A deep learning approach to trespassing detection using video surveillance data

Muzammil Bashir
Department of Computer Science
Worcester Polytechnic Institute
United States
mbashir@wpi.edu

Elke A. Rundensteiner
Department of Computer Science
Worcester Polytechnic Institute
United States
rundenst@wpi.edu

Ramoza Ahsan
Department of Computer Science
Worcester Polytechnic Institute
United States
rahsan@wpi.edu

***Abstract*—Railroad trespassing is a dangerous activity with significant security and safety risks. However, regular patrolling of potential trespassing sites is infeasible due to exceedingly high resource demands and personnel costs. This raises the need to design automated trespass detection and early warning prediction techniques leveraging state-of-the-art machine learning. To meet this need, we propose a novel framework for Automated Railroad Trespassing detection System using video surveillance data called ARTS. As the core of our solution, we adopt a CNN-based deep learning architecture capable of video processing. However, these deep learning-based methods, while effective, are known to be computationally expensive and time consuming, especially when applied to a large volume of surveillance data. Leveraging the sparsity of railroad trespassing activity, ARTS corresponds to a dual-stage deep learning architecture composed of an inexpensive pre-filtering stage for activity detection, followed by a high fidelity trespass classification stage employing deep neural network. The resulting dual-stage ARTS architecture represents a flexible solution capable of trading-off accuracy with computational time. We demonstrate the efficacy of our approach on public domain surveillance data achieving 0.87 f_1 score while keeping up with the enormous video volume, achieving a practical time and accuracy trade-off.**

***Index Terms*—Trespassing detection, Railroad security, Deep learning, Video surveillance, Deep Convolutional Neural Networks, Background subtraction, Computer vision**

I. Introduction

Railroad trespassing is a widely discussed problem in railroad security. From 2006 to 2015, 2717 deaths and 9595 injuries have been reported to be a direct result of trespassing activity in United States [28]. This amounts to 3.37 casualties every day. According to Federal Railroad Administration (FRA), there are around 210,000 railway crossings in USA and around 61% of them are exposed to potential trespassing activity [28]. Each of these sites pose a risk to both trespassers as well as trains and their passengers and cargo. In most cases, collision with a train proves to be fatal for the trespasser. Aside from these tragic human costs, these accidents, whether fatal or not, are exceedingly expensive. Property damage, emergency services, safety investigations, insurance, legal and delay costs account for hundreds of thousands to millions of dollars per accident [5].

Although railroad trespassing related accidents have been shown to be the leading cause of fatality [2], [11], [13], [16], it remains an under-researched area [10]. One simple solution would be to set up a surveillance network of CCTV cameras and employ human analysts to review the video feed on a 24×7 basis. This could be useful for determining locations and times for dangerous trespassing incidents with the ultimate objective to develop more efficient resource utilization i.e. deployment of limited personnel (from police officers to social workers) to potential trespassing sites only on a need basis. However, a major limitation of this solution is the overwhelming demand this would impose on trained human analysts. These analysts have to review tens of hours of CCTV data from hundreds of cameras, making it a tedious and time-consuming process. Manual processing of this "big data" is simply infeasible and non-practical. Further, human analysis has additional drawback of subjectivity and unreliability due to the dull and mundane nature of the task [14].

Due to the above mentioned reasons, bringing automation and artificial intelligence to tackle this trespassing prevention challenge is of vital importance. Trespassing detection serves as the first step towards any future automated AI-based trespassing prevention solution. A reliable automated trespassing detection system would not only provide detection in a timely manner but may also allow us to develop advanced analytics for trespassing patterns over time. For example, analysis over a period of three months may reveal that a group of youth likes to play football during a certain evening time near a section of the track. Certain locations might see increased trespassing during the morning or evening times with people taking a short-cut when returning home from jobs. Other locations such as underpasses and bridges may provide a preferred meeting location for drug addicts. Use of "big data analytics" can help us to ultimately make better predictions and thus assist in reducing trespassing activity substantially.

A. Goals of this research

Given a surveillance video, the problem of trespassing detection is to decide whether each given frame has human trespassing activity or not. We define a trespasser as a human within the camera field of view. We not only want to predict the occurrence of trespassers but we also want to do so in a time and resource efficient manner. We observe that a railroad surveillance video is extremely sparse in terms of trespassing

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

activity, i.e., in a given 24 hours of railroad surveillance video, most of the video shows no trespassing activity. We thus propose to leverage this property of sparseness to reduce the processing time. Further, we postulate that the detection accuracy[1] and speed of detection are two conflicting goals. Generally, if one wishes to improve the speed of detection, they will have to sacrifice accuracy and vice versa. Therefore, we are interested in developing a flexible solution that is capable of trading-off accuracy with computational time.

B. State-of-the-art

Despite the obvious need, there has been limited activity among the research community towards solving this problem. Recently, Zhang et. al. [28] proposed a technique focusing on detecting the near-misses during railroad trespassing. However, their proposed methodology makes the assumption that the train always constitutes the majority of the moving pixels. Their strategy may thus fail if the camera is located away from the train. Salmane et. al. [22] proposed a technique for detecting hazard situations at railway crossings. However, their technique only detects the movement and does not discriminate moving objects into train, vehicle or person as would be required for trespassing detection. Another shortcoming common to these prior two methods is that they do not leverage advanced deep learning methods. Recent research in computer vision has shown that Convolutional Neural Network (CNN) based deep neural network architectures are the model of choice for image and video analysis [7].

C. Approach

Our framework **ARTS** (Automated Railroad Trespassing detection System) solves this problem of automated trespassing detection by adopting a two-step approach. The first stage is responsible for filtering out frames that show little to no activity, this way reducing the amount of data to be processed by the later extremely compute-intensive stage. The second stage adopts a state-of-the-art deep learning model based on CNN to ensure effective detection of trespassing activity. To realize the impact of this approach, consider a scenario in which we have 1 hour of surveillance video (at 10 frames per second). If we were to use a single staged state-of-the-art solution that detects trespassing at every frame, we would need to spend approximately 5 hours. Whereas our proposed ARTS approach only takes 0.8 hours (48 minutes) even when a rather high ratio of activity[2] (10%) were to be present in the above said surveillance video.

