Nine videos related to traffic, with a resolution of 720 × 576 pixels at 30 fps were selected.

Grocery Store Video Dataset: These videos are collected from retail surveillance cameras at a middle-sized grocery store. The camera captures top-down views monitoring both the incoming and outgoing customer flow at the entry gate. Eight video clips with a resolution of 1920 × 1080 pixels at 30 fps were selected.

NBA Dataset: This is a publicly available NBA dataset to test our proposed framework on high-speed sport motions. In the video, two groups of basketball players are moving fast, which is significantly different from other scenes. We selected 7 video clips with a resolution of 640 × 480 pixels at 30 fps for the experiments.

3.2 Training Details

E2E-CNN Training and Validation. We have six compressed versions of the same video sets to train the E2E-CNN reconstruction modules, *i.e.*, using $B = 6, 8, 10, 12, 15, 20$ and the network structure proposed in [31][2]. We combine the compressed video segments from the selected video datasets for training and testing. We randomly select 80% of the measurements for training and the rest for validation. Since not all of these public datasets provide annotations, we directly employ the open YOLOv3 network[3] on the original public video dataset to obtain labels (bounding boxes of targets) and treat these labels as the ground truth.

Following [6], we define the *normalized measurement* from the forward model of SCI in (1) as

$$\bar{\mathbf{Y}} = \mathbf{Y}./\textstyle\sum_{b=1}^{B}\mathbf{C}_b. \tag{5}$$

This normalized measurement removes the mask artifacts especially in the background and we use it to show the speed of the scene when presenting the results.

RL Training. The RL algorithm seeks to maximize a certain measure of the agent's cumulative reward, as the agent interacts with the environment. In this work, we use the OpenAI Gym framework [3] to build the RL environment. OpenAI Gym focuses on the episodic setting of RL, where the agent's experience is divided into a series of episodes. For each episode, the starting state of the agent is randomly sampled from a distribution, and the interaction proceeds until it reaches a terminal state under the specific environment. For each use case, we selected the related types of video clips to train the RL model on an NVIDIA GPU workstation (4×GeForce RTX 2080 Ti graphics cards), with the goal of maximizing the expectation of total reward per episode, and to achieve a high level of performance in as few episodes as possible. We retrained the object detection model (YOLOv3-Tiny) on the SCI measurements, along with the RL model.

[2] Code from: https://github.com/mq0829/DL-CACTI.

[3] The YOLO series algorithms were firstly proposed in [33], and are well known for fast detection speed by simple and clear algorithm structure. One popular algorithm, YOLOv3 [34], automatically selects the suitable initial regression frame by incorporating the K-means clustering approach for a specific input dataset.

3.3 Adaptive Sensing Results Based on PSNR

To prove the concept, we first only consider the reconstruction module with PSNR available but without using the detection rate, aiming to verify the RL module. The adaptive B results as well as the PSNR are shown in Fig. 4 for the Urban and Highway data, and in Fig. 5 for the Grocery-store and NBA data. Note that in the Urban and Highway data, we freeze the videos (in the middle part) and speed them up by skipping frames (last part) to simulate different velocities of the vehicles.

It can be seen from Fig. 4 that starting from a random B, when the video is frozen, RL will adjust B to a larger value such as 15 and when the video is speeding up in the last hundreds of frames, B is adjusted to a small value such as 6 or 8. Differently from these simulated videos, persons in the grocery store and players in the NBA data change speed by themselves, which are real videos that SCI cameras may be deployed for. Again, as shown in Fig. 5, starting from a random B, when the persons or players move fast, our RL module will infer a smaller B and when nobody moves, a large B such as 20 is inferred. When people start to move, B drops again. These four videos clearly verify that our RL works well with respect to reconstruction quality and PSNR.

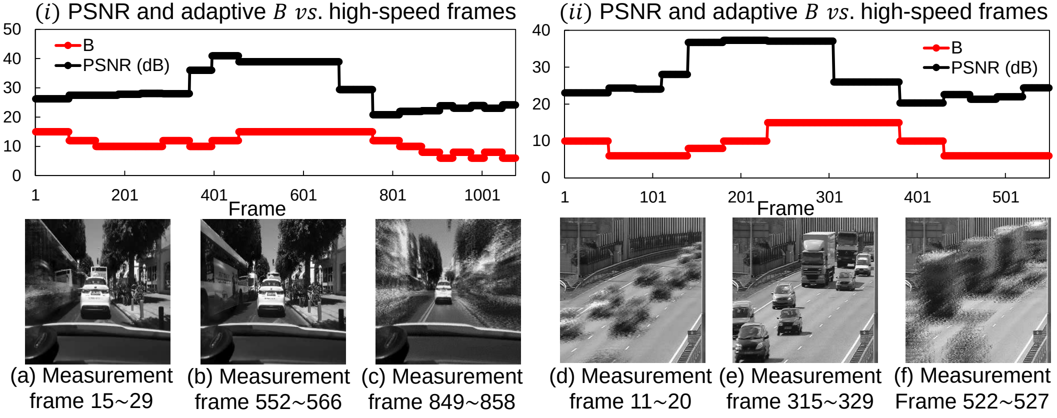


Fig. 4. (*i*-*ii*) Reconstruction PSNR (dB) and adaptive B estimated from the reconstructed Urban (left) and Highway (right) video based on PSNR only, plotted against frame number. (a-f) Normalized measurements with vehicles at different velocities.

