Anomaly Detection in Residential Video Surveillance on Edge Devices in IoT Framework

Mayur R. Parate, Kishor M. Bhurchandi, Member IEEE, and Ashwin G. Kothari, Senior Member, IEEE

Abstract—Intelligent resident surveillance is one of the most essential smart community services. The increasing demand for security needs surveillance systems to be able to detect anomalies in surveillance scenes. Employing high-capacity computational devices for intelligent surveillance in residential societies is costly and not feasible. Therefore, we propose anomaly detection for intelligent surveillance using CPU-only edge devices. A modular framework to capture object-level inferences and tracking is developed. To cope with partial occlusions, posture deformations, and complex scenes, we employed feature encoding and trajectory association governed by two metrics complementing to each other. The elements of an anomaly detection framework are optimized to run on CPU-only edge devices with sufficient frames per second (FPS). The experimental results indicate the proposed method is feasible and achieves satisfactory results in real-life scenarios.

Keywords— Feature encoding, Trajectory association, Anomaly detection, Edge computing, Only-CPU Edge device.

I. INTRODUCTION

The recent concept of Smart Cities influences the town planners, investigators and researchers to strengthen the security and safety of a resident. Increasing demand for residential security needs an infrastructure of surveillance cameras deployed for video analysis. A leading challenge in analyzing surveillance videos is the detection of abnormal events or anomalies which require exhaustive human efforts. Most of these surveillance applications create massive context-ual data that require significant storage and computing resources. An Internet-of-Things (IoT) framework provides excellent flexibility and scalability to handle increasing surveillance needs. Embedding smart surveillance in IoT framework needs computer vision tasks such as human detection, classification, and tracking to be performed essentially at the perception or sensor layer of an IoT architecture to ensure response time, accuracy, and energy efficiency. In safety-critical applications such as anomaly detection, response time is an important parameter. The visual tasks at the sensor layer of an IoT architecture depend on the ability to capture, process visual data, and making decisions in real-time. Thus, it solves the application problem using only a single available frame without storing the intermediate frames for processing. This need is not fulfilled by cloud computing, although high accuracy can be achieved for visual tasks. The need of extensive bandwidth and inevitable network latency will prevent real-time response for targeting moving objects [1]. Some approaches have been proposed for combining the edge and cloud to reduce network traffic and latencies [2], however, due to the data-intensive nature of computer vision tasks, the network latencies are inevitable.

Convolutional Neural Networks (CCN) are becoming popular as they provide near-human accuracy for object detection and tracking. Convolutional Siamese Network, a type of CNN is used in modern object detection and tracking, establishing benchmark [3], [4]. However, to achieve the required accuracy, CNN-based object detection and tracking depend on the high processing power of underlying hardware/computers. They involve a huge quantity of computations making them unsuitable for edge devices. Employing edge computing for surveillance is considered as the answer to these shortcomings [6], [7]. Consequently, its integration with the IoT framework possesses the following advantages: real-time response, reduced network workload, lower power consumption, and higher data security and privacy. Despite the promising benefits of edge computing, one of the critical challenges is how to efficiently process the data-intensive computer vision tasks in real-time on a resource-constrained device. Various smart video surveillance approaches for object detection and tracking propose to use Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) algorithms and they usually have huge computational requirements. How to migrate these data-intensive and high-computing tasks to the edge devices at IoT nodes are still significant challenge. The research contributions of this work are as follows:

1) A lightweight framework for anomaly detection in residential video surveillance is developed for CPU-only edge devices to achieve real-time performance.
2) We employ spatio-temporal features for trajectory association based on two metrics to cope with partial occlusions, posture deformations, and complex scenes.
3) We optimized framework elements to run on CPU-only edge device and effect of configuration parameters on the overall accuracy and efficiency of the anomaly detection system is analyzed.
4) We integrated the surveillance system into an IoT framework for real-time automated alerts and analyzed the performance of anomaly detector on real-world residential surveillance video streams for CPU-only edge devices.

The rest of the paper is organized as follows: in section II we present the learning-based approaches for anomaly detection and then the attempts made for anomaly detection at the Edge/terminal devices are discussed. Section III presents the methodology and framework for the proposed anomaly detection. Experimental analysis for different configurations of parameters is presented in Section IV. Results and benchmarking on standard datasets are presented in Section V. Finally, Section VI concludes this paper.