D. Contributions

Next we enumerate the key contributions of this work:

- We propose a flexible trespassing detection framework called **ARTS** that can trade-off speed and accuracy by leveraging the observed property of activity sparseness. (Sec. IV-B)
- Our proposed ARTS framework adopts a *plug and play* design to allow embedding any algorithm suitable to individual stages of our framework. (Sec. III-B)
- Our solution combines an inexpensive yet effective traditional computer vision approach with a state-of-the-art deep learning architecture to develop an overall robust technology. (Sec. III-C and III-D)
- We conduct an in-depth experimental evaluation of the proposed **ARTS** approach considering a variety of experiments to demonstrate relative trade-off and the overall effectiveness of ARTS.[3] (Sec. IV)

E. Organization of paper

The remainder of this paper is organized as follows. Section II discusses the related work. Section III explains the technical details of proposed approach while Section IV discusses the experimental evaluation. Section V concludes the paper and discusses future directions.

II. Related Work

A. Computer vision in trespassing detection

Little prior work has been done to use computer vision for railroad trespassing safety. Shah et al. [24] employed color-based background subtraction to detect moving objects. They use color, motion and size-based features to track those objects. Tracked objects are classified into people, a group of people or a vehicle. Salmane et al. [22] proposed a multi-stage system that uses frame-based background subtraction for moving object detection. However, moving objects are not discriminated into train, vehicle or persons as would be required for trespassing detection. Further, as noticed by Zhang et al. [28], even though the system is expected to run in real time, no information regarding the speed of the algorithm is reported. Zhang et al. [28] developed a near-miss trespassing detection system. Their focus is on the detection of near-miss events on gated crossings rather than general trespassing detection. They first determine the time interval during which the gates are closed and then detect moving objects using background subtraction. However, they use the number of pixels as classification metric to distinguish between the train and other objects (people and vehicles). This strategy works for their test data but may fail if the camera is located away from train, since it is based on the simple assumption that the train will always constitute the majority of the moving pixels.

One drawback common to all these prior approaches is that they use simple background subtraction method (based on subtracting mean or median pixel values) for detecting moving objects. This method is known to generate noisy output [26]. As one exception, Shah et al. [24] use hand-crafted features to classify the object type. However, recent research suggests that deep-learning based approaches outperform these hand-crafted feature-based approaches [1]. These limitations in previous works motivates us to explore a deep learning based trespassing detection solution that can reliably detect human trespassing activity.

[1]Experimental evaluation uses $f1$ score and AUC as concrete evaluation metrics.

[2]Activity ratio is precisely defined in Sec. IV-A1

[3]The code shall be released to the research community after publication.

B. Deep learning based object detection

Since the success of Alexnet [7] in the 2011 ILSVRC [21] challenge, a considerable effort has been put in by the computer vision community to explore deep learning methods. Overfeat [23] was among the first attempts in improving object detection accuracy using deep learning. However, it was extremely slow as it considered all sliding window positions. Instead of using sliding windows, Girshick et al. [4] proposed to use a RoI (Region of Interest) based approach using selective search [27]. These RoIs (far less in number than sliding windows) are then passed through the same pipeline as Overfeat. Though, it resulted in significant reduction of time; this solution still applied the expensive convolution operation on each RoI independently. Most RoIs are overlapping and thus computational resources are wasted by the re-computation of overlapping RoIs. Fast-RCNN [3] overcomes this shortcoming by sharing the convolutional features. This technique makes Fast-RCNN 9x faster than RCNN [4] in training and 213x faster in inference [3]. Sharing of features speeds up the computation to such a degree that it renders selective search [27], a non-deep learning process, the bottleneck in the object detection pipeline. In response, Ren et. al. proposed Faster-RCNN [20] which replaced the conventional selective search process with RPN (Region Proposal Network) as subset of the overall neural network architecture. RPN, being part of the neural network, is much faster than selective search. Both Fast-RCNN and Faster-RCNN are two-step object detectors in that they propose regions in first step and then proceed to classify them in second step.

As opposed to two-step detectors, single step detectors such as YOLO [18], [19] and SSD [9] have also recently been proposed. These solutions merge the region proposal process and classification process into a single unified deep pipeline leading to faster response times. However, in terms of accuracy, they lack behind two-step detectors since the number of region proposals considered by single-step detectors are far less than those considered by two-step detectors [19].

III. Methodology

A. Problem formulation

Given an input surveillance video containing N frames, we want to produce a binary time series of the same length N such that at each index i, we have the label y_i of the corresponding frame f_i. The human trespassing label is assigned to the positive class (1) while the "other activity" label is assigned to the negative class (0). Since each prediction depends only on the corresponding frame f_i, our problem corresponds to determining a function D with parameter θ such that:

$$D(f_i;\theta) = \hat{y}_i$$

The aim is to find a θ^* such that $D(f_i;\theta^*) \rightarrow y_i$ where y_i is the ground truth label corresponding to f_i. The ground truth label has the following definition:

$$y_i = \begin{cases} 1, & \text{if } f_i \text{ contains trespassing activity} \\ 0, & \text{otherwise} \end{cases}$$

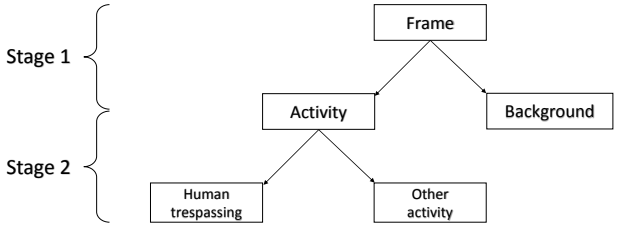

Fig. 1. Trespassing detection framework

Here, we define trespassing activity as the presence of at least one person in the frame.