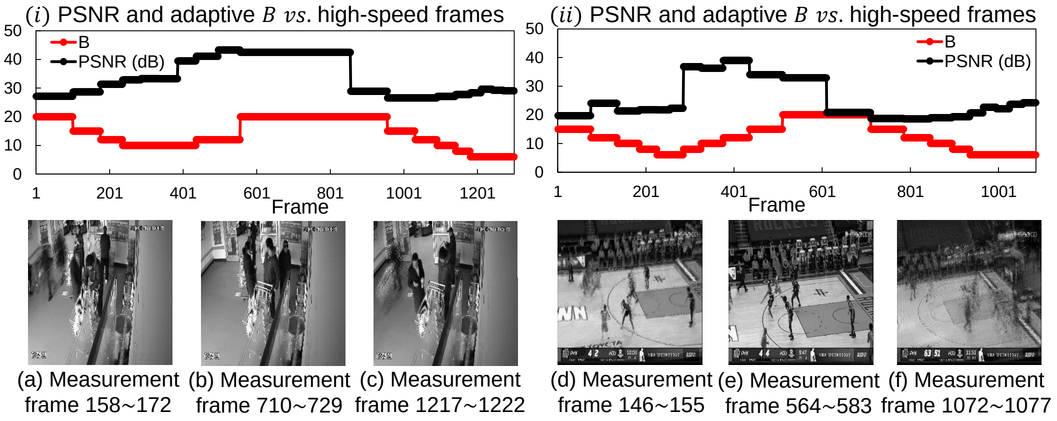


Fig. 5. (*i*-*ii*) Reconstruction PSNR (dB) and adaptive B estimated from the reconstructed Grocery-store video (left) and NBA video (right), all are plotted against frame number. (a-f) Normalized measurements with vehicles at different velocities.

Next, we show results based on the detection rate, as the PSNR is usually not available in real cases.

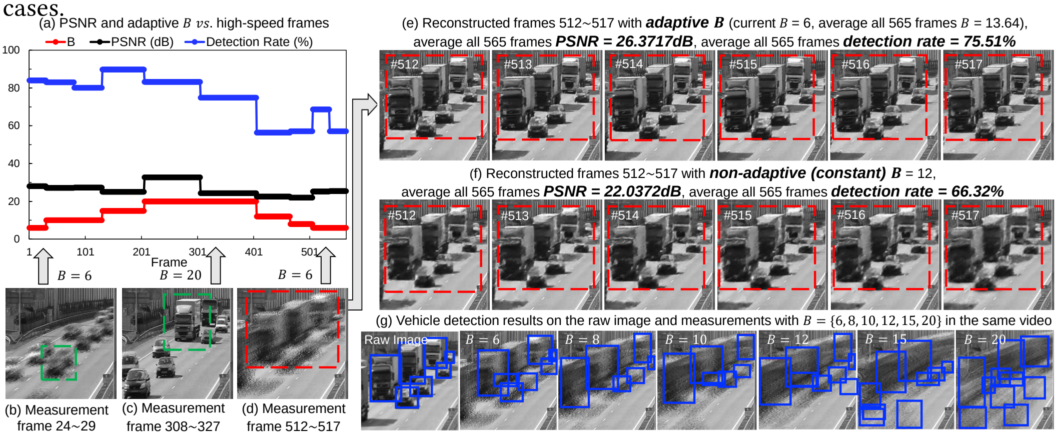


Fig. 6. Adaptive B from the detection rate on the measurements directly. (a) Reconstruction PSNR (dB) and adaptive B (frames) (average adaptive B=13.64) from the *measurements*, all are plotted against frame number. (b-d) Normalized measurements when there is no truck, two trucks, and four trucks moving inside the scene, adapted $B = 6, 20, 6$, respectively. (e) Reconstructed frames 512∼517 from the measurement in (d) with *adaptive* B. (f) Reconstructed frames 512∼517 with non-adaptive (constant) $B = 12$. (g) Vehicle detection results on the raw images and measurements with different $B = \{6, 8, 10, 12, 15, 20\}$ in the same video clip. Videos in the SM.

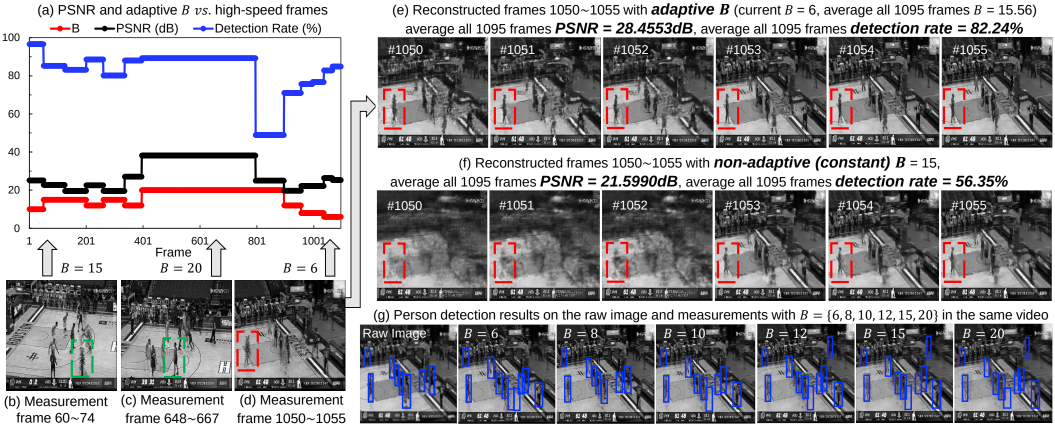


Fig. 7. Adaptive B from the detection rate on the measurements directly. (a) Reconstruction PSNR (dB) and adaptive B (frames) (average B=15.56) from the *measurements*, against frame number. (b-d) Normalized measurements when the basketball players are running from the left-hand scene to stop at the right-hand scene, adapted $B = 15, 20, 6$, respectively. (e) Reconstructed frames 1050∼1055 from the measurement in (d) with *adaptive* B. (f) Reconstructed frames 1050∼1055 with non-adaptive (constant) B=15. (g) Person detection results on the raw images and measurements with different $B = \{6, 8, 10, 12, 15, 20\}$ in the same video clip. Videos in the SM.

