Master's Thesis of Data Science

When to Watch: Efficient On-demand Violence Detection for Surveillance Camera

상황 판단을 기반으로 한 폭력 감지 인공 지능 모델의 효율화: 감시 카메라 시나리오를 중심으로

February 2023

Graduate School of Data Science
Seoul National University
Data Science Major

Ahreum Seo

When to Watch: Efficient On-demand Violence Detection for Surveillance Camera

Hyung-Sin Kim

Submitting a master's thesis of
Data Science

December 2022

Graduate School of Data Science
Seoul National University
Data Science Major

Ahreum Seo

Confirming the master's thesis written by
Ahreum Seo
January 2023

Chair	Wen-Syan Li	(Seal)
Vice Chair	Hyung-Sin Kim	(Seal)
Examiner	Min-hwan Oh	(Seal)

Abstract

When to Watch: Efficient On-demand Violence Detection for Surveillance Camera

Ahreum Seo
Data Science Major
Department of Data Science
Seoul National University

Recently, CCTVs are installed everywhere and play an important role in crime prevention and investigation. However, there is a problem in that a huge amount of manpower is required to monitor CCTV recordings. From this point of view, deep learning (DNN) based models that can automatically detect violence have been developed. However, they used heavy architectures such as 3D convolution or LSTM to process video data. For this reason, they require offloading to the central server for recordings to be processed so that incur huge transmission cost and privacy concern. Furthermore, given violence does not occur frequently, it is inefficient to run heavy video recognition model all the time.

To solve these problems, this study proposes *WhenToWatch*, to enhance efficiency of violence detection system on surveillance camera.

Main goals of this study are as follows: (1) To devise DNN-based violence detection system fully run on the CCTV devices to avoid offloading cost and privacy issues. (2) To reduce energy consumption of the device and processing time by introducing pre-screening module checking existence of people and deciding whether violence detection model should be executed or not. (3) To minimize computation overhead of the pre-screening module by combining lightweight non-DNN based methods and executing them according to previous status. In conclusion, *WhenToWatch* can be helpful when running violence detection models on edge devices such as CCTV, where power and computing resources are limited.

Experiments show that *WhenToWatch* can reduce the execution of the violence detection model by 17% on the RWF-2000 dataset [12] and 31% on the CCTV-Busan dataset [29]. In addition, *WhenToWatch* reduces average processing time per a video from 310.46 seconds to 255.60 seconds and average power consumption from 3,303 mW to 3,100 mW on Jetson Nano [3], confirming it contributes to efficient on-device system operation.

keywords: Violence detection, Edge AI, Video recognition, Surveillance system
student number: 2021-21846

Table of Contents

Abstract **i**

Table of Contents **iii**

1 Introduction **1**

2 Related Work **6**

2.1 Violence Detection 6

2.2 Edge AI 7

2.3 Early-skipping in Neural Networks 8

3 Methodology **9**

3.1 *WhenToWatch* Overview 9

3.2 Implementation Details of Sub-modules 12

3.3 Dataset 15

3.4 On-device Inference 16

4 Evaluation **18**

4.1 Performance of Violence Detector 18

4.2 Effect of Pre-screening Module 19

4.3 Efficiency Measurement on Jeton Nano 21

5 Discussion and Future Work **23**

5.1 Discussion and Future Work 23

6 Conclusion **24**

6.1 Conclusion 24

Bibliography **25**

Abstract in Korean **35**

Chapter 1 Introduction

CCTVs play important roles in modern society including criminal investigation and surveillance. Recent monitoring systems also enable police to intervene at the crime scene before it gets severe. However, CCTVs require huge manpower in practice. Police officers should collect and look back at recordings manually to find recorded crime scenes. Many agents also monitor surveillance videos to detect suspicious events.

For this reason, deep neural network (DNN) is emerging as a novel solution for reducing such costs, so researchers have tried to detect violence with vision-based DNN models [51, 12, 5, 27, 34, 39, 17, 38]. There are various algorithms and model architectures, aiming for smart surveillance systems that save only violent scenes or send an alarm in case of emergency.

Nonetheless, there are many limitations in current violence detection models. First, *few works can fully run on edge device* as pointed out by [22, 45] even if they target CCTVs. There are two possible explanations for this. CCTVs are resource-constrained, so it is challenging to run neural networks on CCTV devices. Fortunately, recent CCTVs get smarter and support image-based AI models such as object detection [2]. To detect violence effectively, however, complex video recognition models are needed since it depends on the context and movements. Video recognition models require more computational resources for input frames and model architectures than image recognition models. Therefore, on-device violence detection remains challenging.

A simple solution could be sending the video to the central server having abundant computing resources but it demands expensive communication costs. Furthermore, considering the rareness of the violence, most of the communications are likely to be meaningless. In addition, it may bring about privacy issues if CCTVs are installed in personal places.