B. ARTS framework

Figure 1 shows the outline of our proposed ARTS framework. In the first stage, we decide whether a particular frame has an activity or not. If it turns out that the given frame has no activity, then it is classified as background frame. No further action needs to be taken for this frame. On the other hand, if it shows activity, then the next step (stage 2) is to investigate whether it can be classified as human trespassing activity or not. The rationale for using a dual-stage architecture is that the first stage is much faster than the second stage. This allows the first stage to filter out most of the non-activity frames and therefore only prospective frames are processed by stage 2.

Figure 2 illustrates the pipeline modeling the ARTS framework. Input to our pipeline is a video with each frame being processed one by one. Only the frames classified as "activity frames" are further processed by stage 2. Output of ARTS corresponds to a time series of predicted labels.

Input to stage 1 is a video frame, whereas the output is a binary decision indicating whether it is an activity frame or background frame. Stage 2 not only gets the output of stage 1 as input, but also the corresponding frame. It skips the processing of the frame in case it has been classified by stage 1 as background type; otherwise, it processes the frame for verification of the human trespassing activity. The output of stage 2 is a binary decision indicating human trespassing or other activity. Concatenation of the stage 2 output for each input frame produces the output time series. The proposed ARTS framework is flexible in the sense that it allows any pair of algorithms to be plugged into the trespassing detection pipeline given they perform the functions of stage 1 and stage 2 respectively. Also, the processing time per frame of stage 1 algorithm should be considerably less than that of stage 2 algorithm to reap the practical benefit of ARTS.

C. Stage 1 of the ARTS framework

The goal of this first stage is to filter non-activity frames from activity frames. Thus, this is modeled as a background subtraction problem. Figure 3 shows the block diagram of background subtraction method.

We propose to model the background as mixture of gaussians [17], [26]. Since, an image usually represents many different surfaces (objects), each surface is expected to give

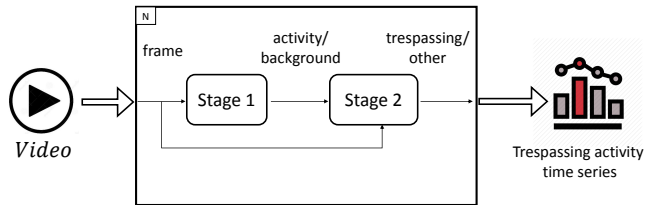

Fig. 2. Trespassing detection pipeline

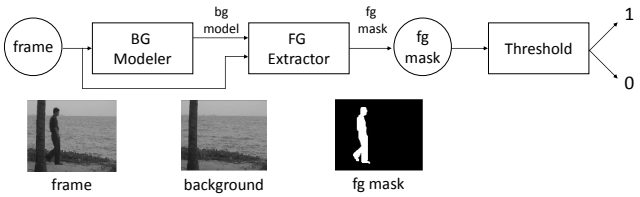

Fig. 3. Stage 1 - Background subtraction model

rise to a new gaussian. Thus all pixel values are overall represented by a mixture (sum) of gaussians. Notice that this model represents both foreground and background simultaneously. In order to apply this model to the background subtraction problem, we associate each pixel with a particular surface and then associate that surface as a whole with either foreground or background type. The label of each pixel, namely either foreground or background, is determined by the label of its corresponding surface. Table I shows the notation used in stage 1.

1) Background (BG) modeler: Each surface (or uniform object) that comes into the camera view is represented by a state $k \in 1, 2, 3, ..., K$. Some of these states correspond to background while the remaining ones are considered to be foreground. The process $\mathbf{k}$ which generates the states is modeled by parameters set $\{w_1, w_2, ..., w_K\}$ where $w_k = P(k)$. Each of these parameters represents a priori probability of surface k appearing in the image. Further, we have $\sum_{k=1}^{K} w_k = 1$.

This surface process $\mathbf{k}$ is hidden and is only indirectly observable through the pixel value process $\mathbf{X}$. The pixel value process $\mathbf{X}$ is an observable random variable modeled by a gaussian process for given surface k. $\mathbf{X}$ is 1-D in case of gray scale images and 3-D for color images. If $\theta_k = \{\mu_k, \Sigma_k\}$ represents the associated gaussian process, then the pixel value process $\mathbf{X}$ given k is:

$$f_{\mathbf{X}|k}(X|k, \theta_k) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_k|}} e^{-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1} (X-\mu_k)}$$

where μ_k denotes the mean and Σ_k the covariance matrix of the associated k^{th} gaussian.

We assume these k events are disjoint, so $\mathbf{X}$ can be modeled as the sum of gaussians.

$$f_{\mathbf{X}}(X|\Phi) = \sum_{k=1}^{K} w_k f_{\mathbf{X}|k}(X|k, \theta_k)$$

TABLE I
STAGE 1 NOTATION

notation	description
k	state representing a surface in image
K	total number of target surfaces in image
w_k	prior probability of surface k
$P(k)$	w_k
X	pixel value
$\mathbf{X}$	random variable modeling pixel values
μ_k	mean pixel value of surface k
Σ_k	covariance matrix of pixel value of surface k
θ_k	$\{\mu_k, \Sigma_k\}$
Φ	$\{w_1, \mu_1, \Sigma_1, ..., w_K, \mu_K, \Sigma_K\}$
$f_{\mathbf{X}\|k}(X\|k, \Phi)$	prob. of pixel value process $\mathbf{X}$ given k
$P(k\|X, \Phi)$	posterior probability

where $\Phi = \{w_1, \mu_1, \Sigma_1, ..., w_K, \mu_K, \Sigma_K\}$

2) Foreground (FG) modeler: To apply the model to our background subtraction problem, we must determine which of the K states is most likely to give rise to the current pixel value $\mathbf{X} = X$. The posterior probability $P(k|X, \Phi)$ denotes the likelihood that pixel value X was generated by surface k. Using the Bayes's theorem, we have:

$$P(k|X, \Phi) = \frac{P(k) f_{\mathbf{X}|k}(X|k, \Phi)}{f_{\mathbf{X}}(X, \Phi)}$$

The k which maximizes the $P(k|X, \Phi)$ is then determined to be the surface associated with X:

$$\hat{k} = \underset{k}{\operatorname{argmax}} P(k|X, \Phi)$$

Once X has been associated with a particular surface $\hat{k}$, it needs to be determined whether $\hat{k}$ is a foreground or background surface.