3.4 RL based on Detection Rate

In real life applications, the detection rates are sent to the RL module to adjust B. As mentioned before, we employ YOLOv3 [34] on the original video dataset to obtain labels (bounding boxes of targets) and treat these labels as the ground truth. Then, we employ YOLOv3-Tiny [11], a lightweight DL algorithm designed for resource-constrained devices, *on the measurements* to detect vehicles and person for the sake of speed. The detection can also be performed on the *reconstructed videos*, which can potentially increase the accuracy by trading off power and latency [24]. In this work, aiming to conduct adaptive video CS on the end-user cases with limited power but requiring instant responses such as in self-driving vehicles, we use the *detection on measurements directly.*

In terms of detection metrics, a common way is to compute the intersection-over-union (IoU) between ground truth and prediction. IOU is a measure of the degree of overlap between two detected frames for target detection:

$$\mathrm{IOU} = \frac{\text{area}\left(BBOX_p \cap BBOX_{gt}\right)}{\text{area}\left(BBOX_p \cup BBOX_{gt}\right)}, \tag{6}$$

where $BBOX_{gt}$ represents the bounding box of the ground truth (GT), and $BBOX_p$ of the predicted frame. Predictions whose IoUs are larger than 0.5 are considered as true positives (TP). We use mAP (mean Average Precision) as our detection rate score:

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{TP}{\text{all detections}}, \tag{7}$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{TP}{\text{all ground truths}}, \tag{8}$$

where TP is the number of detection frames with IoU > 0.5 and FP with IoU $\leqslant 0.5$ detection frames, or the number of redundant detection frames detecting the same GT. FN refers to the number of missing detections.

In our four datasets, we only detect `vehicles` in the highway and urban scenarios, and in the other two scenarios, we only detect `persons`.

During implementation, we calculate the mAP for each batch size corresponding to $Q = BatchSize \times B$ video frames (for the *BatchSize* measurements). The reason for this is that the calculated `DetectionRate` (mAP) will not fluctuate sharply, but will change with the scene within a certain range. This is also the adaptation time of our RL module and the *BatchSize* can be set to one for fast adaptation in real applications.

For the reward design, we set the threshold (lower bound) of the acceptable detection rate as 75%, *i.e.*, $drth$ = 75%, and obey the reward mechanism in Algorithm 1 for adaptive video CS. The detection rate here is calculated by taking the IOU of all the bounding boxes predicted by YOLO-Tiny on the vehicle/edge and YOLOv3 on EdgeServer, and the bounding boxes in the measurements and reconstructed frames cannot be completely overlapped. Since the bounding boxes in reconstructed frames do not completely coincide with the bounding boxes in measurements, we found in our experiments that 75% is a reasonable value, and the normal training and application of the reconstructed frame can be achieved in selected videos under diffenent scenarios. We also show the PSNR of the reconstructed videos for comparison purposes.

We believe that it is the right approach to compare our proposed method against a fixed compression ratio (B:1). For adaptive sensing of video CS considered here, the only paper related to ours is [51], which considers the same problem by using a motion estimation method to adapt B. However, both the reconstruction algorithm and the adaptive sensing framework developed therein produce low-quality results. Specifically, it has been shown in [6, 31] that the E2E-CNN used in this paper can provide much better results than the reconstruction algorithms used therein. Besides, the look-up table used therein is not flexible. Our main goal of this paper is to prove that RL works well in adaptive video compressed sensing.

Highway Scene: Figure 6 presents the testing results based on the traffic video in the highway with the goal of detecting vehicles from the raw *adaptive measurements*. Specifically, Fig. 6 (a) presents the changes in PSNR (dB), detection rate (%) and *adaptive B* (frames) from the measurements against a constant stream of traffic video frames. Starting from a random B, RL module adjusts B based on the learned speed and content from the raw measurements. Similarly to Fig. 4, we keep the original video speed of the first one-third of the video frames, then freeze the video for the middle, and finally skip every two frames to simulate a fast speed scenario for the last two one-third of video frames. Under the decision of our proposed RL, B has approximately maintained a certain range at the beginning, then rises to a higher level (B = 20 in the frozen frames), and then drops back to a lower level after a period of time (due to the high speed). Once a certain B is decided, the calculated Detection Rate and PSNR will lead to the opposite change of B, *i.e.*, an increased B will lead to a decrease in the detection rate and PSNR, and vice versa. Consequently, three normalized measurements with different values of adaptive B are shown in Fig. 6 (b-d) with adaptive B = 6, 20, 6. We can see that the normalized measurement (c) has the largest adaptive B = 20 since its corresponding original video frames are stationary, while the normalized measurement (d) is blurry with the smallest adaptive B = 6 due to the fast object speed in these video frames.