II. RELATED WORK

Currently, most of the video surveillance systems function as an archive of footage and need huge storage and processing in the background. Very few are real-time smart surveillance systems that perform object detection and tracking tasks with the backbone of networking servers and Cloud operations. It has been recognized that heavy communication overhead observed in the cloud is not tolerable in many delay-sensitive, mission-critical tasks such as anomaly detection [8]. The terminal processing at edge devices for automated surveillance is considered to be the alternative for cloud and network-based processing [6], especially when response time is a relatively important parameter. Over the decade, some approaches have been crafted for automated surveillance/object tracking using edge devices. However, very few of them talk about the detection of an anomaly in video surveillance.
Object detection and tracking solutions range from traditional computer vision algorithms to more recent applications of learning-based approaches. Traditional algorithms; Scale Invariance Feature Transformation (SIFT) performs human detection using distinctive invariant features that are used to reliably match different views of an object or scene [12]. Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection [13]. Further, HOG+SVM algorithm [14] shows better performance in human detection. To detect crowd anomalies, histogram of magnitude and momentum [15]. Spatio-temporal motion pattern models with Kernelized SVM [16] are utilized. Simone et al. modeled the bio-inspired interaction to understand behaviour and anomalies in the crowd [17]. Whereas, the potential of modern learning-based strategies has led computer vision to a new extent during the last few years [18]. Deep learning methods and Convolutional Neural Networks (CNN) along with recent computational advancement has opened a new way to handle computer vision tasks such as object tracking [19]. Although the training phase usually demands high computation capability, the benefits of CNN can be exploited using computationally constrained hardware resources once the CNN has been trained. Few approaches have proposed the hardware-efficient neural networks to deliver state-of-the-art performance on embedded systems [20], [21], [22], [23].

Recently, learning-based approaches specifically designed for anomaly detection are observed in the literature [3], [4], [24], [25] detected anomalies in traffic surveillance, utilizing background modeling followed by Faster RCNN to detect vehicles in the extracted background and decide new anomalies under certain conditions. Following an unsupervised approach, [25] used an autoencoder model trained to minimize the reconstruction error between the input and the generated output. Further, the scalability issues due to the different-sized anomaly and their occurrence in a short time in pedestrian pathways are resolved by Region-based Scalable Convolution Neural Network (RS-CNN) [27]. An AnomalyNet in [3] detects anomaly by the synergic association of feature learning, sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM). Despite the rapid development computer vision applications using sparse representation using sparse long short term memory (SLSTM).
C. Object Detection

The object detection model is a version of a pre-trained SSD MobileNet v1 [23] model optimized for low computational overload. Depthwise separable convolution [23] technique reduces the computational overload in the MobileNet. A standard convolution is factorized into a depthwise convolution and pointwise convolution which is a 1x1 convolution. In standard convolution, inputs are filtered and combined into one step to generate the output whereas, depthwise separable convolution performs the same operation into two steps, a separate step for filtering and a separate step for combining. This reduces the computations of depthwise separable convolution as compared to the standard convolution by factor of $\frac{K^2 + 1}{K^2}$. Where, $Oc$ is the number of output channels and $D_{c} \times D_{c}$ is the kernel size. We use images of size 300x300 as an input to the MobiNet and from the output, the detection corresponding to the category of 'person' is used for feature extraction and trajectory association.

D. An Association Problem

The trajectory association must be performed separately on every detected person, which scales the algorithm linearly with the number of persons to be tracked. We use a single hypothesis tracking and data association in consecutive frames for trajectory association. In this framework, the trajectory is defined on the eight dimensional state space $(u, v, y, h, x, \dot{x}, \dot{y}, \dot{h})$. Where, $(u, v)$ represents the object bounding box, $y$ indicates aspect ratio, $h$ indicates height and $(\dot{x}, \dot{y}, \dot{h})$ indicate the respective velocities in the image coordinates.