The procedure for demarcation starts with ranking K states by $w_k/|\Sigma_k|$ in the decreasing order. This ratio is proportional to the height of the weighted distribution $w_k f_{\mathbf{X}|k}(X|k, \theta_k)$. A surface k is considered to be a background, if it occurs more frequently (higher w_k) and does not vary much (low $|\Sigma_k|$). To separate the foreground and background surfaces, an overall prior probability T of an object in the background is used. The first B of the ranked states whose accumulated probability crosses the threshold T are considered to be background.

$$B = \underset{b}{\operatorname{argmin}} (\sum_{j=1}^{b} w_j > T)$$

3) Threshold: The output of the foreground extractor is a binary mask which indicates whether a pixel belongs to the foreground or not. All foreground pixels in the image are summed up and their ratio to the total number of pixels in the frame is compared to a threshold value τ. If the ratio is greater than threshold τ, then this frame is classified as activity frame (1); otherwise as background frame (0).

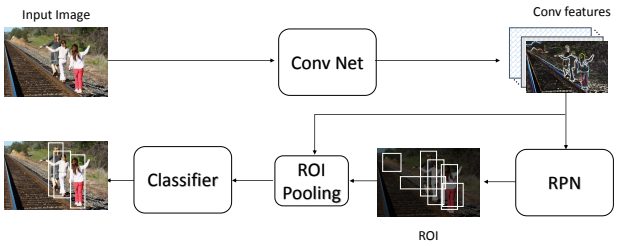

Fig. 4. Stage 2 - Faster-RCNN block diagram

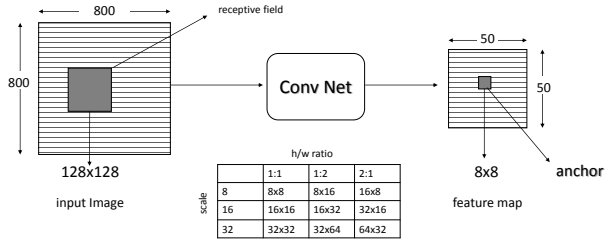

Fig. 5. Anchor illustration

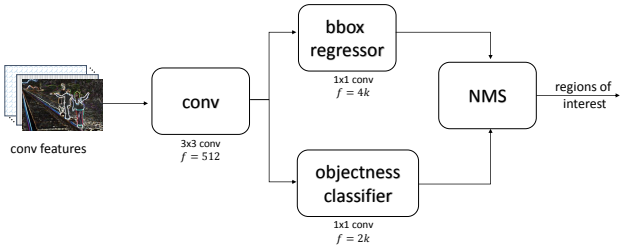

Fig. 6. Region Proposal Network architecture

D. Stage 2 of the ARTS framework

The goal of this second stage is to verify human trespassing in the case of activity reported by stage 1. The state-of-the-art deep learning model Faster-RCNN [20] is used to model this stage. Faster-RCNN predicts the label and location of the objects corresponding to the input frame. The image is labelled with *trespassing activity* if there is at least one person predicted by Faster-RCNN with a confidence score greater than a threshold ν. Figure 4 depicts the block diagram of Faster-RCNN with a detailed explanation given below:

1) Feature extraction (Conv. Net): The first step is to extract the convolutional features, also known as feature map, from the input frame. These convolutional features will be used by subsequent sub-stages as input. Due to the importance of features generated by this sub-stage, it is also known as the backbone of Faster-RCNN. This process can be modeled using VGG [25] or Resnet [6]. Here in this work, we adopt the Resnet-50 network with Feature Pyramidal Network (FPN) [8] for our experiments.

2) Region Proposal Network (RPN): This sub-stage as the name suggests is responsible for proposing regions (rectangles) potentially containing objects (people, cars, etc). As seen in Figure 4, this sub-stage takes in the feature map and produces a list of proposals for the given image. Each proposal consists of a binary label and the proposed bounding box of the region of interest. The label indicates whether the proposal corresponds to an object or background.

a. Anchors of RPN

An anchor acts as default region proposal. This idea has been motivated by multi-scale sliding windows [20]. Suppose we use a feature extraction convolutional network such that it converts a 800×800 image to 50×50 feature map (Figure 5). This means every (x, y) location on the feature map corresponds to a 16×16 window on the original image. Similarly, an 8×8 window on the feature map corresponds to a 128×128 window on original image. This 8×8 window on the feature map is known as anchor. Faster-RCNN proposes multi-scale, multi-aspect ratio anchors. A total of 3 scales (8, 16, 32 on the feature map) with 3 aspect ratios $(1:1, 1:2, 2:1)$ produce 9 anchors on each (x, y) location of the feature map. Since we have 50×50 locations, this setting produces 22,500 anchors in total. However, in practice we use far less than that number. All anchors whose regions lie outside the feature map (e.g., anchors near edges) don't participate in training the network.

b. Architecture of RPN

Figure 6 shows the architecture of RPN sub-network. Input to this network are the features generated by the backbone network discussed in Section III-D1. These features are passed through a 3×3 convolution layer. Faster-RCNN uses 512 output feature depth for this layer. Output of this layer is fed to the bounding box regressor layer and objectness layer which predict bounding box locations and objectness score simultaneously. Both of these layers are modeled with a 1×1 convolution. Bounding box regressor layer has $4k$ output depth with k the number of anchors and 4 indicating that each proposal is defined by 4 scalar values. For similar reasons, objectness layer has $2k$ output features. Each anchor produces such a proposal. All proposals are post-processed by the Non Maximum Suppression (NMS) algorithm.

