An Anomaly Comprehension Neural Network for Surveillance Videos on Terminal Devices

Yuan Cheng*, Guangtai Huang†, Peining Zhen*, Bin Liu†, Hai-Bao Chen*, Ngai Wong‡, Hao Yu†
*Department of Micro/Nano Electronics, Shanghai Jiao Tong University, Shanghai, China
†Department of Electrical and Electronic Engineering, Southern University of Science and Technology, Shenzhen, China
‡Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

***Abstract*—Anomaly comprehension in surveillance videos is more challenging than detection. This work introduces the design of a lightweight and fast anomaly comprehension neural network. For comprehension, a spatio-temporal LSTM model is developed based on the structured, tensorized time-series features extracted from surveillance videos. Deep compression of network size is achieved by tensorization and quantization for the implementation on terminal devices. Experiments on large-scale video anomaly dataset UCF-Crime demonstrate that the proposed network can achieve an impressive inference speed of 266 FPS on a GTX-1080Ti GPU, which is 4.29 faster than ConvLSTM-based method; a 3.34% AUC improvement with 5.55% accuracy niche versus the 3D-CNN based approach; and at least $15k\times$ parameter reduction and $228\times$ storage compression over the RNN-based approaches. Moreover, the proposed framework has been realized on an ARM-core based IOT board with only 2.4W power consumption.**

***Index Terms*—anomaly comprehension, surveillance videos, AI on IOT**

I. Introduction

Fast comprehension (detect and classify) of low-probability anomalous (e.g., violent or illegal) events in surveillance videos are nontrivial [1], [2], especially on resource-limited terminal or edge devices. Among various works, a typical solution is to track motion cues in a video stream, such as color, texture and optical flow [3], [4]. Other approaches use mixtures of probabilistic principal components on sparse dictionary [5], k-Nearest Neighbours [1], as well as Gaussian regression [6] etc. Nonetheless, all these traditional methods are limited by the low-level cues of frame pixels with limited accuracy and speed. Recent schemes based on deep learning have successfully achieved anomaly detection in videos [7], [8]. They employ convolutional neural networks (CNNs) to process video frames one by one to extract features of objects, which however are not able to identify the temporal relationship of objects, i.e., activities. Other works use Recurrent Neural Networks (RNNs) as well as their Long Short-Term Memory (LSTM) variants to build networks with memory for anomalous activity detection [9]. However, the dense computation with raw data in RNN and LSTM incurs heavy memory and storage requirements, and hence is slow as well as inaccurate. Recently, two-stream CNN is applied [10] to separate spatial and temporal computation. Pre-trained CNNs [11] are used to generate non-structured features to construct RNNs. 3D-CNNs are also proposed [12], [13] to learn video representations by replacing the 2D convolution with 3D spatial convolutions. Nevertheless, all the aforementioned methods are limited to anomaly detection *without* classification, and cannot be realized on resource-limited edge or terminal devices.

This work is supported in part by the Nature Science Foundation of China (NSFC) under No. 61604095, and in part by a 985 research fund from Shanghai Jiao Tong University. Correspondence author: Hai-Bao Chen. Email: haibaochen@sjtu.edu.cn.

TABLE I
Comparison of Related Works.

	Approach	Detection		Classification	
		Object	Activity	Object	Activity
Non-AI	Traditional methods [1], *e.g., k-Nearest Neighbours*	×	✓	×	×
	Low-level motion cues [4], *e.g., texture, optical flow*	×	✓	×	×
AI	Single-frame features [7], *e.g., CNNs*	✓	✓	×	×
	Sequence information [9], *e.g., RNNs, 3D-CNN*	×	✓	×	✓
	Proposed framework	✓	✓	✓	✓

In this paper, our contributions are threefold:

- We devise a *novel* anomaly comprehension (viz. *detection+classification* on *both anomalous object and activity*) network for surveillance videos.
- We build an LSTM-based spatio-temporal model from *structured, tensorized time-series features* instead of directly working on the raw video frames or non-structured features.
- We achieve deep compression with tensor decomposition and trained quantization (8-bit), resulting in a lightweight realization on an ARM-core based IOT board.

Table I compares various related works and highlights the difference of the proposed framework. In our experiments, instead of using the typical frame-labeled anomaly dataset, we benchmark on the large-scale video-labeled dataset UCF-Crime [2]. Experiments show that the proposed framework can achieve an impressive inference speed of 266 FPS on a GTX-1080Ti GPU, which is 4.29 faster than ConvLSTM-based method; a 3.34% AUC improvement with 5.55% accuracy niche versus the 3D-CNN based approach [13]; and at least $15k\times$ parameter reduction and $228\times$ storage compression over the RNN-based approach [9].