We use a conventional Kalman filter using linear observation model, and constant velocity motion with object state observation defined by $(u, v, y, h)$. The trajectory $T_k$ over $t$-frames corresponding to a $k$-th bounding box detection $d_k$ is considered to be active if $i < I_{max}$ since last successful trajectory association $a_k$. Here, $I_{max}$ is the predefined value and indicate the maximum number of frames without association for which the trajectory is alive. For this, we increment the count $i$ for each track during kalman filter prediction since the last successful $a_k$ and reset to 0 at the successful $a_k$.

To solve the association problem, we integrate spatial and temporal information by defining two metrics; first, $C^l(m, l)$ representing Mahalanobis distance between the predicted kalman states and newly arrived measurement. Second, $C^r(l, m)$ represents a minimum cosine distance between the appearance descriptors $r_k^l \in \mathbb{R}^{128}$ computed over each bounding box detection $d_k$ and set of descriptors computed at each frame for each trajectory $k$. The combined metric for trajectory association is given by:

$$C_{(k,m)} = \lambda C^l_{(k,m)} + (1 - \lambda) C^r_{(k,m)} \quad (1)$$

Where, $\lambda$ is the hyperparameter to control the influence of each metric.

The motion information $C^m_{(k,m)}$ is further expanded using Mahalanobis distance as,

$$C^m_{(k,m)} = (d_m - y_i)^T S_i^{-1} (d_m - y_i) \quad (2)$$

Where, we represent the projection of the $l$-th track distribution into measurement space $(y_i, S_i)$ and $m$-th detection $d_m \in \mathbb{R}^5$. Practically, it calculates the deviation from the mean track location in terms of the standard deviations. The Mahalanobis distance provides

![Fig. 3: Workflow of the proposed anomaly detector](image)

**TABLE I: CNN Architecture to extract feature descriptor.**

| Layer               | Size | Stride | Output |
|---------------------|------|--------|--------|
| Conv                | 3X3  | 1      | 32X64X32 |
| Conv                | 3X3  | 2      | 32X32X16 |
| Max Pool            | 3X3  | 2      | 64X16X8  |
| Residual            | 3X3  | 2      | 64X16X8  |
| Residual            | 3X3  | 2      | 128X8X4  |
| Dense               | 3X3  | 2      | 128X8X4  |
| BathNormalization   |      |        | 128     |

accurate association when the motion uncertainty is low. While, the unaccounted camera motion and occlusions induce error in the metric. Further, the predicted state distribution from the kalman filter does not estimate accurate object location. Therefore, the second metric $C^r_{(k,m)}$ is integrated in (1). It represents the minimum cosine distance calculated between object descriptor $r_k^l \in \mathbb{R}^{128}$ corresponding to each bounding box detection $d_k$ and a set of descriptors $R_k = (r_k^l, l_k)$ for each track $k$. So, for $l$-th track and $m$-th bounding box detection, the metric is defined as,

$$C^r_{(k,m)} = \min\{1 - r_k^l r_m^l | r_m^l \in R_k \} \quad (3)$$

In practice, we trained a Convolutional Neural Network (CNN) model as shown in Table-1 on a separate dataset to extract an object descriptor for a given bounding box detection. The combined metric (1), serves both aspects of the assignment problem; the Mahalanobis distance investigates object locations based on motion that are especially important for short-term predictions. While, the cosine distance examines appearance information that is useful in recovery of identities after long-term occlusions, especially when motion is less discriminative. Thus, the associated trajectories are robust to appearance changes, fast motion and occlusions. These trajectories and dynamics in them are employed to detect anomalies in the scene.

E. Anomalies in the residential video surveillance

Premeditated anomalies are defined based on the person’s movement in the typical residential video surveillance streams as follows.

- A person standing still for a long time
- A person with a fast motion
- A person with circular/spiral movements
- A person standing still for a long time

We consider, standing still for a considerable time does not fit the normal behavior of a human being and it looks suspicious. If the person’s tracking trajectory is confined in a very small area for a long time, we define that as an anomaly.

- A person with a fast motion

In residential campuses, running is not a usual practice. If a person exceeds the finite fluctuation around the average velocity, we consider this behavior as anomalous.

- A person with the circular/spiral movements

The spiral motion of a person is very rare and surely suspicious. Generally, such motion is observed when anybody is inspecting...
something or someone. The tracking trajectory corresponding to a person is utilized, if it is circular in nature the motion is considered as circular/spiral.