3) Non maximum suppression (NMS): NMS is responsible for removing the duplicate predictions. Figure 7 illustrates the goal of this process graphically. To suppress the duplicate proposal predictions with the less confidence, the first step is to sort all proposals in descending order. The first proposal is made the reference proposal and pushed to the "keep" list. Intersection over Union (IoU) of this reference proposal with all remaining proposals is computed. The proposals which sufficiently overlap with the reference proposal ($IoU > 0.7$) are discarded as they are considered to be the duplicate of the reference proposal. In the next iteration, the first proposal in the list of undecided proposals is made reference proposal and the above process is repeated. Again this leads to the removal of all proposals considered to be duplicate of the reference proposal. This process continues until all proposals are labeled as either kept or discarded. Output of this process is the list

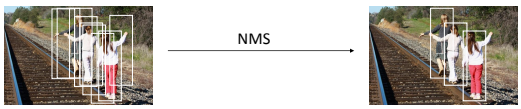

Fig. 7. Non Maximum Suppression (NMS). Highly overlapping predictions with lesser confidence are suppressed.

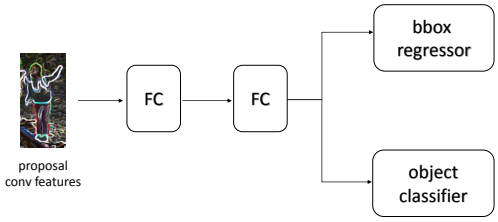

Fig. 8. Fast-RCNN classifier

of "kept" proposals.

4) Fast-RCNN head: For the proposals from RPN, we need to predict the corresponding objects' labels and locations. Faster-RCNN uses Fast-RCNN head [3] for this purpose. Fast-RCNN head has two further sub-components: the Region of Interest (RoI) pooling and the classifier layer (Figure 4). Input to the Fast-RCNN head will be the feature map and the list of kept proposals; and output shall be improved bounding box locations of the corresponding proposals along with their class labels.

Different proposals have different feature map sizes. However, the classifier expects them to be of the same size. RoI pooling is responsible for converting variable sized feature maps into fixed sized. The methodology used by Fast-RCNN in this case is simple. Suppose a feature map of size 8×8 has to be converted to 2×2 size. Then, a grid of size 2×2 is placed on top of the feature map such that its boundaries align with the feature map. The maximum feature value from each grid cell is copied to corresponding cell in the output buffer to convert a 8×8 feature map to a size of 2×2.

Once RoI pooling has adjusted the size of feature map to fixed dimensions, the classifier takes in those features and passes them through two fully connected layers. The output of those two layers is fed to two separate fully connected layers responsible for predicting bounding boxes and object class labels. The bounding boxes and labels so predicted are the final output of Faster-RCNN. Figure 8 illustrates the architecture of classifier.

5) Training loss: While training Faster-RCNN, we train two sub-networks: RPN and classifier. Both networks have two objectives: label classification and bounding box regression. Next, we sketch the two equations that model the RPN and classifier loss. The first term corresponds to the label classification while the second term corresponds to the bounding box regression.

TABLE II
DESCRIPTION OF VARIABLES USED IN LOSS FUNCTION

variable	description
$\hat{p}_i$	anchor label prediction
$\hat{t}_i$	anchor bounding box prediction
$\hat{q}_i$	RoI label prediction
$\hat{u}_i$	RoI bounding box prediction
p_i	anchor ground truth label
t_i	anchor ground truth bounding box
q_i	RoI ground truth label
u_i	RoI ground truth bounding box
λ_r	RPN loss balance coef.
λ_c	classifier loss balance coef.
N_{cls}	mini-batch size
N_{reg}	total anchors

$$Loss_{RPN} = \frac{1}{N_{cls}} \sum_i L_{cls}(\hat{p}_i, p_i) + \frac{\lambda_r}{N_{reg}} \sum_i p_i L_{reg}(\hat{t}_i, t_i)$$

$$Loss_{cla} = \frac{1}{N_{cls}} \sum_i L_{cls}(\hat{q}_i, q_i) + \frac{\lambda_c}{N_{reg}} \sum_i [q_i > 0] L_{reg}(\hat{u}_i, u_i)$$

Table II defines the variables used in the above equations. Classification loss L_{cls} denotes the standard log loss, and regression loss L_{reg} is the smooth-L_1 loss as defined below.

$$L_{cls}(\hat{y}, y) = -\sum_j y_j log(\hat{y}_j)$$

$$L_{reg}(\hat{b}, b) = \sum_{j=1}^{4} SL_1(\hat{b}_j - b_j)$$

$$SL_1(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ x - 0.5, & \text{otherwise} \end{cases}$$

where input parameters to each function carry the standard meaning. The total loss of the network corresponds to the sum of RPN loss and classifier loss.

$$Loss = Loss_{RPN} + Loss_{cla}$$

The term p_i in regression term of RPN loss makes sure that regression loss is activated only if proposal corresponds to an object, not to background. Thus bounding box predictions corresponding to background proposals do not contribute towards training. The expression $[q_i > 0]$ serves a similar job in the classifier loss. This again acts as a flag to add regression loss only corresponding to the actual objects and not the background.

λ_r and λ_c serve as balancing parameters between label classification and bounding box regression. The authors of Faster-RCNN state that λ_r is redundant and Faster-RCNN remains insensitive to a large range of λ_r [20].

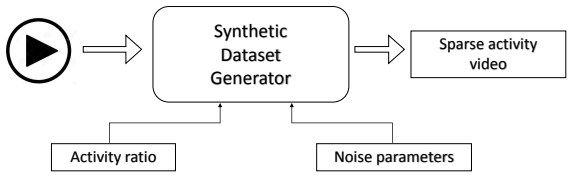

Fig. 9. Synthetic dataset generator

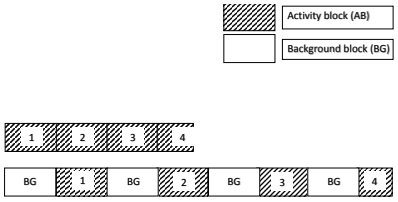

Fig. 10. Synthetic video illustration

IV. Experimental evaluation

Experimental setup: All our experiments have been carried out using single Nvidia Tesla K20 GPU. We use pytorch based Faster-RCNN implementation[4] along OpenCV v4.0.0 running on Red Hat Linux based HPC cluster.