Second, *running heavy DNN incessantly requires huge energy consumption and processing time.* As mentioned above, violence is abnormal and rare. However, most of the existing works assume an always-on video recognition model. Inspecting every moment thoroughly might make us feel safer but extremely inefficient in terms of energy consumption and processing time. In addition, running a violence detection model without much consideration can cause too many false alarms which notoriously bother users. It can make users careless of alarms so that they do not pay much attention to them and miss real urgent moments.

To overcome the limitations, this paper proposes *WhenToWatch*, an efficient and effective on-device violence detection network with a novel pre-screening module. The main design principles of *WhenToWatch* are three-folded.

(1) DNN-based violence detection system that can fully run on edge devices. Violence is complex human action; more than two people interact fiercely, and the context of the action should be considered to differentiate violence from normal physical contact. For this reason, most researchers have adopted complicated video recognition models such as 3D CNN[27, 46], LSTM[34, 6]. However, those require too huge computational resources to be directly run on surveillance cameras and offloading may cause cost and privacy issues as mentioned before.

Hence, *WhenToWatch* chooses lightweight depp learning architectures as its violence detector and object detector, to be fully run on edge devices. To compensate accuracy drop due to small architectures, we pretrain the violence detection model with large-scale action recognition data, Kinetics-600 [10]. It gives the model prior knowledge about general human action, helping the model to interpret the scenes more accurately. More details and experiments are described in Section 3.2 and Section 4.1.

(2) Pre-screening module to reduce the energy consumption of the device and processing time. Running such models every second might be overkill, given violence occurs infrequently. The core idea of this work is to run a violence detection model only when necessary. Violence has an important sufficient condition; *There should be human beings.* If there is no human, violence cannot happen at all. In other words, by checking the condition before running the violence detection model, *WhenToWatch* can minimize the execution of the heavy violence detection model.

In this manner, *WhenToWatch* adopts a pre-screening module to verify the existence of human beings. If the condition is not met, *WhenToWatch* skips the violence detection model and directly generates output. It can minimize the execution of the heavy DNN model and save much energy. Moreover, false-positive errors can be decreased since it cuts off the frames which are not likely to contain violence. Given *WhenToWatch* totally run on resource-constrained edge devices, simple auxiliary pre-screening can have a great effect.

(3) Combining lightweight sub-modules to minimize computation overhead of the pre-screening module. Adding DNN-based object detection models can be a straightforward solution for checking the existence of people. Unfortunately, the computational cost of the pre-screening module should be minimized given the on-device inference scenario. It means object detection models for themselves are suitable for the pre-screening module. Consequently, we choose to design the pre-screening module consisting of three sub-modules to make an efficient and effective pre-screening module. With the object detection model, lightweight motion detector and object tracker are added to the pre-screening module.

Specifically, the motion detector detects rough cues for the appearance of a new human by checking movements in the frame, assuming newly appeared humans will make movements. The object tracker tracks the people detected by the object detector and checks whether they disappear or not. They not only resolve the computational cost of the object detector but also complement each other. The motion detector cannot detect the disappearance of people accurately whereas the object tracker cannot notice people's appearance. In summary, three sub-modules interact with each other overcoming their limitations.

We summarize key contributions as follows:

- Assuming on-device inference and streaming input data, *WhenToWatch* adopts MoViNets [23] and overcome accuracy drops by pretraining the model with large-scale action recognition data and giving prior knowledge about human actions.
- *WhenToWatch*adopts a novel pre-screening module that reduces ex-

ecution of the video recognition module to reduce energy consumption and processing time. Experiments show that the execution of the violence detection model is reduced to 17% in RWF-2000 dataset and 31% in the CCTV-Busan dataset with our pre-screening module.

- After comprehensive analysis of real-world application scenario, we combine three sub-modules to make the pre-screening module. The sub-modules can complement each other, reducing the execution of DNN-based modules and detecting the existence of people more accurately.
- According to the experiments with Jetson Nano [3], it is proved that *WhenToWatch* can reduce energy consumption and processing time compared to stand-alone violence detection model.

Chapter 2 Related Work

2.1 Violence Detection

Importance of time-axis information in violence detection Violence detection refers to a task to detect violence in videos or images. Violence being a complex concept, the definition has a wide range from simple fighting [8, 17, 38] to explosion, shooting [14, 51], etc.

However violence is defined, there has been a consensus for the characteristics of violence: *It contains rapid, extreme, and abnormal movements.* In this manner, 3D CNN [27, 46], LSTM [48, 39] or their hybrid models [35, 6] are widely used to capture the change of movement aligning with time-axis. Moreover, auxiliary features such as frame difference [39, 53] or optical flow [12, 35, 57] are widely used to fully extract such movement features. [4], however, argue that a single image is enough to decide whether they are fighting or not without time-axis information. Nevertheless, [4] found that their methodology is hard to generalize and does not work on complex datasets such as Surveillance Camera Fight [5].