In the following, Section II presents the spatio-temporal LSTM model processing the structured, tensorized time-series features. Section III elaborates the deep compression via tensor decomposition and trained quantization. Section IV presents

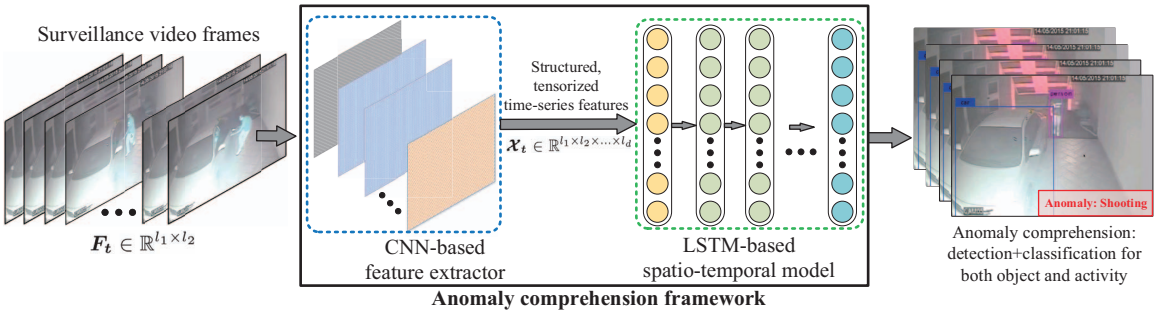

Fig. 1. The proposed framework of anomaly comprehension for surveillance videos.

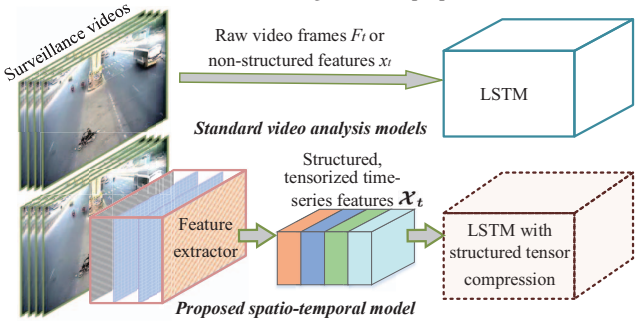

Fig. 2. Comparison between the standard methods with raw video frames or non-structured features and the proposed spatio-temporal LSTM model with structured, tensorized time-series features.

the network realization on an ARM-core based IOT board. Section V presents results on several large-scale datasets and Section VI draws the conclusion.

II. Spatio-temporal LSTM Model

This section elaborates the proposed anomaly comprehension framework shown in Fig. 1, whose design considers both the network algorithm and hardware realization. To achieve comprehension (detection+classification) on both object and activity in the videos, we introduce a spatio-temporal LSTM model as follows.

Structured, tensorized time-series features: Instead of using raw video frames or non-structured features, we construct a spatio-temporal LSTM model with the structured, tensorized time-series features aggregated from a CNN feature extractor. As shown in Fig. 2, the feature extractor uses several convolutional layers to learn the distilled features from raw video frames, namely $\boldsymbol{\mathcal{F}_t} \in \mathbb{R}^{l_1 \times l_2}$ (l the mode size of dimension). After that, we stack the deeply learned time-series features into a structured tensor $\boldsymbol{\mathcal{X}_t} \in \mathbb{R}^{l_1 \times l_2 \times \ldots \times l_d}$ (d the dimensionality of tensor) and feed it into the spatio-temporal LSTM model for the anomaly comprehension. The proposed process can be represented by: $extract(\boldsymbol{F_t}) = \boldsymbol{\mathcal{X}_t}$, where the $extract$ operation represents the corresponding extraction method in feature extractor. Then the LSTM cells (consisting of fully-connected layers) in the LSTM network take the structured, tensorized time-series features $\boldsymbol{\mathcal{X}_t}$ as inputs, instead of raw video frames $\boldsymbol{F_t}$ or non-structured features $\boldsymbol{x_t}$, to learn the spatio-temporal information. Each LSTM cell keeps track of an internal state that represents its memory and learns to update its state over time based on the current input and past states, as in the following:

$$\begin{aligned}
&\boldsymbol{\mathcal{S}_t} = \sigma(\boldsymbol{\mathcal{W}_s}\boldsymbol{\mathcal{X}_t} + \boldsymbol{\mathcal{U}_s}\boldsymbol{\mathcal{H}_{t-1}} + \boldsymbol{\mathcal{B}_s}),\ \boldsymbol{\mathcal{Z}_t} = \sigma(\boldsymbol{\mathcal{W}_z}\boldsymbol{\mathcal{X}_t} + \boldsymbol{\mathcal{U}_z}\boldsymbol{\mathcal{H}_{t-1}} + \boldsymbol{\mathcal{B}_z}),\\
&\boldsymbol{\mathcal{D}_t} = \sigma(\boldsymbol{\mathcal{W}_d}\boldsymbol{\mathcal{X}_t} + \boldsymbol{\mathcal{U}_d}\boldsymbol{\mathcal{H}_{t-1}} + \boldsymbol{\mathcal{B}_d}),\ \tilde{\boldsymbol{\mathcal{C}}}_t = tanh(\boldsymbol{\mathcal{W}_c}\boldsymbol{\mathcal{X}_t} + \boldsymbol{\mathcal{U}_c}\boldsymbol{\mathcal{H}_{t-1}} + \boldsymbol{\mathcal{B}_c}),\\
&\boldsymbol{\mathcal{C}_t} = \boldsymbol{\mathcal{S}_t} \odot \boldsymbol{\mathcal{C}_{t-1}} + \boldsymbol{\mathcal{Z}_t} \odot \tilde{\boldsymbol{\mathcal{C}}}_t,\ \boldsymbol{\mathcal{H}_t} = \boldsymbol{\mathcal{D}_t} \odot tanh(\boldsymbol{\mathcal{C}_t}),
\end{aligned} \tag{1}$$

where $\odot$ denotes the element-wise product, $\sigma(\circ)$ represents the sigmoid function and $tanh(\circ)$ represents the hyperbolic tangent function. $\boldsymbol{\mathcal{H}_{t-1}}$ and $\boldsymbol{\mathcal{C}_{t-1}}$ are the previous hidden state and previous update factor, $\boldsymbol{\mathcal{H}_t}$ and $\boldsymbol{\mathcal{C}_t}$ are the current hidden state and current update factor, respectively. The weight matrices $\boldsymbol{\mathcal{W}_*}$ and $\boldsymbol{\mathcal{U}_*}$ weigh the input $\boldsymbol{\mathcal{X}_t}$ and the previous hidden state $\boldsymbol{\mathcal{H}_{t-1}}$ to update factor $\tilde{\boldsymbol{\mathcal{C}}}_t$ and three sigmoid gates, namely, $\boldsymbol{\mathcal{S}_t}$, $\boldsymbol{\mathcal{Z}_t}$ and $\boldsymbol{\mathcal{D}_t}$. It should be noted that all these parameters have been performed with tensorization.

For each frame in video anomaly comprehension, the LSTM cell computes their information by combining previous states and current features. Therefore, all video information can be captured from the beginning till the current frame. However, the proposed framework still suffers from high dimensional inputs and huge number of parameters, making it hard to train and susceptible to overfitting. To solve this, a deep compression using the tensor decomposition and 8-bit quantization is applied to the whole framework, described in the next section.

III. Deep Compression

The proposed framework is highly integrated and compressed to be hardware friendly and execute fast on edge or terminal devices. To satisfy these requirements, we systematically develop the tensorized and quantized compression on the spatio-temporal LSTM model.

A. Structured Tensor Compression

A tensor is a d-dimensional generalization of a vector or matrix, denoted by calligraphic letters $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{l_1 \times l_2 \times \ldots \times l_d}$ where $\boldsymbol{\mathcal{X}}(h_1, h_2, \ldots, h_d)$ is an element specified by the indices $h_1, h_2, \ldots, h_d$.

The total number of elements is $l_1 l_2 \cdots l_d$ which grows exponentially as d increases. In practice, tensor decomposition is used to find a low-rank approximation that expresses the original tensor by a number of small tensor factors. This often reduces the computational complexity from exponential to only linear, thereby eluding the *curse of dimensionality*. In our proposed framework, the initial inputs of spatio-temporal

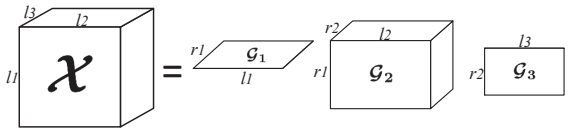

Fig. 3. Decomposing a 3-way tensor into tensor cores.