- A person with jumping from the fence or gate
- Unauthorized entry or trespassing into the residential campus is always attempted by jumping a fence or gate. If the vertical fluctuations in the trajectory exceed the average vertical fluctuations, the situation is declared as an anomaly.
- Gathering of four or more persons simultaneously
- Gathering in a residential campus is common, however, simultaneous gathering of four or more persons is not expected as common event and thus considered abnormal. Trajectories with different identities approaching towards the common point are identified and declared as anomalies.
- Dispersion of four or more persons simultaneously
- Dispersion is the same as gathering except for the directions of trajectories are opposite and are extending from the common meeting point.

F. Anomaly Detection

Consider, the set of predefined anomalies $A = \{A_1, A_2, \ldots, A_N\}$ and the estimated set of trajectories $T = \{T^1, T^2, \ldots, T^K\}$ corresponding to $k$ detections. The optimal similarity between them is defined by the function $\omega^* : \{0, \ldots, N-1\} \rightarrow \{0, \ldots, K-1\}$ that maximizes the similarities between their elements:

$$\omega^* = \max_{w \in W} \Theta(w) = \max_{w \in W} \sum_j \text{sim}(A(j)T(w(j))) \tag{4}$$

Where, $w(j)$ provides position of the elements in $T$. In practice, we used a positive cosine distance to calculate similarity in (4). The set $W$ of possible shifts $w$ is updated recursively so that, their computations are efficient and tolerant to some deformations. So, if any of the aforementioned situations occur, the anomaly is declared and the alert is sent to the authorized personnel via the LoRa-based IoT-framework. Further, the only data transmitted using LoRa network is the alert of anomaly detection securing complete privacy which is sensitive in residential surveillance.

IV. EXPERIMENT

The proposed anomaly detection approach employs SSD MobileNet for person detection, as it provides good performance and accuracy. We compared the performance of two versions v1 and v2 of MobileNet at normal CPU clock and also with the overclocked settings on Raspberry Pi 4B. During the tests, the CPU is overclocked from 1500MHz to 1900MHz to analyze the boost in performance of MobileNet at these clock rates as shown in Fig. 4.

We analyzed the performance of feature encoder and trajectory association together on Raspberry pi at normal CPU clock and also at overclocked CPU. We found that the operation took 109ms for a single detection. As the feature encoding and trajectory association are scaled with the number of persons detected in the surveillance scene, the current settings of the operation are not suitable for real-time processing of the surveillance scene with multiple people. Most of the processing time out of 109ms is consumed in the feature encoder which is a CNN trained on MARS dataset [33] with an image input size of 128x64. We retrained it with an image input size of 64x32 and got a speed improvement of about 62%. With this, the feature encoding and trajectory association took 45ms and even improved by overclocking to 42ms per detection. This setting is considerably faster and practical for computer vision applications running on CPU-only edge devices, as it provides sufficient speed for anomaly detection in residential video surveillance.

Setting hardware for the optimum performance, i.e. CPU clock at 1900MHz and feature encoder with input image size 64x32, the overall execution time can be estimated as;

(a) Object Detection (OD) in the surveillance frame takes 92ms.
(b) Feature Encoding (FE) per detected object takes 38ms.
(c) Trajectory Association (TA) takes 4ms.

Thus, the overall execution time $\tau$ considering $d_k$ bounding box detections in the surveillance scene is given as,

$$\tau = 96 + 38 \times d_k \tag{5}$$

With this setting, we can detect anomalies with eight detections in the surveillance scene maintaining 2.5 FPS, five detections with 3.5 FPS, two detections with 6 FPS, and one detection with 8 FPS.

Hardware accelerators like, Movius Neural Compute Stick (MNS) are very useful to achieve higher processing speed for real-time applications. We used Movius Neural Compute Stick V1 which is powered by Myriad Vision Processing Unit (VPU) and an AI-optimized chip for accelerating vision computing based on Convolutional Neural Networks (CNN). Integration of MNS with Raspberry pi 4B provides considerable improvements in the results as shown in Table 2. The hardware accelerator (MNS) boosts the overall performance by about 20% and facilitates detection of anomalies with a sufficient frame rate for residential surveillance applications.