Violence Detection for Edge Devices There are several advantages of running a violence detection model on edge devices. Given the scale of CCTVs around us, building a central server to process all of the recordings can be too much burden. In contrast, on-device violence detection can fully exploit the resource of individual CCTVs. Given the continuous improvement of CCTV hardware, on-device inference can maximize the

efficiency of managing and analyzing CCTV recordings. Moreover, an on-device model can preserve the privacy of users and other pedestrians since CCTVs do not send or save recordings.

However, few DNN-based violence detection models can be run on CCTVs as pointed out by [22, 45]. Because of the complexity and context-dependent characteristics of violence, violence detection needs video recognition architecture, which is relatively heavy and not suitable for edge devices. Still, there are some trials for on-device violence detection with limitations. [44] set Raspberry Pi as their target device, but only the pre-screening model is run on an edge device whereas ConvLSTM-based violence detection is executed on a cloud server. [48] develop a violence detection model fully run on an edge device, using a U-Net-like network consisting of MobileNetV2 encoder and LSTM decoder. However, it cannot overcome severe accuracy drops.

2.2 Edge AI

There are roughly two mainstreams considering on-device inference of neural networks. One of them chooses to compress the model to meet the resource constraint of edge devices, by devising efficient model architecture [20, 37, 19, 41, 42, 23] or adopting model compression techniques such as quantization [50, 7], pruning [16, 28, 31] and knowledge distillation [18, 40, 21]. Their main purpose is to make the entire model efficient, so their model can be easily generalized for other scenarios. The other chooses to design an efficient system by combining other lightweight neural network models, non-neural methodologies [13, 25, 11] or of-

floading data, model to central server [54, 56]. They usually come with specific scenarios where the whole system is optimized to. This work follows the second approach in that this work adopts pre-screening module to reduce the execution of the heavy video recognition model.

2.3 Early-skipping in Neural Networks

WhenToWatch adopts pre-screening module to check the condition for violence and skip the following layers if the condition is not satisfied. Early-skipping is not new, and several researches are using the skipping mechanism to make efficient neural networks or systems. [9, 49, 26, 33, 25] end the inference without going through the rest part of the layers according to the difficulty of the task or intermediate result.

In video recognition, where time and spatial information are considered, skipping can be applied to both axes. When skipping in the time axis, the network skips the rest of the frames when highly confident decisions are made [52, 15] or sample the frames to be processed based on the pre-defined rules [43, 24]. When skipping the spatial axis, only salient space can be selected and processed further. In other words, other parts of the model are skipped when it comes to non-salient space. For example, [47] select salient patches of the picture using a lightweight policy network and process those patches only.

Chapter 3 Methodology

3.1 *WhenToWatch* Overview

In this paper, violence is defined as *violent interaction between people.* Therefore, other violent events (accidents, explosions, etc.) are out of scope in this work. From our definition of violence, violence entails a sufficient condition: the existence of human beings. An intuition here is that *if there is no human, violence cannot happen at all.* Then we can prevent unnecessary frames from being processed by a heavy violence detection network if we check the condition in advance. In addition, by filtering out the frames which are assuredly non-violent, *WhenToWatch* can reduce false positive errors caused by the violence detection module. Given false positive alarms severely bother users and can be an obstacle to introducing an automated violence detection system in the real world, our pre-screening module can contribute to better user experience and practicality.

To realize the idea, *WhenToWatch* introduces a pre-screening module to check the existence of humans and it consists of three sub-modules: motion detector, object detector, and object tracker. Figure 3.1 briefly describes the overview of *WhenToWatch*. On a high level, the pre-screening module checks whether there are people or not and save the detection results. If people are detected in more than three frames among five frames, the violence detection module is executed with cached five frames. Otherwise, the early-skipping path is opened and the output is set to non-

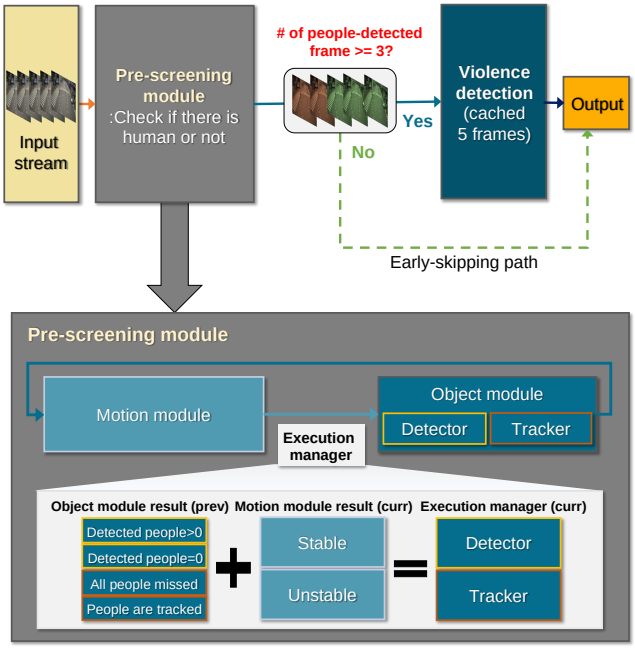


Figure 3.1: Overview of *WhenToWatch* violence detection system.