model are time-series features, which are already structured into a 3-dimensional tensor. In fact, we adopt tensor train [14] as the structured tensor compression method, which scales nicely with the tensor dimensions. Given a d-dimensional tensor $\boldsymbol{\mathcal{X}}$, the decomposition reads

$$\begin{aligned}\boldsymbol{\mathcal{X}}(h_1, h_2, \ldots, h_d) = \sum_{\alpha_1,\ldots,\alpha_{d-1}}^{r_1,\ldots,r_{d-1}} & \boldsymbol{\mathcal{G}_1}(1, h_1, \alpha_1) \\ & \boldsymbol{\mathcal{G}_2}(\alpha_1, h_2, \alpha_2) \ldots \boldsymbol{\mathcal{G}_d}(\alpha_{d-1}, h_d, 1),\end{aligned} \tag{2}$$

where $\boldsymbol{\mathcal{G}_k} \in \mathbb{R}^{r_{k-1} \times l_k \times r_k}$ is the tensor core and r_k is the tensor train rank, α_k is the summation index ranging from 1 to r_k. Using the notation $\boldsymbol{\mathcal{G}_k}(h_k) \in \mathbb{R}^{r_{k-1} \times r_k}$ (a matrix slice from the 3-dimensional tensor $\boldsymbol{\mathcal{G}_k}$), (2) can be written compactly as

$$\boldsymbol{\mathcal{X}}(h_1, h_2, \ldots, h_d) = \boldsymbol{\mathcal{G}_1}(h_1)\boldsymbol{\mathcal{G}_2}(h_2) \ldots \boldsymbol{\mathcal{G}_d}(h_d). \tag{3}$$

The decomposition of a 3-dimensional tensor is intuitively shown in Fig. 3. Since each integer l_k in (3) can be further decomposed as $l_k = n_k \cdot m_k$, each tensor core $\boldsymbol{\mathcal{G}_k}$ can be reformed with $\boldsymbol{\mathcal{G}_k^t} \in \mathbb{R}^{n_k \times m_k \times r_{k-1} \times r_k}$, and $\boldsymbol{\mathcal{G}_k^t}(j_k, i_k) \in \mathbb{R}^{r_{k-1} \times r_k}$. Therefore, the decomposition for the tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{(n_1 \times m_1) \times (n_2 \times m_2) \times \ldots \times (n_d \times m_d)}$ can be reformulated as:

$$\boldsymbol{\mathcal{X}}((j_1, i_1), (j_2, i_2), \ldots, (j_d, i_d)) = \boldsymbol{\mathcal{G}}_1^t(j_1, i_1)\boldsymbol{\mathcal{G}}_2^t(j_2, i_2) \ldots \boldsymbol{\mathcal{G}}_d^t(j_d, i_d). \tag{4}$$

Such double-index trick is then used to tensorize an LSTM. Specifically, the most costly computation in LSTM is the large-scale matrix-vector multiplication generically represented as $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ where $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ is the weight matrix, $\boldsymbol{x} \in \mathbb{R}^M$ is the feature vector, $\boldsymbol{b} \in \mathbb{R}^N$ is the bias vector. To approximate $\boldsymbol{W}\boldsymbol{x}$ with much fewer parameters, we first reshape $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ into a tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{(n_1 \times n_2 \times \cdots \times n_d) \times (m_1 \times m_2 \times \cdots \times m_d)}$, where $N = \Pi_{k=1}^d n_k$ and $M = \Pi_{k=1}^d m_k$. Following (4), $\boldsymbol{\mathcal{W}}(h_1, h_2 \cdots, h_d)$ can be rewritten as $\boldsymbol{\mathcal{G}}_1^t(j_1, i_1)\boldsymbol{\mathcal{G}}_2^t(j_2, i_2) \ldots \boldsymbol{\mathcal{G}}_d^t(j_d, i_d)$. Similarly, we can reshape $\boldsymbol{x} \in \mathbb{R}^M$, $\boldsymbol{b} \in \mathbb{R}^N$ into d-dimensional tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_d}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$. As a result, the output $\boldsymbol{y} \in \mathbb{R}^N$ also becomes a d-dimensional tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$. Therefore, the matrix-vector multiplication is now in the tensor form with usually low-rank cores

$$\begin{aligned}\boldsymbol{\mathcal{Y}}(j_1, j_2, \ldots, j_d) = \sum_{i_1=1}^{m_1} \sum_{i_2=1}^{m_2} \cdots \sum_{i_d=1}^{m_d} & [\boldsymbol{\mathcal{G}}_1^t(j_1, i_1)\boldsymbol{\mathcal{G}}_2^t(j_2, i_2) \ldots \\ & \boldsymbol{\mathcal{G}}_d^t(j_d, i_d)\ \boldsymbol{\mathcal{X}}(i_1, i_2, \ldots, i_d)] + \boldsymbol{\mathcal{B}}(j_1, j_2, \ldots, j_d).\end{aligned} \tag{5}$$

Therefore, all fully-connected matrix-vector products in Eq. (1) can be tensorized similarly to Eq. (5).