A. Dataset

The dataset used for experimental evaluation is VIRAT 2.0 [15]. This video dataset has been developed for activity classification and is with human activity; meaning frames mostly contain humans in motion. In contrast, trespassing data is known to be extremely sparse because trespassing is an anomaly instead of common steady traffic pattern. To model this, we augment the VIRAT 2.0 dataset to control the amount of activity to background ratio as described below.

1) Synthetic dataset generator: Figure 9 illustrates our synthetic dataset generator which adopts the following steps:

1) identify background frame
2) make background block(s)
3) write background block(s)
4) write activity block(s)
5) repeat (3) and (4)

Step 1 is human labor intensive but simple. We manually scroll through a video and identify a variety of frames with no human. We call these frames, the background frames. In step 2, we repeat each background frame $\Delta \times fps$ times to generate the corresponding background block. Here, the fps denotes the frames-per-second of the original video and Δ indicates the target length of background blocks in seconds. In step 3, we randomly select a certain number of background blocks (determined in step 2) and write them to output video stream. The exact number of background blocks written depends on activity ratio, discussed at the end of this subsection. Step 4 writes the activity block in output video stream. An activity block is simply a section of the original video. By default, we use $30s$ long activity blocks. The length of each background block is also kept at $30s$ by default. Figure 10 explains the concept of activity block and synthetic video.

An important metric associated with this procedure is *Activity Ratio*. It captures the fraction of activity present in the output video. It is defined as:

$$\text{Activity Ratio} = \text{AR} = \frac{1}{1 + nBG}$$

[4]https://github.com/facebookresearch/maskrcnn-benchmark

TABLE III
Relationship of number of background blocks and activity ratio

nBG	nAB:nBG	AR
1	1:1	0.5
3	1:3	0.25
5	1:5	0.16

where nBG = number of background blocks per activity block. Figure 10 has $nBG = 1$. Table III explains the relationship between nBG and AR.

2) Adding noise: Outdoor surveillance videos are often subjected to environmental challenges such as rain and snow. In order to test the robustness of our approach in those challenging situations, we add noise to our synthetic data. Salt and pepper noise [12] is known for modeling rain and snow effect, therefore we use it in our experiments. Table IV describes the parameters that control the level of noise.

To add noise, we first select $p\%$ of frames from each background block. Each frame is equally likely to be selected. Now we draw an integer r from the normal distribution with parameters μ and σ. For each selected frame, we select r pixels and add noise to them. Again all pixels are equally likely.

Experimental methodology: To validate our approach, we carry out a comprehensive experimental evaluation study. We study the proposed ARTS approach from three different perspectives:

1) time-accuracy trade-off: to investigate the end-to-end accuracy of the pipeline while trading-off computational time
2) stage-wise analysis: to helps us understand which stage represents the potential bottleneck in terms of accuracy
3) noise analysis: to understand the robustness of our approach to noise.

TABLE IV
Noise parameters

params	description
p	percentage of noisy frames in BG block
μ	avg. number of noisy pixels in noisy frame
σ	standard deviation of number of noisy pixels

TABLE V
SYNTHETIC DATA PARAMETERS FOR TIME-ACCURACY TRADE-OFF

parameter	value
p	1%
μ	0.50%
σ	0.20%

B. Time-accuracy trade-off experiment

In the time-accuracy trade-off experiment, we study the trade-off between data processing time and detection accuracy. We vary the control parameters such as the threshold τ in stage 1, and observe the time to process the data along with the accuracy of the complete pipeline. Figure 11 depicts the time-accuracy trade-off for varying activity ratios (AR). The trade-off curve depicts the f1 score on the y-axis and the normalized processing time on the x-axis. F1 score and normalized processing time are defined as:

$$f1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$\text{normalized time} = \frac{\text{total time to process video data}}{\text{length of video data}}$$

where $precision$ and $recall$ carry the standard meaning. In this study, we use the f1 score as the evaluation metric as opposed to the simpler accuracy metric because we have an unbalanced dataset with respect to classes. The dataset has much more background data (-ve class) as compared to trespassing data (+ve class). Therefore, f1 score shall give a more realistic picture of the quality of the prediction.

It is clear by Figure 11 that as the f1 score goes up, the normalized processing time also goes up, indicating the trade-off. As AR decreases (as indicated by different lines in Figure 11), the trade-off curve shifts to the left. This is because in a less active environment i.e., low AR, stage 1 is able to filter out more frames and thus less frames need to be processed by stage 2. Thus the curve shifts to the left, indicating less overall processing time is consumed.

For any curve in Figure 11, with a given AR fixed, the top right point corresponds to a small value of the stage 1 threshold (τ). As τ is increased, we move towards the left; consuming less time at the cost of lower accuracy (f1 score). This is expected as stage 1 filters more and more frames with an increasing threshold (τ) and thus a lesser number of frames are processed by stage 2. The parameters controlling dataset properties used in this experiment are shown in Table V.

C. Stage-wise analysis study

In this study, we independently analyze stage 1 and 2 using f1 score and Area Under the Curve (AUC). AUC has been used as an additional accuracy metric since it measures the accuracy independent of any particular threshold while f1 score is useful in understanding the accuracy w-r-t a threshold. Table VI shows the AUC and mean processing time for each of the stages. Stage 1 has AUC of 0.94 whereas stage 2 shows an AUC of 0.81. The first stage takes 30 ms (on average) to process a frame while stage 2 takes around 500 ms on a 1080×960 sized frame. This shows that stage 1 is approx. 16.7 times faster than stage 2 which resonates with our goal stated in Section I-A.