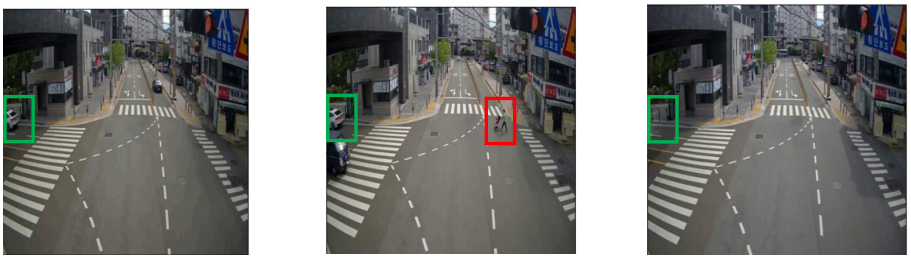
violence.

Pre-screening module consists of three sub-modules as mentioned, and the object detector and object tracker is grouped as the object module while the motion detector itself makes up the motion module. The execution manager controls the object module so either the detector or tracker or none of them is executed on each frame. Decisions of the execution manager are made according to the result of the object module on the previous frame and the result of the motion module on the current frame. The motion module detects the change in edge intensity, which implies the change in components in the frame. If there is little change in edge intensity, the motion module results in a 'stable' status (i.e., the amount of change is under threshold), otherwise results in an 'unstable' status, which means there is a change of appearance of a new person

or other objects. More details will be discussed in Section 3.2. On the other hand, the object module detects or tracks people. Object detector, relatively heavier but accurate, confirms the status of human existence while the lightweight object tracker efficiently tracks the detected people and notifies tracking status to the execution manager. Detailed process is described as follows.

- If the motion module detects the change and results in the 'unstable' status in the current frame, execute the motion detector in the current frame to verify the assumption.
- If the object detector detected no people in the previous frame and motion status is stable (i.e., there were no people and there is little chance of change in the environment), skip the object module in the current frame.
- If the object detector detected people in the previous frame and motion status is stable (i.e., there were people and there is little chance of change in the environment), run the object tracker to track detected people.
- If the object tracker tracked people successfully in the previous frame and motion status is stable, maintain executing tracker.
- If the object tracker missed all people in the previous frame, run the object detector regardless of motion status.

Figure 3.2: The constant change of traffic signs will cause false-positive when using subtraction-based motion detector.



(a) No people (b) People detected (c) No people

Figure 3.3: An example of uncertainty for baseline edge intensity. At 3.3b, people detected and disappeared. However, edge intensity of 3.3c does not return to that of 3.3a because a car also disappeared in 3.3c (i.e., edges of the car disappeared).

3.2 Implementation Details of Sub-modules

Motion detection module The main goal of a motion detector is to detect rough cues of the appearance of human beings. In general, motion detection is implemented by a simple subtraction-based method: calculating differences between a current frame and previous frames. If there are pixel differences, regard it as motion appeared. This method detects all kinds of motions but is sometimes used as a rough human detector on the assumption that moving objects are likely to be humans. It works

well in an indoor environment since there are no cars, birds, or other noisy moving objects and light changes are very subtle. However, the subtraction-based method is not suitable for outdoor environments, since there are many noises. For example, if there are traffic signs in the frame as shown in Figure 3.2, the subtraction-based motion detector will be on incessantly causing a high false-positive rate.

To overcome the challenge, this work adopts an edge detection algorithm for the motion detector since edge detection can be more robust to light change than the subtraction-based method. If edge intensity increases, it can be regarded as new objects appearing even if we do not know whether the objects are humans or not.

Still, there is a limitation remained: there is uncertainty in the baseline edge intensity. That is, we are not sure when are there no people if considering only edge intensity. In an outdoor environment, there are many moving objects other than human beings and they also cause the change in edge intensity. Therefore, even if we remember the edge intensity of a frame that contains no people, it cannot be used as a baseline for checking the 'no human' status since there can be unexpected changes in edge intensity. An example of this uncertainty is illustrated in Figure 3.3. To overcome the weakness of the motion detector, *WhenToWatch* adopts object tracker to check the disappearance of people more accurately by tracking them.