B. Trained Quantization

The original network with full-precision parameters requires unnecessarily large software and hardware resources. Here we present an 8-bit quantization strategy on the whole framework for further compression (we determine the best quantization bitwidth by testing all realizations from 4-bit to 10-bit). Previous work [15] has suggested using quantization constraints during neural network training called *trained quantization*. Since the main parameters in an LSTM are weights and features, a quantized LSTM with 8-bit weights and features can result in high compression and efficiency. Assuming w_k is the full-precision weight entry, it can be quantized into its 8-bit counterpart w_k^q as:

$$w_k^q = \begin{cases} \frac{w_k}{|w_k|}, & 0 < |w_k| \leq \frac{1}{2^7}, \\ floor(2^7 \times w_k), & \frac{1}{2^7} < |w_k| < 1, \\ (2^7 - 1)\frac{w_k}{|w_k|}, & |w_k| \geq 1, \\ 0, & |w_k| = 0, \end{cases} \tag{6}$$

where the function $floor$ takes the smaller nearest integer. We also enforce 8-bit features by quantizing a real feature element x_k into its 8-bit $x_k^q \in [0, 1]$:

$$x_k^q = \frac{1}{2^8} \times \begin{cases} floor(2^8 \times x_k), & 0 \leq x_k < 1, \\ 2^8 - 1, & x_k \geq 1. \end{cases} \tag{7}$$

The batch normalization and max-pooling layers are also quantized into 8-bit similarly.

IV. Anomaly Comprehension on Terminal Devices

The proposed framework integrates the 8-bit-quantized feature extractor and the tensorized and 8-bit-quantized LSTM (named tqLSTM) which takes structured, tensorized time-series features for anomaly comprehension. The overall workflow is shown in Fig. 4, first, the video clip is fed into the 8-bit-quantized feature extractor as inputs. Then, the extracted time-series features are structured into a tensor and fed into the tqLSTM. We note in passing that the time-series features are coming from the *last* convolution layer of the CNN-based feature extractor, which can also be processed for

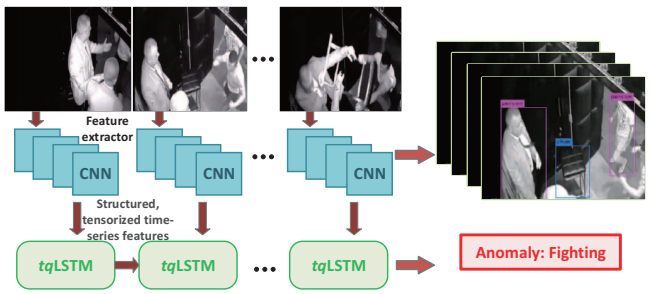

Fig. 4. Workflow of the proposed anomaly comprehension for surveillance videos on terminal devices.

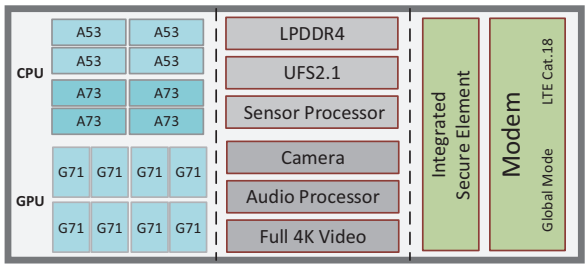

Fig. 5. Architecture of the used ARM-core based IOT terminal device.