This video has a total of 565 frames, achieving a mean compression ratio (average B) of 13.64. To demonstrate the usability of adapting B based on the sensed video data, we compare adaptive reconstructions (Fig. 6(e)) to those obtained when B is fixed at or near its expected value (Fig. 6(f) at B=12). Fig. 6(f) shows the reconstructed frames 512∼517 from the measurement in (d) with *non-adaptive* (constant) B. Comparing Figs. 6(e) and 6(f), we notice that adapting B provides a significant (4.3dB) higher reconstruction quality (average all 565 frames PSNR=26.37dB) than fixing B even lower than its expected value (average PSNR=22.04dB). Besides, it also improves the average detection rate from 66.32% to 75.51%. To present the effects of diverse B on the object detection based on measurements, we visualize the vehicle detection results on the raw (original) images and measurements with different B = {6, 8, 10, 12, 15, 20} in the same video clip in Fig. 6(g). It can be seen that a decent detection rate is obtained at B = 6 or 8, while a larger B will lead to false alarms.

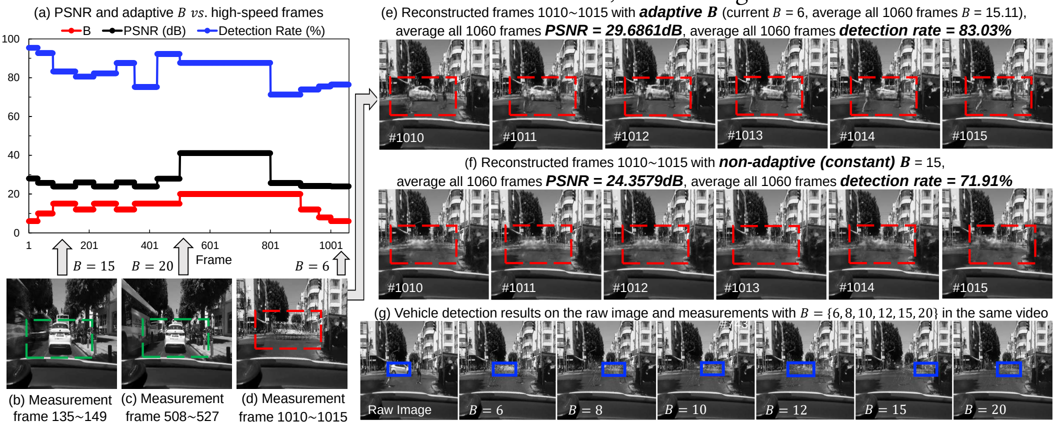
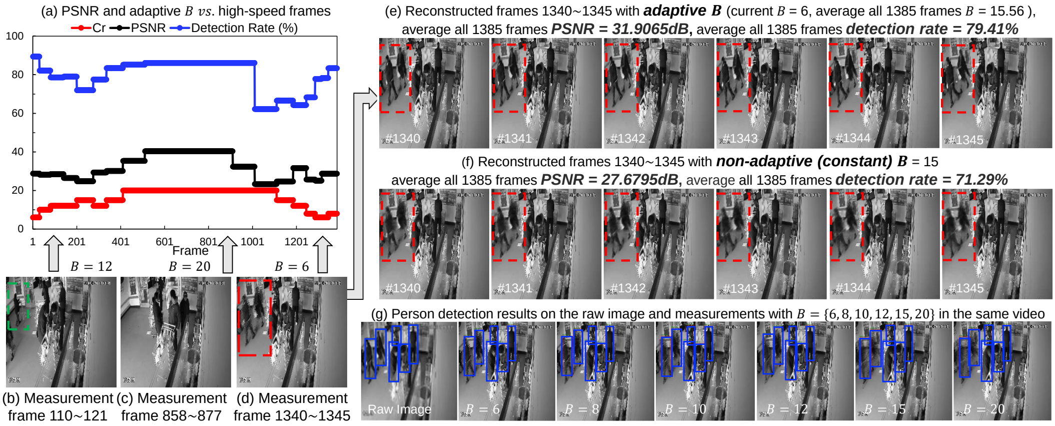


Fig. 8. Adaptive B from based on the detection rate from the measurements directly. (a) Reconstruction PSNR (dB) and adaptive B (frames) (average adaptive B = 15.11) from the *measurements*, all are plotted against frame number. (b-d) Measurements when there is one moving front vehicle, one stopping front vehicle, and one front vehicle *passing vertically and suddenly* inside the scene, adapted B = 15, 20, 6, respectively. (e) Reconstructed frames 1010∼1015 from the measurement in (d with *adaptive B*. (f) Reconstructed frames 1010∼1015 with non-adaptive (constant) B = 15. (g) Vehicle detection results on the raw images and measurements with different B = {6, 8, 10, 12, 15, 20} in the same video clip.

NBA Scene: Following similar steps, Fig. 7 presents the testing results for the publicly available NBA video. Unlike previous vehicle-related scenes, NBA videos are used to detect basketball players. Although the speed of human movement may be not as fast as that of vehicles, the corresponding inference of human-related video frames may not necessarily have better results. Because a single target (here is the person) occupies fewer pixels compared to vehicles, especially the rapid movement of players and mutual occlusion will make the measurements more blurry as in Fig. 7(b)-(d). As shown in Fig. 7(a), in the latter part, the detection rate has a relatively sharp drop, caused by the dramatic transition from slow to very rapid changes in adjacent frames of the video clip. From the selected reconstructed frames in Fig. 7(e)-(f) and detection frames in (g), we can see that adapting B leads to a 6.85 dB improvement in PSNR and a 25.89% increase in detection rate. This clearly verified the efficacy of our proposed RL for adaptive sensing in saving memory and bandwidth (an average higher B), power (detection on the raw measurements directly) and potential cost.