V. RESULT AND DISCUSSION

The experimental setup accepts live camera feed as input to the algorithm. However, to test the algorithm using standard datasets, we set up a system to accept pre-recorded video sequences instead of the live camera feed. We used UCSD Ped1 [34] and UMN [35] dataset to test a proposed system both qualitatively and quantitatively.

In general, the regularity score whose threshold is manually specified, defines whether the input frame is normal or abnormal. The optimal value of this parameter is very important as higher threshold increases false negative rate, while a lower threshold increases false positive. Thus, we use the Area Under Curve (AUC) which is a more suitable metric [36], [37]. It measures the performance by changing different thresholds. We adopt frame-level comparison that predicts which frames contain anomalous events. This is compared to the frame-level ground-truth anomaly annotations to determine the number of true- and false-positive frames. Following same protocol, the proposed system is tested extensively and the obtained AUC for the set of experiment is given in Table-3.

We tested several parameter configurations; (a) the ‘non-maximum suppression’ (nms-overlap) threshold for object detector to avoid spurious overlapping detection boxes, (b) for feature encoder, we
TABLE III: Anomaly detection scores at different parameter configuration

| Feature Encoder | max-cos -distance | nms -overlap | AUC Score  |
|-----------------|-------------------|-------------|------------|
| 128x64          | 0.6               | 0.3         | 0.892      |
| 128x64          | 0.6               | 0.6         | 0.921      |
| 128x64          | 0.6               | 0.9         | 0.873      |
| 128x64          | 0.9               | 0.3         | 0.925      |
| 128x64          | 0.9               | 0.6         | 0.848      |
| 128x64          | 0.9               | 0.9         | 0.802      |
| 64x32           | 0.6               | 0.3         | 0.885      |
| 64x32           | 0.6               | 0.6         | 0.913      |
| 64x32           | 0.6               | 0.9         | 0.838      |
| 64x32           | 0.9               | 0.3         | 0.918      |
| 64x32           | 0.9               | 0.6         | 0.851      |
| 64x32           | 0.9               | 0.9         | 0.835      |

| Feature Encoder | max-cos -distance | nms -overlap | AUC Score  |
|-----------------|-------------------|-------------|------------|
| 128x64          | 0.6               | 0.3         | 0.923      |
| 128x64          | 0.6               | 0.6         | 0.895      |
| 128x64          | 0.6               | 0.9         | 0.931      |
| 128x64          | 0.9               | 0.3         | 0.927      |
| 128x64          | 0.9               | 0.6         | 0.891      |
| 128x64          | 0.9               | 0.9         | 0.886      |
| 64x32           | 0.6               | 0.3         | 0.931      |
| 64x32           | 0.6               | 0.6         | 0.917      |
| 64x32           | 0.6               | 0.9         | 0.891      |
| 64x32           | 0.9               | 0.3         | 0.936      |
| 64x32           | 0.9               | 0.6         | 0.872      |
| 64x32           | 0.9               | 0.9         | 0.882      |

TABLE IV: Comparative results of anomaly detection on CPU-only edge device

| Methods        | UCSD Ped-1 Dataset | UMN Dataset |
|----------------|-------------------|-------------|
| [6]            | 0.781             | 0.816       |
| [38]           | 0.863             | 0.872       |
| [4]            | 0.890             | 0.912       |
| [39]           | 0.772             | 0.785       |
| Proposed       | **0.918**         | **0.936**   |

tested size of feature encoder 64x32, 128x64, and 256x128 and (c) the max-cos-distance threshold for trajectory association.

It is observed that, reducing the size of input feature encoder does not affect anomaly score significantly but reduces the processing time drastically. We also observed, lower non-maximum-suppression effectively suppress some spurious boxes that are generated by clusters of people and achieves slightly improved performance.

Further, for trajectory associations, setting the max-cos-distance threshold to higher value achieves better performance. Lower values of the threshold may result into unwanted identity swaps as the cosine distances between two encoded feature vectors are not large enough to differentiate people wearing similar color clothing.