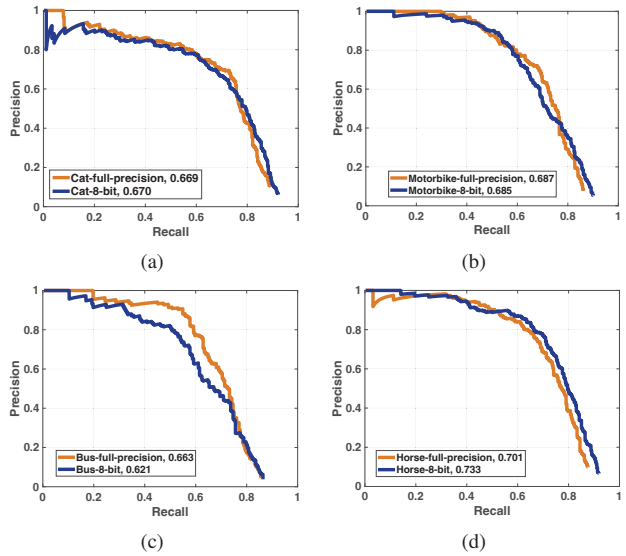

Fig. 6. Average precision comparison between 8-bit feature extractor qYOLO and full-precision YOLOv2-tiny on classes: (a) cat, (b) motorbike, (c) bus and (d) horse.

the real-time object detection and classification. Finally, the detection+classification results on both object and activity towards the anomaly comprehension are obtained.

Remarkable speedup, energy-efficiency and saving in resources are delivered by the tensorization and quantization, which make the proposed framework lightweight and hardware friendly. To realize the anomaly comprehension on terminal devices, we use an ARM-core board with 6GB RAM memory and 64 GB ROM storage. This board contains a cluster of four ARM Cortex-A73 cores with 2.36 GHz working frequency and four ARM Cortex-A53 cores with 1.80 GHz working frequency, whose architecture is shown in Fig. 5. These cores are organized in a big.LITTLE configuration [16].

V. Experiments

We develop the feature extractor based on YOLOv2-tiny [17] and its 8-bit-quantized version is called qYOLO. We first evaluate the proposed anomaly comprehension framework on GPU. The initialization environment is set as: Theano in Keras with a single NVIDIA GTX-1080Ti. Benchmarking is made on the large-scale anomalous activity dataset UCF-Crime [2] and the object dataset VOC [18]. The UCF-Crime contains $1,900$ untrimmed surveillance videos with 128 hours in total, falling into 13 real-world anomalies (*Abuse*, *Robbery*, *Stealing*, *Fighting*, etc.) of importance for public safety. The dataset VOC contains $11,530$ images with $27,450$ annotated objects, falling into 20 subclasses.

A. Evaluation on Object

For the anomaly comprehension on object, we use the Average Precision (AP) score to evaluate object detection and classification. The AP scores on several representative subclasses in VOC of the proposed 8-bit feature extractor and the full-precision one are reported in Fig. 6. It can be seen that the 8-bit qYOLO produces AP curves close to the full-precision ones. Due to the outstanding performance, we can further develop it to detect and classify anomalous objects in surveillance videos based on an anomalous object dataset (a customized one is introduced in Section V-D).

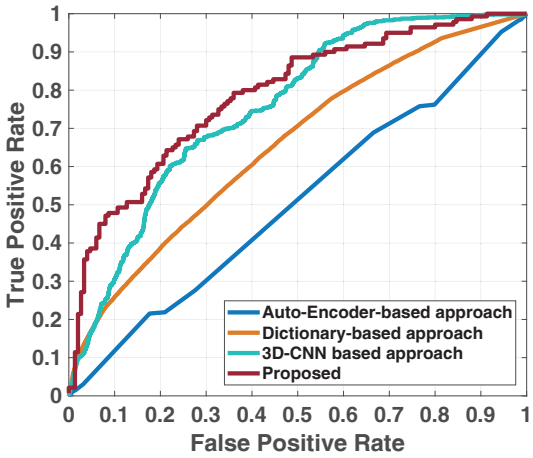

Fig. 7. ROC curves for UCF-Crime by various popular schemes.

TABLE II
AUC comparison with state-of-the-arts on UCF-Crime.

	Approach	AUC
Traditional methods	Auto-Encoder-based [3]	50.60
	Dictionary-based [5]	65.51
Single-frame features	CNN [7]	66.28
	ConvLSTM [8]	59.64
	ConvLSTM-AE [19]	61.92
Sequence information	Vanilla-RNN [9]	55.36
	Original LSTM [20]	57.06
	Two-stream CNN [10]	69.59
	3D-CNN [13]	75.41
Traditional CNN+RNN	CNN+RNN [11]	72.64
Proposed	qYOLO+tqLSTM	**78.75**

B. Evaluation on Activity

We now demonstrate that a *unique* benefit arises from the use of structured, tensorized time-series features, namely *accuracy improvement under deep compression*. Similar to most existing works, we use frame-based Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC) to evaluate the accuracy. Here we report the experimental results using UCF-Crime in three scenarios.