Urban Scene: Figure 8 shows the testing result of an urban video clip taken by the front camera of a driving connected vehicle, with the goal of detecting surrounding vehicles from the raw *adaptive measurements*. Differently from the highway video, the captured surrounding vehicles have smaller relative speed compared with the camera (host vehicle) at the beginning, as the host and surrounding vehicles are driving along the same road. Then the traffic light at the intersection turns from green to red, and the relative speed differences between the host and surrounding vehicles become smaller and smaller until all vehicles become stationary. In the latter part of this video, the traffic lights become green again and all vehicles speed up aiming to cross the intersection. Here, we can notice some front vehicles passing perpendicularly with respect to the image plane with higher speed *suddenly*, which *simulates the driving situation where pedestrians or vehicles suddenly cross the road and the host vehicle needs a quick emergency response by analyzing captured measurements to avoid collisions and fatal crashes.*

Specifically, Fig. 8 (a) presents the changes in reconstruction PSNR (dB), detection rate (%) and the related *adaptive B* (frames) from the measurements against a constant stream of traffic video



(b) Measurement frame 110~121 (c) Measurement frame 858~877 (d) Measurement frame 1340~1345

(e) Reconstructed frames 1340~1345 with ***adaptive B*** (current B = 6, average all 1385 frames B = 15.56), average all 1385 frames ***PSNR = 31.9065dB*****,** average all 1385 frames ***detection rate = 79.41%***

(f) Reconstructed frames 1340~1345 with ***non-adaptive (constant) B*** = 15 average all 1385 frames ***PSNR = 27.6795dB*****,** average all 1385 frames ***detection rate = 71.29%***

(g) Person detection results on the raw image and measurements with $B = \{6, 8, 10, 12, 15, 20\}$ in the same video

Fig. 9. Adaptive B based on the detection rate from the measurements directly. (a) Reconstruction PSNR (dB) and adaptive B (frames) (average adaptive B = 15.84) from the *measurements*, all are plotted against frame number. (b-d) Measurements when there are one customer entering, no customers entering or leaving, and two customers leaving the grocery store, adapted B = 12, 20, 6, respectively. (e) Reconstructed frames 1340~1345 from the measurement in (d with *adaptive B*. (f) Reconstructed frames 1340~1345 with non-adaptive (constant) B = 10. (g) Person detection results on the raw images and measurements with different $B = \{6, 8, 10, 12, 15, 20\}$ in the same video clip.

frames. Starting from a random B, the RL module adjusts B based on learning the speed and content from the raw measurements. Three measurements with different values of the adaptive B are shown in Fig. 8 (b-d) with adaptive B = 15, 20, 6. We can see that the measurement is clear with the largest adaptive B = 20 since its corresponding original video frames are stationary, while measurement (d) is more blurry with the smallest adaptive B = 6 due to the fast speed of the related video frames and the fast speed of the front vehicle that is passing perpendicularly to the camera. This video takes a total of 1060 frames to capture, achieving a mean compression ratio (average B) of 15.11.

To demonstrate the usability of adapting B based on the sensed video data, we compare adaptive reconstructions (Fig. 8(e)) to those obtained when B is fixed at or near its expected value (Fig. 8(f) at B=15). Fig. 8(f) shows the reconstructed frames 1010~1015 from the measurement in (d) with *non-adaptive* (constant) B. Comparing Fig. 8(e) and Fig. 8(f), we notice that adapting B provides a significant (5.3dB) higher reconstruction quality (average PSNR of all 1060 frames is equal to 29.69dB) than fixing B even lower than its expected value (average PSNR=24.36dB). Besides, it also improves the average detection rate from 71.91% to 83.03%. To present the effects of diverse B on the object detection based on measurements, we visualize the vehicle detection results on the raw images and measurements with different $B = \{6, 8, 10, 12, 15, 20\}$ in the same video clip in Fig. 8(g).

Grocery Store Scene: Following similar steps, Fig. 9 presents the testing results based on the surveillance videos collected from a middle-sized grocery store. As shown in Fig. 9(a), B has approximately maintained a certain range at the beginning, then rises to a higher level (B = 20 in the frozen frames), and then drops back to a lower level after a period of time (due to high speed). Once a certain B is decided, the calculated detection rate and PSNR will lead to the opposite change of B, *i.e.*, an increased B will lead to a decrease in the detection rate and PSNR, and vice versa. From the exemplar reconstruction frames in Fig. 9(e)-(f) and detection frames in (g), we can see that our adaptive B provides a higher (4.2dB) reconstruction quality than fixing B even lower than its expected value, and it also improves the average detection rate from 71.29% to 79.41%.

3.5 Performance of the Reconstruction

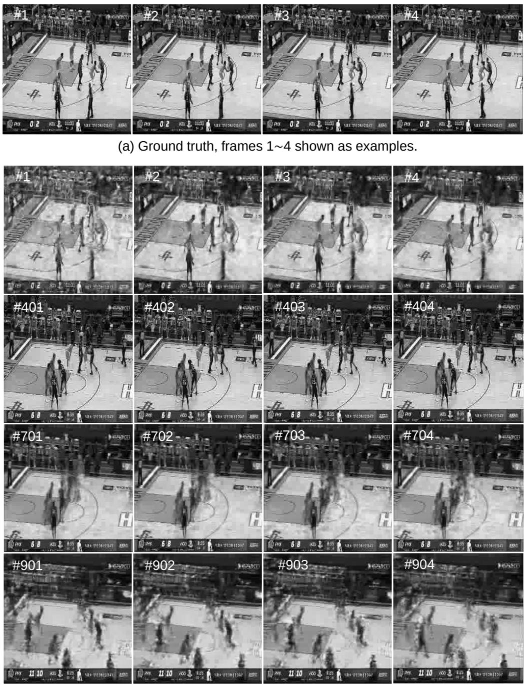
Person Related Videos: Figure 10 presents an adaptive B on the NBA video. Specifically, Fig. 10(a) shows the ground truth of the first four frames as examples. Several reconstructed frames based on the adaptive B are shown in Fig. 10(b). In comparison, the reconstructed images of the NBA video are more blurry than those in the grocery store video since the movement speed of players is much higher than the speed of customers.