Object detection module *WhenToWatch* adopts EfficientDet [42] as object detection module. It is a well-performing and lightweight object detection model and can be easily implemented using TensorFlow Model Garden [55]. As *WhenToWatch* needs only human detection, this work

changes the configuration for the model output to contain only human detection results. In addition, the threshold for the confidence score is set to 0.2 to minimize MAE and false-negative errors after experiments with CCTV-Busan dataset.

Object tracking module Optical flow is adopted in our object tracker. In general, optical flow tracks the key points extracted by keypoint detection algorithms such as Shi-tomasi or ORB. This paper uses ORB feature [36] because of its speed and performance.

However, key points can be assigned everywhere including cars, windows, traffic signs, crosswalks, etc. Since our main interest is people previously detected by the object detector, this is not appropriate for our scenario. We tried cropping images around the bounding box and applying feature extraction but key points are hardly extracted especially when bounding boxes are too small. Therefore, ORB feature extraction is applied to the whole image, and we select a point inside each bounding box that has the highest Harris score among them.

Violence detection module Violence detection module of *WhenToWatch* adopts MoViNets [23] architecture. Especially, *WhenToWatch* use stream model of MoViNets since it uses less memory and supports faster inference on edge devices with the aid of CausalConv [32] blocks and stream buffer. Moreover, MoViNets-stream can handle stream data which is suitable for CCTV scenarios.

To be self-contained, here is a brief explanation of MoViNets-stream model. The stream buffers cache the previous frames' feature map to capture the time-level relationship and CausalConv prevents the network from making inferences with the future data. Therefore, it can process

streaming data without entire input frames so that memory usage can be reduced. Technically, it means that the model can give the results with smaller amount of input frames with the information of previous frames cached on the stream buffer. In this paper, the number of input frames is set to 8 frames for training and 5 frames for inference. Additionally, CausalConv of stream model substitute 3D Convolution layer, which is widely used in many violence detection and action recognition models but has limited support on edge devices as of now [30].

On the other hand, we adopt the smallest MoViNets architecture (a0) since there is no remarkable improvement while memory usage increases as model size gets bigger. In addition, since violence detection is a subset of general action recognition tasks to some extent, pretraining with action recognition data can give prior information for violence detection. Therefore, we pretrain violence detection module with Kinetics-600 [10] and observed significant performance gain. Detailed evaluation results are illustrated in Section 4.1.

3.3 Dataset

RWF-2000 RWF-2000 [12] contains CCTV data collected from YouTube. As the name implies, it has 2,000 clips in total, and clips are 5 seconds long. RWF-2000 has a video-level, binary label; A five-seconds video is labeled as violence or non-violence. FPS of RWF-2000 is originally set to 30, but it is reduced to 5, following the original condition of MoViNets [23], our violence detection module.

CCTV-Busan CCTV-Busan [29] is collected from CCTVs installed at

seven different places in Busan, Republic of Korea. There are 6,600 video clips and each video clip durates 3 minutes. Since it contains CCTV recordings from various locations and times, we can deliberately analyze real-world scenarios and optimize the system to them. Even though the dataset does not have violent scenes or related labels, it is used to design and evaluate our pre-screening module. Because the total length of the dataset is 330 hours, only videos from two locations are selected as a test dataset for efficiency.

3.4 On-device Inference

As mentioned in 1, *WhenToWatch* assumes on-device inference. While all CCTVs have different hardware specifications according to their cost and purpose, Jetson Nano Developer Kit (Jetson Nano) is used to prove the feasibility of our work. Jetson Nano has 128-core Maxwell GPU, Quad-core ARM 157 CPU, and 4GB RAM. It also supports external microSD storage, and camera extension [3].

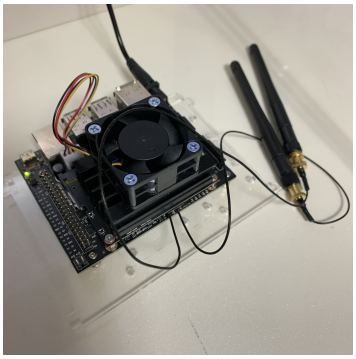


Figure 3.4: NVIDIA Jetson Nano Developer Kit

Optimization for Jetson Nano The DNN-based modules, object de-

tector and violence detector, are implemented by TensorFlow. Even if Jetson Nano supports TensorFlow framework, some optimizations are needed to run the models on Jetson Nano efficiently. Object detector, which is based on EfficientDet, is converted into TensorRT [1] model for optimized inference on the GPU of Jetson Nano. The violence detector, based on MoViNets-stream model, consists of many custom objects such as stream buffer and customized train/test graph, so it is not compatible with TensorRT as of now. Instead, it is converted to TensorFlow Lite framework for faster inference, while GPU processing is not supported on Jetson Nano.