Anomalous activity detection – We use 800 normal and 810 anomalous videos for training, and 150 normal and 140 anomalous videos for testing. The tqLSTM output is a binary classifier: *normal* or *anomaly*. Several representative ROC curves are shown in Fig. 7, and the AUC comparisons against other popular schemes are shown in Table II. It can be seen that tqLSTM significantly outperforms all approaches. The AUC of the proposed qYOLO+tqLSTM framework can reach 78.75% with 3.34% higher than the next best 3D-CNN approach and 16.83% higher than the ConvLSTM-AE based approach.

Specific anomalous activity detection – We use 800 normal and $50 \sim 150$ anomalous videos from one of the 13 subclasses to train tqLSTM. The tqLSTM output is a binary classifier: *normal* or *activity-specific anomaly*. For this specific anomalous activity detection, two state-of-the-art detectors

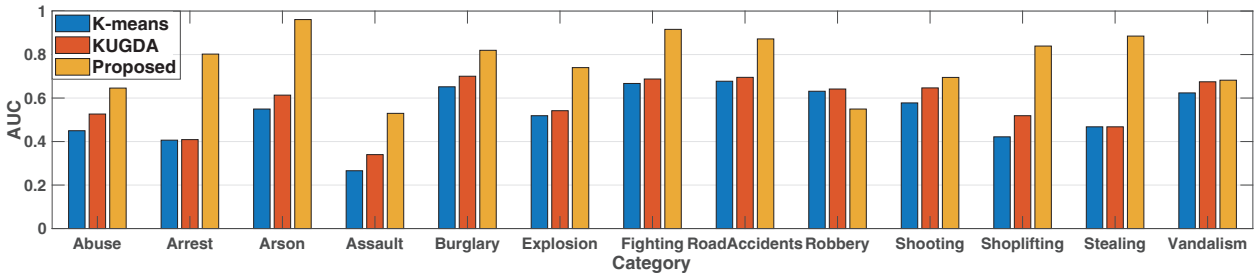

Fig. 8. AUC comparison with state-of-the-arts on each anomalous activity subclass in the UCF-Crime dataset.

TABLE III
ACCURACY COMPARISON ON ANOMALOUS ACTIVITY CLASSIFICATION.

Approach	TCNN [21]	3D-CNN [13]	qYOLO+tqLSTM
Accuracy	23.00%	28.40%	**33.95%**

TABLE IV
PERFORMANCE EVALUATION ON COMPRESSION: THE NUMBER OF PARAMETERS INVOLVED IN THE INPUT-TO-HIDDEN MAPPINGS.

Approach	Parameters	Storage size
Vanilla-RNN [9]	95,656,800	976.9MB
Original LSTM [9]	29,491,200	661.5MB
ConvLSTM [8]	41,834,200	783.4MB
ConvLSTM-AE [19]	32,191,600	712.1MB
GRU [9]	30,118,400	691.3MB
qYOLO+tqLSTM	**1960**	**2.9MB**

TABLE V
THE PERFORMANCE EVALUATION ON INFERENCE SPEED: THE RUNTIME INVOLVED IN THE REAL-TIME PROCESSING OF A SAMPLED SURVEILLANCE VIDEO IN 12 MINS.

Approach	Runtime	FPS
Dictionary-based [5]	148.8s	150
ConvLSTM-AE [19]	362.3s	62
3D-CNN [12]	171.6s	130
Original LSTM [9]	156.0s	143
Traditional CNN+RNN [11]	194.1s	116
qYOLO+tqLSTM	**83.9s**	**266**

are chosen for comparison: a K-means approach [1] and a KUGDA approach [1]. Fig. 8 shows the AUC comparison on all 13 anomaly subclasses. The proposed scheme reaches an average AUC of 76.43%, which is 19.02% higher than KUGDA and 23.29% higher than K-means.

Anomalous activity classification – We use the same training data as in the specific anomalous activity detection case, but this time the tqLSTM has 14 outputs, namely, it *classifies one of the* 13 *anomalies* or *normal*. Since the UCF-Crime videos are long, low-resolution and noisy, all schemes perform poorly. Nonetheless, it can be seen from Table III that tqLSTM still outperforms others, and in particular 5.55% higher than 3D-CNN based approach [2].

C. Performance Quantification

Besides the excellent capability and accuracy of the proposed framework by qYOLO+tqLSTM, the model compression and its operating speed are also remarkable compared to existing methods. Table IV shows the comparison on storage size. Evidently, tqLSTM achieves excellent compression which fares at least $15k\times$ parameter reduction and $228\times$ storage compression versus other RNN-based approaches. Note that reducing the parameters from tens of millions to thousands helps model to achieve remarkable speedup as well as saving in resources. We also observe from Table V that tqLSTM runs at 266 FPS (Frames Per Second), which is considered to be *"very fast"* for surveillance videos.