Vehicle Related Videos: Similarly, Fig. 11 and Fig. 12 implement adaptive B on the urban video and the highway video captured by the front camera of a driving vehicle and the traffic camera, respectively. Fig. 11(a) and Fig. 12(a) also present the ground truth of the first four frames as examples. Selected reconstructed frames based on the adaptive B are presented in Fig. 11(b) and Fig. 12(b).

It can be seen from these plots that by using our proposed adaptive video sensing approach, the reconstructed frames are consistently at a high quality level.

3.6 Additional Considerations

Recovery from Noisy Measurements: We also verified the proposed RL module's robustness to noise by investigating the recovery from noisy measurements. Specifically, as shown in Table 2, when zero-mean Gaussian noise $\boldsymbol{n} \sim \mathcal{N}(0, \sigma)$ is added to the measurements (normalized to $[0, 1]$), both the quality of the reconstruction (as measured by PSNR in dB), as well as the detection rates (DR, 1 is the highest value) are high for different noise levels.



(a) Ground truth, frames 1~4 shown as examples.

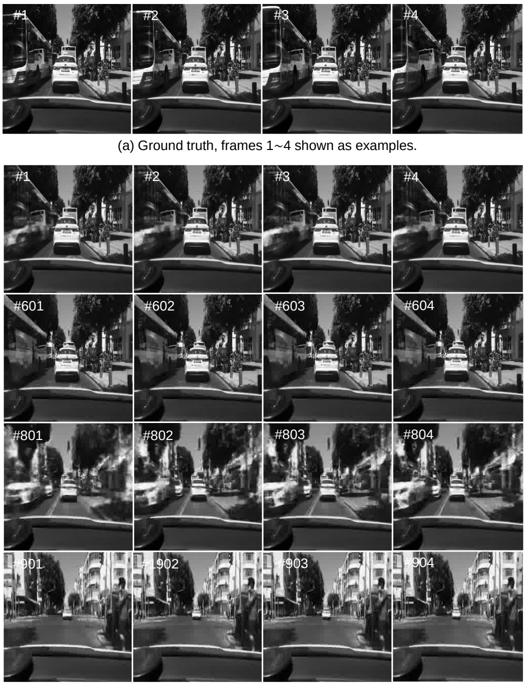
(b) Reconstructed video frames, 16 selected frames shown as examples.

Fig. 10. Selected reconstructed frames (b) based on the adaptive B presented in the NBA scene. Frames 1 to 4 in (a) are shown as examples of ground truth.

Table 2. Reconstruction PSNR and detection rate *vs.* noise σ.

PSNR, DR B σ	**6**	**10**	**15**
0	28.73, 0.8543	28.44, 0.8557	28.33, 0.8138
0.005	28.56, 0.8521	28.30, 0.8436	28.19, 0.8018
0.010	28.18, 0.8374	27.99, 0.8162	27.89, 0.7745
0.050	24.70, 0.7534	24.62, 0.7633	24.52, 0.7126
0.100	21.58, 0.7147	21.52, 0.7123	21.44, 0.6849

Inference Speed: In addition, the inference speed of our RL module is high for many time-sensitive applications. For example, in terms of autonomous driving, when a connected and autonomous vehicle (CAV) is driving in an urban area at a speed of 40 kilometers per hour, the execution time of each real-time task should be less than 100 milliseconds [22]. On average, our whole RL module for inference takes 12 milliseconds per measurement. The inference time of object detection models, *i.e.*, YOLOv3 and YOLOv3-Tiny, are 42 milliseconds and 16 milliseconds, respectively. Regarding the E2E-CNN (not necessary), the inference time is 29 milliseconds. The total of all those inference speeds is much less than 100 milliseconds, which shows actionable insights of employing our work for real-world CAV applications.



(a) Ground truth, frames 1~4 shown as examples.

(b) Reconstructed video frames, 16 selected frames shown as examples.

Fig. 11. Selected reconstructed frames (b) based on the adaptive B presented in the urban scene. Frames 1 to 4 in (a) are shown as examples of ground truth.

4 CONCLUSIONS

We introduce reinforcement learning to perform adaptive temporal compressive sensing of video. The proposed RL algorithm conducts adaptive sensing directly on the raw measurements and thus saves memory, bandwidth and power on the end-users equipped with SCI cameras. Extensive results demonstrated the potential of our proposed methods in real life applications of video compressive sensing. We are working on building an end-to-end system of video SCI and RL to conduct real-time adaptive sensing experiments and demonstrations using our proposed algorithm.

REFERENCES

[1] Rajarshi Bhattacharyya, Archana Bura, Desik Rengarajan, Mason Rumuly, Srinivas Shakkottai, Dileep Kalathil, Ricky KP Mok, and Amogh Dhamdhere. 2019. QFlow: A reinforcement learning approach to high QoE video streaming over wireless networks. In *Proceedings of the twentieth ACM international symposium on mobile ad hoc networking and computing*. 251–260.