Chapter 4 Evaluation

4.1 Performance of Violence Detector

In this paper, MoViNets [23] is adopted as the violence detection module. With the aid of CausalConv and stream buffer, MoViNets can handle streaming video data with minimal memory cost which is helpful in our on-device CCTV analysis scenario. Specifically, CausalConv prevents the model from inferencing from future data, and stream buffer caches results of previous frames so that the model do not need to hold entire video frames. With MoViNets architecture, several experiments are conducted to maximize the accuracy of violence detection and the results are described below.

Effect of pretraining and model size Since violence can be regarded as a subset of action, general action data can give essential prior knowledge for the violence detection task. To prove the hypothesis, we pretrain the MoViNets with large-scale action recognition data, Kinetics-600 [10], and retrain the model with RWF-2000 data without freezing layers.

In addition, since there are two types of MoViNets according to model architecture and input data type, we compare the results with both of them. MoViNets-base refers to the model without stream buffer. It needs an entire video clip for inference, which is the same as previous violence detection research including original RWF-2000 paper [12]. On the contrary, MoViNets-stream is a model with stream buffer so it can process streaming data and give a result every 5 frames, for example. Because

Table 4.1: The effect of pretraining and model size. The result shows pretraining with action recognition data (Kinetics-600) leads to performance improvement while increasing model size does not.

Model	Accuracy (%)	
	Pretrained	From scratch
MoViNets-stream-a0 (ours)	**90.25**	79.75
MoViNets-stream-a1	89.50	81.75
MoViNets-stream-a2	87.75	**82.25**

MoViNets-stream is more appropriate for our CCTV scenario but has different inference conditions, we conduct experiments with both stream and base model to separate the effect of pretraining from other factors.

As shown in Table 4.1, pretraining brings remarkable accuracy gain in both base and stream models. MoViNets-base model achieves state-of-the-art 91.25% accuracy while the MoViNets-stream model achieves 87.25% accuracy. Nonetheless, both models show explicit improvement after pretraining with action recognition data, 13.75%p and 6.5%p, respectively.

Table 4.1 also shows bigger models do not guarantee performance improvement. Rather, bigger models suffer from overfitting which causes performance degradation. As a result, the smallest model, MoViNets-stream-a0, is the best option given both accuracy and memory usage.

4.2 Effect of Pre-screening Module

The main goal of this paper is to reduce the execution of the violence detection module since it is a heavy video recognition model. Moreover, violence is rare in the real world and cannot happen when there are no people, turning off the violence detection module when not necessary

Table 4.2: Accuracy and number of frames processed by each module, tested on RWF-2000 dataset.

Pre-screening module	Object tracking module	Accuracy on RWF-2000 (%)	Object detector execution	Object tracker execution	Violence detector execution
X	X	**90.25**	-	-	10,000
O	X	88.50	1,870	-	8,375
O	**O**	88.00	**887**	**7,531**	**8,290**

Table 4.3: Accuracy and number of frames processed by each module, tested on CCTV-Busan dataset.

Pre-screening module	Object tracking module	Accuracy on RWF-2000 (%)	Object detector execution	Object tracker execution	Violence detector execution
X	X	**100.00**	-	-	71,280
O	X	100.00	1,870	-	-
O	**O**	100.00	**1,243**	**59,742**	**48,680**

can greatly improve the efficiency of the surveillance system.

In this section, we compare the accuracy and the number of execution by each module according to the different system architecture designs. There are three types of design: (1) Without the pre-screening module, (2) With the pre-screening module, but the module consists of the only motion detector and object detector, (3) With pre-screening module and the module includes all three sub-modules including object tracker (*WhenToWatch*). Note that the input resolution is 512x512.

Table 4.2 shows the result with the RWF-2000 dataset, originally collected for the violence detection task. The original accuracy without the pre-screening module is 90.25% and accuracy drops with the pre-screening module. However, the number of violence detector execution is reduced to 8,290 (17.10% decrease rate). In addition, the execution of the object detector is halved with the adoption of the object tracker. This means the computation cost of the pre-screening module is further reduced with the object tracker.

Table 4.3 shows a similar result with the CCTV-Busan dataset. Since directly collected from real-world CCTVs, this dataset is more likely to contain the frames with no people. That is why the decrease in the execution of the violence detector is more remarkable, from 71,280 frames to 48,680 frames (31.70% decrease rate). Meanwhile, the decrease rate in execution of the object detector is less than that of the RWF-2000 dataset. A possible explanation is the size of a human is smaller in the CCTV-Busan dataset, causing frequent missing by the object tracker. Lastly, no false-positive error is observed in every system design.

4.3 Efficiency Measurement on Jeton Nano

The main goal of *WhenToWatch* is to make an efficient surveillance system by reducing power consumption and execution time when running on an edge device. *WhenToWatch* utilizes pre-screening module and early-skipping gate to pass the heavy violence detection module. To prove the design of *WhenToWatch* contribute to efficiency, we implement *WhenToWatch* on Jetson Nano and conduct experiments with the CCTV-Busan dataset which more reflects the real-world environment.