Fig. 9. Realization of the proposed framework on an ARM-core based IOT terminal device.

D. Evaluation on IOT Board

After the evaluation on GPU, we further report the experiment results of the proposed anomalous activity comprehension framework on an ARM-core based IOT board. The test verification platform is shown in Fig. 9, which comprises of an ARM-core based IOT board, a DC regulated power supply and a computer monitor. Using a customized anomaly dataset (object-labeled with $fire$ and $smoke$, activity-labeled with $explosion$), we can obtain the visual results of anomaly from the monitor, as seen in Fig. 9. When the anomalous activity comprehension is running, the memory consumption is up to 700M, consuming only 12% of all 6GB RAM memory. Furthermore, the average power consumption of the proposed framework is measured to be 2.4W. By sampling some surveillance videos using the proposed anomaly comprehension framework, the visual results of detection+classification on both object and activity are shown in Fig. 10. We observe that all existing objects and activities in these videos can be detected and classified precisely in real time.

VI. CONCLUSION

We have proposed a lightweight yet accurate anomaly comprehension network for surveillance videos. The network

Fig. 10. Sample visual results of the proposed framework for anomaly comprehension on UCF-Crime.

is built on top of a tensorized and (8-bit) quantized LSTM network processing structured, tensorized time-series features extracted from a (8-bit) quantized CNN. We have implemented the proposed network on an ARM-core based IOT board with only 2.4W power consumption. Such CNN+RNN solution is proposed for the first time in anomaly comprehension, which readily outperforms other state-of-the-arts on the large-scale UCF-Crime dataset: inference speed of 266 FPS on a single-GPU machine; a 3.34% AUC improvement with 5.55% accuracy niche versus the 3D-CNN based approach; and at least $15k\times$ parameter reduction and $228\times$ storage compression over RNN-based approaches.

REFERENCES

[1] M. U. K. Khan, H.-S. Park, and C.-M. Kyung, "Rejecting motion outliers for efficient crowd anomaly detection," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 541–556, 2019.

[2] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," *Center for Research in Computer Vision, University of Central Florida*, 2018.

[3] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 733–742.

[4] E. Ricci, G. Zen, N. Sebe, and S. Messelodi, "A prototype learning framework using emd: Application to complex scenes analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 513–526, 2013.

[5] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *IEEE Conference on Computer Vision*, 2013, pp. 2720–2727.

[6] K.-W. Cheng, Y.-T. Chen, and W.-H. Fang, "Video anomaly detection and localization using hierarchical feature representation and gaussian process regression," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2909–2917.

[7] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Computer Vision and Image Understanding*, vol. 172, pp. 88–97, 2018.

[8] J. R. Medel and A. Savakis, "Anomaly detection in video using predictive convolutional long short-term memory networks," *arXiv preprint arXiv:1612.00390*, 2016.

[9] D. Arifoglu, "Activity recognition and abnormal behaviour detection with recurrent neural networks," *Procedia Computer Science*, vol. 110, pp. 86–93, 2017.

[10] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[11] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional lstm with cnn features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.

[12] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette, "Deep-cascade: cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1992–2004, 2017.

[13] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "Multi-fiber networks for video recognition," in *European Conference on Computer Vision*, 2018, pp. 352–367.

[14] Y. Yang, D. Krompass, and V. Tresp, "Tensor-train recurrent neural networks for video classification," *arXiv preprint arXiv:1707.01786*, 2017.

[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *arXiv preprint arXiv:1609.07061*, 2016.

[16] S. Kamdar and N. Kamdar, "big. little architecture: Heterogeneous multicore processing," *International Journal of Computer Applications*, vol. 119, no. 1, pp. 1278–1284, 2015.

[17] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2017.

[18] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[19] W. Luo, W. Liu, and S. Gao, "Remembering history with convolutional lstm for anomaly detection," in *IEEE International Conference on Multimedia and Expo*, 2017, pp. 439–444.

[20] M. Hasan and A. K. Roy-Chowdhury, "Incremental activity modeling and recognition in streaming videos," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 796–803.

[21] R. Hou, C. Chen, and M. Shah, "Tube convolutional neural network (t-cnn) for action detection in videos," in *IEEE International Conference on Computer Vision*, 2017.