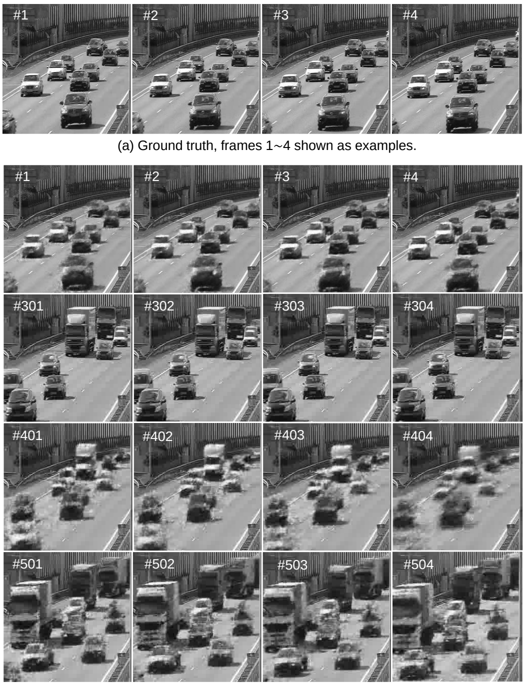
[2] J.M. Bioucas-Dias and M.A.T. Figueiredo. 2007. A New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration. *IEEE Transactions on Image Processing* 16, 12 (December 2007), 2992–3004.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] E. J. Candes, J. Romberg, and T. Tao. 2006. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory* 52, 2 (Feb 2006), 489–509.

[5] Z. Cheng, B. Chen, G. Liu, H. Zhang, R. Lu, Z. Wang, and X. Yuan. 2021. Memory-Efficient Network for Large-scale Video Compressive Sensing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[6] Ziheng Cheng, Ruiying Lu, Zhengjue Wang, Hao Zhang, Bo Chen, Ziyi Meng, and Xin Yuan. 2020. BIRNAT: Bidirectional Recurrent Neural Networks with Adversarial Training for Video Snapshot Compressive Imaging. In *European Conference on Computer Vision (ECCV).*



(a) Ground truth, frames 1~4 shown as examples.

(b) Reconstructed video frames, 16 selected frames shown as examples.

Fig. 12. Selected reconstructed frames (b) based on the adaptive B presented in the highway. Frames 1 to 4 in (a) are shown as examples of ground truth.

[7] D. L. Donoho. 2006. Compressed sensing. *IEEE Transactions on Information Theory* 52, 4 (April 2006), 1289–1306. https://doi.org/10.1109/TIT.2006.871582

[8] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk. 2008. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine* 25, 2 (2008), 83–91.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. K. Nayar. 2011. Video from a single coded exposure photograph using a learned over-complete dictionary. In *2011 International Conference on Computer Vision*. 287–294. https://doi.org/10.1109/ICCV.2011.6126254

[11] Rachel Huang, Jonathan Pedoeem, and Cuixian Chen. 2018. YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2503–2510.

[12] Michael Iliadis, Leonidas Spinoulas, and Aggelos K. Katsaggelos. 2020. DeepBinaryMask: Learning a binary mask for video compressive sensing. *Digital Signal Processing* 96 (2020), 102591. https://doi.org/10.1016/j.dsp.2019.102591

[13] S. Jalali and X. Yuan. 2019. Snapshot Compressed Sensing: Performance Bounds and Algorithms. *IEEE Transactions on Information Theory* 65, 12 (Dec 2019), 8005–8024. https://doi.org/10.1109/TIT.2019.2940666

[14] Bahare Kiumarsi, Kyriakos G Vamvoudakis, Hamidreza Modares, and Frank L Lewis. 2017. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems* 29, 6 (2017), 2042–2062.

[15] Roman Koller, Lukas Schmid, Nathan Matsuda, Thomas Niederberger, Leonidas Spinoulas, Oliver Cossairt, Guido Schuster, and Aggelos K Katsaggelos. 2015. High spatio-temporal resolution video with compressed sensing. *Opt. Express* 23, 12 (2015), 15992–16007.

[16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[18] Liang Liu, Hao Lu, Hongwei Zou, Haipeng Xiong, Zhiguo Cao, and Chunhua Shen. 2020. Weighing counts: Sequential crowd counting by reinforcement learning. In *European Conference on Computer Vision*. Springer, 164–181.

[19] Y. Liu, X. Yuan, J. Suo, D. J. Brady, and Q. Dai. 2019. Rank Minimization for Snapshot Compressive Imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 12 (Dec 2019), 2990–3006. https://doi.org/10.1109/TPAMI.2018.2873587

[20] Yang Liu, Xin Yuan, Jinli Suo, David J Brady, and Qionghai Dai. 2019. Rank minimization for snapshot compressive imaging. *IEEE transactions on pattern analysis and machine intelligence* 41, 12 (2019), 2990–3006.

[21] Patrick Llull, Xuejun Liao, Xin Yuan, Jianbo Yang, David Kittle, Lawrence Carin, Guillermo Sapiro, and David J. Brady. 2013. Coded aperture compressive temporal imaging. *Opt. Express* 21, 9 (May 2013), 10526–10545. https://doi.org/10.1364/OE.21.010526

[22] Sidi Lu and Weisong Shi. 2021. The Emergence of Vehicle Computing. *IEEE Internet Computing Magazine* (2021).

[23] Sidi Lu, Xin Yuan, and Weisong Shi. 2020. Edge compression: An integrated framework for compressive imaging processing on cavs. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 125–138.