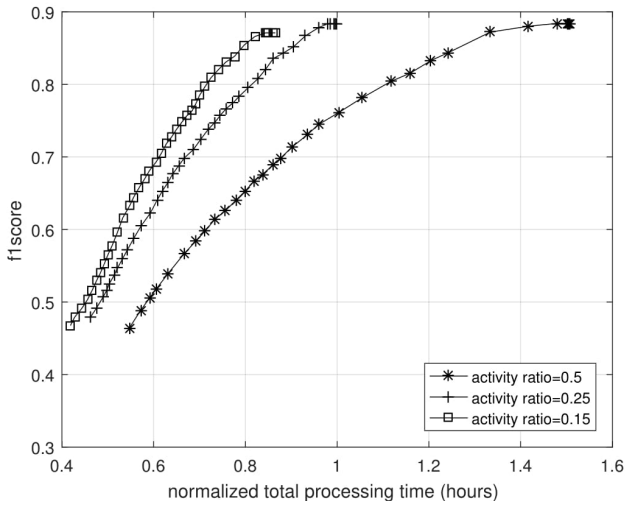

Fig. 11. Time-accuracy trade-off for varying AR

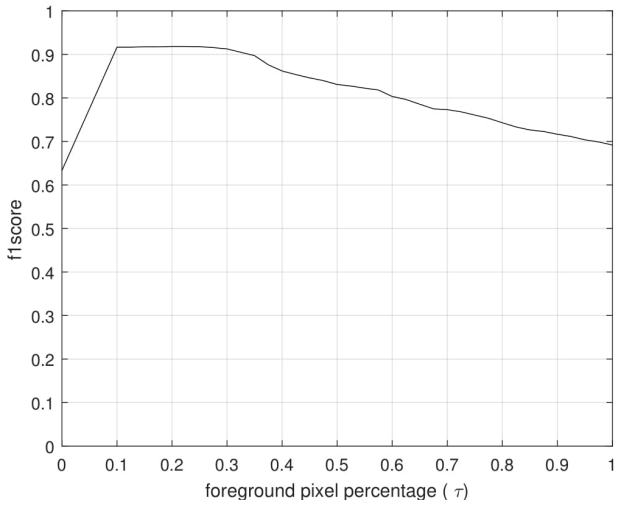

Fig. 12. Stage 1 evaluation

Figure 12 shows how the f1 score of stage 1 changes w-r-t threshold τ (percentage of foreground pixels). At 0.1%, it achieves a maximum f1 score of 0.91. Therefore, for our ARTS solution, we henceforth operate stage 1 at this threshold.

Figure 13 shows the f1 score variation for stage 2. Y-axis indicates the f1 score whereas the x-axis varies the prediction probability threshold ν. We notice that the maximum f1 score of 0.89 is achieved around $\nu = 0.3$. Therefore, we opt to operate stage 2 at this threshold.

Note that Faster-RCNN, which is the backbone of our stage 2, produces multiple object predictions corresponding to each

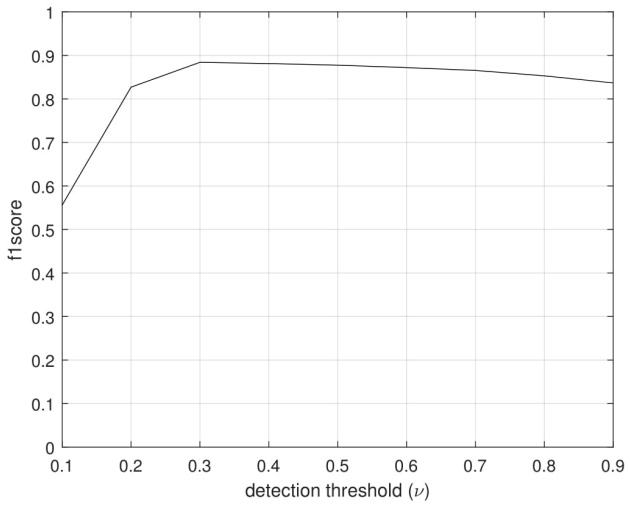

Fig. 13. Stage 2 evaluation

TABLE VI
STAGE 1 AUC AND TIME ANALYSIS

stage	AUC	mean processing time (ms)
stage 1	0.94	30
stage 2	0.81	500

input image[5]. However, we model our stage 2 as a binary classification of the input frame as either a trespassing activity frame or not. Therefore, we map Faster-RCNN predictions to binary labels, i.e., trespassing or other. For the purpose of evaluation, we define a frame to exhibit trespassing activity if there is at least one valid person detection. A person detection is said to be valid if the probability of detection is above a particular threshold ν, and the predicted bounding box sufficiently overlaps (i.e. $IOU > 0.5$) with some ground truth bounding box. Mathematically speaking:

$$\text{prediction} = \begin{cases} 1, & \text{if any valid person detection exists} \\ 0, & \text{otherwise} \end{cases}$$

where
valid person detection $= \hat{p}_i > \nu \quad \& \quad IoU(\hat{b}_i, b_j) > 0.5$,
$\hat{b}_i = i^{th}$ prediction bounding box,
$b_j = j^{th}$ ground truth bounding box,
$\hat{p}_i = i^{th}$ prediction probability,
$\nu =$ prediction probability threshold.

D. Noise analysis experiments

Third and final study of our experimental evaluation concerns noise analysis. In this part, we evaluate the robustness of stage 1 against noise. We have 3 parameters: $p, \mu,$ and σ to define the level of noise. A description of these parameters is given in Table IV. To study the influence of one parameter, we vary it while keeping the others constant.

[5]Image and frame have been used interchangeably in this text.

TABLE VII
NOISE ANALYSIS - AUC FOR VARYING p AND μ
p SWEEP: $AR = 0.5$, $\mu = 0.5\%$, $\sigma = 0.2\%$
μ SWEEP: $AR = 0.5$, $p = 2\%$, $\sigma = 0.2\%$

p	AUC	μ	AUC
2%	0.93	0.5%	0.93
4%	0.87	0.7%	0.92
6%	0.83	0.9%	0.91
8%	0.78	1.1%	0.88

Table VII shows AUC for stage 1 when varying p and μ. We use the noise parameters ($AR = 0.5$, $\mu = 0.5\%$, $\sigma = 0.2\%$) while varying p, and ($AR = 0.5$, $p = 2\%$, $\sigma = 0.2\%$) while varying μ. For both p and μ, it is clear that an increase in the noise level decreases the AUC of stage 1. This result is also confirmed by f1 score plots of the stage 1 (Figures 14 and 15). Figures 14 and 15 show the effect of changes in p and μ respectively. Figure 14 illustrates that as p increases, the overall f1 score curve shifts downward. This indicates the degradation of accuracy with an increase in p. A similar observation is also seen in Figure 15 that shows the effect of changes in μ. However, the change is less prominent in this later case.