To verify the efficiency of the pre-screening module, table 4.4 shows the average execution time for the sub-modules of the pre-screening module and violence detection model. Notably, non-DNN-based sub-modules, motion detector and object tracker, can be executed within about 10ms. Object detector is relatively slower among sub-modules, and it means that the more frames are executed by the object tracker or neither of them, the faster the whole system can be executed. Note that the violence de-

Table 4.4: Average execution time for each sub module implemented on Jetson Nano.

Module	Avg. execution time (sec.)
Motion detector	0.012
Object detector	0.153
Object tracker	0.006
Violence detector	2.150

Table 4.5: The effect of pre-screening module on processing time and power consumption on Jetson Nano.

Pre-screening module	Avg. processing time per video (sec.)	Avg. power consumption during evaluation (mW)
X	310.46	3,303
O (ours)	**255.60**	**3,100**

tector processes five frames at once while the object detector processes one frame. If the object detector is executed five frames in a row, it might be an extra burden. However, after the execution of the object detector, the object tracker or only the motion detector without the object module will be executed as described in Section 3.1.

Table 4.5 gives more comprehensive results about the effectiveness of the pre-screening module. With reduced execution of the violence detection module, the average processing time of *WhenToWatch* is reduced from 310.46 seconds to 255.60 seconds. Average power consumption during the whole evaluation is also reduced from 3,303 mW to 3,100 mW. It further supports our hypothesis and the effectiveness of *WhenToWatch*.

Chapter 5 Discussion and Future Work

5.1 Discussion and Future Work

Efficiency of object detector However, there are some limitations. Since the object detection model needs to be accurate, we adopt a DNN-based model which can be another bottleneck of the system. Moreover, EfficientDet consumes relatively more memory than other object detection networks because of BiFPN architecture. Therefore, a more lightweight and efficient object detection network can further improve the efficiency of *WhenToWatch*.

False-negative errors When analyzing error cases, accuracy reduction is mainly due to false-negative errors of the object detector. In other words, if the object detector fails to detect a human, the violence detector will not be executed even if there is violence. For future work, a more accurate object detection model can be considered or other complementary architecture can reduce false-negative errors.

Chapter 6 Conclusion

6.1 Conclusion

This work proposes a novel violence detection system for surveillance cameras to reduce computational cost and energy consumption given the on-device inference scenario. With a deliberate and efficient pre-screening module, *WhenToWatch* can run the violence detection model only when the sufficient condition for violence is met. Experiments show that our system design can contribute to efficient system operation by reducing the execution of heavy DNN networks. We hope this kind of approach can contribute to the efficient video recognition system for surveillance cameras.

Bibliography

[1] Nvidia tensorrt.

[2] The best indoor cameras for artificial intelligence, Mar 2022.

[3] Jetson nano developer kit, Sep 2022.

[4] Şeymanur Aktı, Ferda Ofli, Muhammad Imran, and Hazim Kemal Ekenel. Fight detection from still images in the wild. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 550–559, 2022.

[5] Şeymanur Aktı, Gözde Ayşe Tataroğlu, and Hazım Kemal Ekenel. Vision-based fight detection from surveillance cameras. In *2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6. IEEE, 2019.

[6] Mujtaba Asad, Jie Yang, Jiang He, Pourya Shamsolmoali, and Xiangjian He. Multi-frame feature-fusion-based model for violence detection. *The Visual Computer*, 37(6):1415–1431, 2021.

[7] Mustafa Ayazoglu. Extremely lightweight quantization robust real-time single-image super resolution for mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2472–2479, 2021.

[8] Enrique Bermejo Nievas, Oscar Deniz Suarez, Gloria Bueno García, and Rahul Sukthankar. Violence detection in

video using computer vision techniques. In *International conference on Computer analysis of images and patterns*, pages 332–339. Springer, 2011.

[9] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR, 2017.

[10] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[11] Kaifei Chen, Tong Li, Hyung-Sin Kim, David E Culler, and Randy H Katz. Marvel: Enabling mobile augmented reality with low energy and low latency. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 292–304, 2018.

[12] Ming Cheng, Kunjing Cai, and Ming Li. Rwf-2000: an open large scale video database for violence detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4183–4190. IEEE, 2021.

[13] Yousung Choi, Ahreum Seo, and Hyung-Sin Kim. Scriptpainter: Vision-based, on-device test script generation for mobile systems. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 477–490. IEEE, 2022.

[14] Claire-Hélène Demarty, Cédric Penet, Mohammad Soleymani, and Guillaume Gravier. Vsd, a public dataset for the detection of violent scenes in movies: design, annotation, analysis and evaluation. *Multimedia Tools and Applications*, 74(17):7379–7404, 2015.