[24] Sidi Lu, Xin Yuan, and Weisong Shi. 2020. An Integrated Framework for Compressive Imaging Processing on CAVs. In *ACM/IEEE Symposium on Edge Computing (SEC)*.

[25] Jiawei Ma, Xiaoyang Liu, Zheng Shou, and Xin Yuan. 2019. Deep Tensor ADMM-Net for Snapshot Compressive Imaging. In *IEEE/CVF Conference on Computer Vision (ICCV)*.

[26] Ziyi Meng, Jiawei Ma, and Xin Yuan. 2020. End-to-End Low Cost Compressive Spectral Imaging with Spatial-Spectral Self-Attention. In *European Conference on Computer Vision (ECCV)*.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[28] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. 2020. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712* (2020).

[29] Renaud Péteri, Sándor Fazekas, and Mark J Huiskes. 2010. DynTex: A comprehensive database of dynamic textures. *Pattern Recognition Letters* 31, 12 (2010), 1627–1632.

[30] D Poobathy and R Manicka Chezian. 2014. Edge detection operators: Peak signal to noise ratio based comparison. *IJ Image, Graphics and Signal Processing* 6, 10 (2014), 55–61.

[31] M. Qiao, Z. Meng, J. Ma, and X. Yuan. 2020. Deep learning for video compressive sensing. *APL Photonics* 5, 3 (2020), 030801. https://doi.org/10.1063/1.5140721

[32] D. Reddy, A. Veeraraghavan, and R. Chellappa. 2011. P2C2: Programmable pixel compressive camera for high speed imaging. In *CVPR 2011*. 329–336. https://doi.org/10.1109/CVPR.2011.5995542

[33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[34] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[35] Nicolas Saunier, Håkan Ardö, Jean-Philippe Jodoin, Aliaksei Laureshyn, Mikael Nilsson, Åse Svensson, Luis Miranda-Moreno, Guillaume-Alexandre Bilodeau, and Kalle Åström. 2014. A public video dataset for road transportation applications. In *Transportation Research Board Annual Meeting Compendium of Papers*. 14–2379.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[37] Victor do Nascimento Silva and Luiz Chaimowicz. 2017. MOBA: a new arena for game AI. *arXiv preprint arXiv:1705.10443* (2017).

[38] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[39] Y. Sun, X. Yuan, and S. Pang. 2017. Compressive high-speed stereo imaging. *Opt Express* 25, 15 (2017), 18182–18190. https://doi.org/10.1364/OE.25.018182

[40] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

[41] O Vinyals, T Ewalds, S Bartunov, P Georgiev, AS Vezhnevets, M Yeo, A Makhzani, H Küttler, J Agapiou, J Schrittwieser, et al. 2017. A New Challenge for Reinforcement Learning. *arXiv preprint ArXiv:1708.04782* (2017).

[42] Z. Wang, H. Zhang, Z. Cheng, B. Chen, and X. Yuan. 2021. MetaSCI: Scalable and Adaptive Reconstruction for Video Compressive Sensing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[43] Dong Xiao, Feng Shan, Ze Li, Ba Tuan Le, Xiwen Liu, and Xuerao Li. 2019. A Target Detection Model Based on Improved Tiny-Yolov3 Under the Environment of Mining Truck. *IEEE Access* 7 (2019), 123757–123764.

[44] J. Yang, X. Liao, X. Yuan, P. Llull, D. J. Brady, G. Sapiro, and L. Carin. 2015. Compressive Sensing by Learning a Gaussian Mixture Model from Measurements. *IEEE Transaction on Image Processing* 24, 1 (January 2015), 106–119.

[45] J. Yang, X. Yuan, X. Liao, P. Llull, G. Sapiro, D. J. Brady, and L. Carin. 2014. Video Compressive Sensing Using Gaussian Mixture Models. *IEEE Transaction on Image Processing* 23, 11 (November 2014), 4863–4878.

[46] P. Yang, L. Kong, X. Liu, X. Yuan, and G. Chen. 2020. Shearlet Enhanced Snapshot Compressive Imaging. *IEEE Transactions on Image Processing* 29 (2020), 6466–6481.

[47] Zhe Yang, Phuong Nguyen, Haiming Jin, and Klara Nahrstedt. 2019. MIRAS: Model-based reinforcement learning for microservice resource allocation over scientific workflows. In *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*. IEEE, 122–132.

[48] X. Yuan. 2016. Generalized alternating projection based total variation minimization for compressive sensing. In *2016 IEEE International Conference on Image Processing (ICIP)*. 2539–2543.

[49] X. Yuan, D. J. Brady, and A. K. Katsaggelos. 2021. Snapshot Compressive Imaging: Theory, Algorithms, and Applications. *IEEE Signal Processing Magazine* 38, 2 (2021), 65–88. https://doi.org/10.1109/MSP.2020.3023869

[50] X. Yuan, Y. Liu, J. Suo, and Q. Dai. 2020. Plug-and-Play Algorithms for Large-scale Snapshot Compressive Imaging. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[51] X. Yuan, J. Yang, P. Llull, X. Liao, G. Sapiro, D. J. Brady, and L. Carin. 2013. Adaptive Temporal Compressive Sensing for Video. In *2013 IEEE International Conference on Image Processing (ICIP)*. 14–18.

[52] Huanhuan Zhang, Anfu Zhou, Jiamin Lu, Ruoxuan Ma, Yuhan Hu, Cong Li, Xinyu Zhang, Huadong Ma, and Xiaojiang Chen. 2020. OnRL: improving mobile video telephony via online reinforcement learning. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.