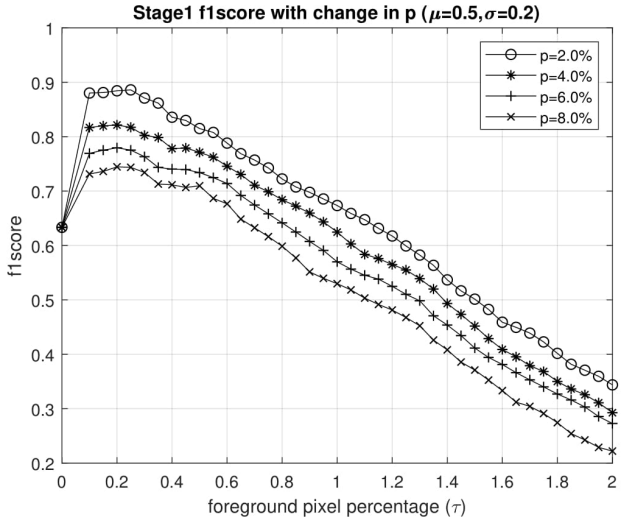

Fig. 14. Noise analysis - varying p

V. CONCLUSION AND FUTURE WORK

In this work, we propose and then comprehensively study a flexible trespassing detection solution framework (called ARTS) adopting state-of-the-art deep learning methods. The proposed framework can trade-off processing speed and accuracy. Although initially envisioned for railroad security, the proposed approach has potential applications in video surveillance domains characterized by a sparsity in activity. The ARTS framework features two stages with the first

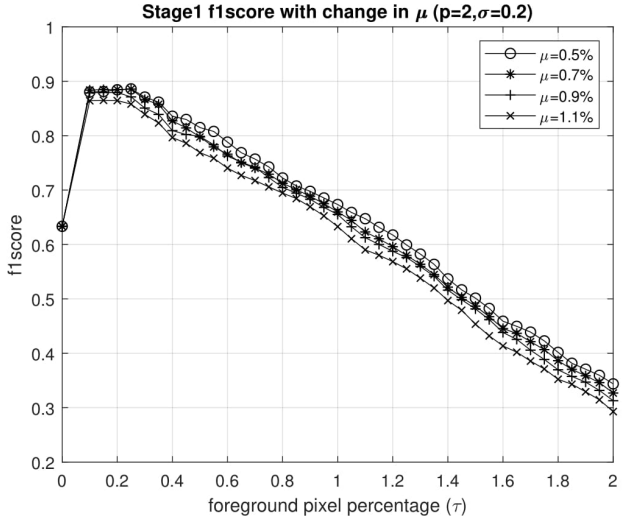

Fig. 15. Noise analysis - varying μ

stage designed to efficiently remove the background frames from the activity frames. The second stage is responsible for differentiating between human trespassing and any other unknown activity. Our proposed ARTS framework adapts a plug and play infrastructure to allow researchers to plug in other algorithms relevant to stage 1 or to stage 2, with ease in the future. The effectiveness of our approach has been demonstrated on a public domain surveillance dataset.

Future directions include building a trespassing prediction system that uses the output of the ARTS to predict future trespassing events. Another direction is to improve the accuracy of the proposed techniques. We note that the current accuracy is limited by the accuracy of stage 2. Currently stage 2 does not use any temporal information, i.e., each frame is treated independently and thus not conditioned on the previous frames (history). The utilization of the temporal information has the potential of significantly improving the accuracy specially for challenging cases of occlusion and background.

REFERENCES

[1] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *European Conference on Computer Vision*, pages 613–627. Springer, 2014.

[2] Andrew W Evans. Accidental fatalities in transport. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 166(2):253–260, 2003.

[3] Ross Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[5] Benjamin A Goldberg, Ravi K Mootha, and Ronald W Lindsey. Train accidents involving pedestrians, motor vehicles, and motorcycles. *American Journal of Orthopedics (Belle Mead, NJ)*, 27(4):315–320, 1998.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer vision*, pages 21–37. Springer, 2016.

[10] Brenda Lobb. Trespassing on the tracks: A review of railway pedestrian safety research. *Journal of Safety Research*, 37(4):359–365, 2006.

[11] Brenda Lobb, Niki Harre, and Nicola Terry. An evaluation of four types of railway pedestrian crossing safety intervention. *Accident Analysis & Prevention*, 35(4):487–494, 2003.

[12] Oge Marques. *Practical image and video processing using MATLAB*. John Wiley & Sons, 2011.

[13] Richard Matzopoulos and Leonard B Lerer. Hours to hell and back: A social epidemiology of railway injury in a south african city, 1890–1995. *Social Science & Medicine*, 47(1):75–83, 1998.

[14] Ehsan Norouznezhad, Abbas Bigdeli, Adam Postula, and Brian C Lovell. A high resolution smart camera with gige vision extension for surveillance applications. In *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–8. IEEE, 2008.

[15] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*, pages 3153–3160. IEEE, 2011.

[16] Andrew Pelletier. Deaths among railroad trespassers: the role of alcohol in fatal injuries. *JAMA*, 277(13):1064–1066, 1997.

[17] P. Wayne Power and Johann A Schoonees. Understanding background mixture models for foreground segmentation. In *Proceedings Image and Vision Computing New Zealand*, volume 2002, 2002.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[19] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[22] Houssam Salmane, Louahdi Khoudour, and Yassine Ruichek. A video-analysis-based railway–road safety system for detecting hazard situations at level crossings. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):596–609, 2015.

[23] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[24] Mubarak Shah, Omar Javed, and Khurram Shafique. Automated visual surveillance in realistic scenarios. *IEEE MultiMedia*, 14(1):30–39, 2007.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252. IEEE, 1999.

[27] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer vision*, 104(2):154–171, 2013.

[28] Zhipeng Zhang, Chintan Trivedi, and Xiang Liu. Automated detection of grade-crossing-trespassing near misses based on computer vision analysis of surveillance video data. *Safety Science*, 110:276–285, 2018.