Based on the results obtained for different parameter configurations, we set the best configuration for the proposed system as feature encoder size to 64x32, max-cos-distance to 0.9, and nms-overlap threshold to 0.3. With this optimum configuration, we analyzed the power consumption for running and sleeping states at different CPU clock rates as shown in Fig. 5. We also noticed that the core temperature spikes up to about 68°C without active cooling and about 59°C with active cooling. The effect of overclocking on CPU cores is shown in Fig. 6. However, without active cooling, further increase in the CPU clock, increases the core temperature and halts the system due to overheating.

Finally, with the optimum setting, we compare the proposed approach with existing methods on UCSD Ped-1 and UMN dataset as shown in Table-4. Images capturing anomalies in the surveillance scenes are shown in Fig. 7. Results demonstrate that the proposed method outperforms the other approaches by achieving higher scores. The spatio-temporal feature encoding effectively works in associating trajectories and ensures consistency in person identities even in complex scenes, deformed postures, and partial occlusions. Furthermore, the improvement in time efficiency, though not suitable for large-scale surveillance environments, is considerable for facilitating anomaly detection using CPU-only edge devices in residential video surveillance systems.

VI. CONCLUSION

In this work, to make the intelligent surveillance system highly portable and cost effective, we used CPU-only edge devices to detect anomalies in residential video surveillance. The associated trajectories and dynamics in them are utilized to detect abnormal behavior of people in surveillance scenes. The trajectory association corresponding to each detection is performed by integrating spatial and temporal information. The spatio-temporal integration is governed by two metrics that provide promising results in cluttered...
environment. The alerts for the anomalies are sent through LoRa
edge devices. Moreover, we also establish that the use of hardware
achieve optimized processing time and FPS by slightly compromising
Through several experiments, we optimized the feature encoder to
environment. The alerts for the anomalies are sent through LoRa

\[16\] N. Priyadharsini and D. Chitra, “A kernel support vector machine

\[19\] E. Gundogdu and A. A. Alatan, “Good features to correlate for

\[17\] S. Chiappino, P. Morerio, L.Marcenaro, and C. S. Regazzoni, “Bio-

\[15\] S. D. Bansod and A. V. Nandedkar, “Crowd anomaly detection and

\[13\] N. Dalal and B. Triggs, “Histograms of oriented gradients for human

\[12\] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,”

\[11\] S. K. Kumaran, D. P. Dogra, and P. P. Roy, “Anomaly detection

\[10\] A. A. Sodemann, M. P. Ross, and B. J. Borghetti, “A review of anomaly

\[9\] P. V. K. Borges, N. Conci, and A. Cavallaro, “Video-based human

\[8\] S. Teerapittayanon, B. McDanel, and H. T. Kung, “Distributed deep

\[7\] D. R. Patrikar and M. R. Parate, “Anomaly detection using edge computing

\[6\] R. Xu, S. Y. Nikouei, Y. Chen, A. Polunchenko, S. Song, C. Deng, and T. R. Faughnan, “Real-time human objects tracking for smart surveillance at the edge,” in 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

\[5\] Y. Liu, Y. Chen, H. Yu, X. Fang, and C. Gong, “Real-time expert system for anomaly detection of aers based on computer vision technology and existing surveillance cameras,” CoRR, vol. abs/1810.04108, 2018. [Online]. Available: http://arxiv.org/abs/1810.04108

\[4\] R. Navaratne, D. Alahakoon, D. De Silva, and X. Yu, “Spatiotemporal anomaly detection using deep learning for real-time video surveillance,” IEEE Transactions on Industrial Informatics, vol. 16, no. 1, pp. 393–402, 2019.

\[3\] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh, “Anomalynet: An anomaly detection network for video surveillance,” IEEE Transactions on Information Forensics and Security, vol. 14, no. 10, pp. 2537–2550, 2019.

\[2\] A. Ghosh and K. Grolierger, “Edge-cloud computing for iot data ana-
lytics: Embedding intelligence in the edge with deep learning,” IEEE Transactions on Industrial Informatics, 2020.

\[1\] S. K. Kumaran, D. P. Dogra, and P. P. Roy, “Anomaly detection in road traffic using visual surveillance: A survey,” arXiv preprint arXiv:1901.08292, 2019.