[15] Hehe Fan, Zhongwen Xu, Linchao Zhu, Chenggang Yan, Jianjun Ge, and Yi Yang. Watching a small portion could be as good as watching all: Towards efficient video classification. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[17] Tal Hassner, Yossi Itcher, and Orit Kliper-Gross. Violent flows: Real-time detection of violent crowd behavior. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6. IEEE, 2012.

[18] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[21] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[22] Gurmeet Kaur and Sarbjeet Singh. Violence detection in videos using deep learning: A survey. *Advances in Information Communication Technology and Computing*, pages 165–173, 2022.

[23] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. Movinets: Mobile video networks for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16020–16030, 2021.

[24] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6232–6242, 2019.

[25] Sam Leroux, Steven Bohez, Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. The cascading neural network: building the internet of smart things. *Knowledge and Information Systems*, 52(3):791–814, 2017.

[26] Sam Leroux, Steven Bohez, Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. The cascading neural network: building the internet of smart things. *Knowledge and Information Systems*, 52(3):791–814, 2017.

[27] Ji Li, Xinghao Jiang, Tanfeng Sun, and Ke Xu. Efficient violence detection using 3d convolutional neural networks. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2019.

[28] Jiachen Mao, Huanrui Yang, Ang Li, Hai Li, and Yiran Chen. Tprune: Efficient transformer pruning for mobile devices. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–22, 2021.

[29] Republic of Korea National Information society Agency (NIA). Cctv video dataset for floating population analysis. https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=489, 2022.

[30] Kondratyuk NDan, Yuan Liangzhe, and Li Yeqing. Github repository of movinets. https://github.com/tensorflow/models/tree/master/official/projects/movinet, 2021.

[31] Wei Niu, Xiaolong Ma, Sheng Lin, Shihao Wang, Xuehai Qian, Xue Lin, Yanzhi Wang, and Bin Ren. Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning. In *Proceedings of the Twenty-Fifth International Conference on Archi-*

tectural Support for Programming Languages and Operating Systems, pages 907–922, 2020.

[32] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[33] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong-Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. Big/little deep neural network for ultra low power inference. In *2015 international conference on hardware/software codesign and system synthesis (codes+ isss)*, pages 124–132. IEEE, 2015.

[34] Fernando J Rendón-Segador, Juan A Álvarez-García, Fernando Enríquez, and Oscar Deniz. Violencenet: Dense multi-head self-attention with bidirectional convolutional lstm for detecting violence. *Electronics*, 10(13):1601, 2021.

[35] Fernando J Rendón-Segador, Juan A Álvarez-García, Fernando Enríquez, and Oscar Deniz. Violencenet: Dense multi-head self-attention with bidirectional convolutional lstm for detecting violence. *Electronics*, 10(13):1601, 2021.

[36] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[37] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and

linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[38] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999.

[39] Swathikiran Sudhakaran and Oswald Lanz. Learning to detect violent videos using convolutional long short-term memory. In *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[40] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

[41] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[42] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[43] Yansong Tang, Yi Tian, Jiwen Lu, Peiyang Li, and Jie Zhou. Deep progressive reinforcement learning for skeleton-based action recog-

nition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5323–5332, 2018.

[44] Fath U Min Ullah, Khan Muhammad, Ijaz Ul Haq, Noman Khan, Ali Asghar Heidari, Sung Wook Baik, and Victor Hugo C de Albuquerque. Ai-assisted edge vision for violence detection in iot-based industrial surveillance networks. *IEEE Transactions on Industrial Informatics*, 18(8):5359–5370, 2021.

[45] Fath U Min Ullah, Mohammad S. Obaidat, Amin Ullah, Khan Muhammad, Mohammad Hijji, and Sung Wook Baik. A comprehensive review on vision-based violence detection in surveillance videos. *ACM Comput. Surv.*, aug 2022. Just Accepted.

[46] Fath U Min Ullah, Amin Ullah, Khan Muhammad, Ijaz Ul Haq, and Sung Wook Baik. Violence detection using spatiotemporal features with 3d convolutional neural network. *Sensors*, 19(11):2472, 2019.

[47] Thomas Verelst and Tinne Tuytelaars. Blockcopy: High-resolution video processing with block-sparse feature propagation and online policies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5158–5167, 2021.

[48] Romas Vijeikis, Vidas Raudonis, and Gintaras Dervinis. Efficient violence detection in surveillance. *Sensors*, 22(6):2216, 2022.

[49] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[50] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4820–4828, 2016.

[51] Peng Wu, Jing Liu, Yujia Shi, Yujia Sun, Fangtao Shao, Zhaoyang Wu, and Zhiwei Yang. Not only look, but also listen: Learning multimodal violence detection under weak supervision. In *European conference on computer vision*, pages 322–339. Springer, 2020.

[52] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, Yi Yang, and Shilei Wen. Dynamic inference: A new approach toward efficient video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 676–677, 2020.

[53] Qing Xia, Ping Zhang, JingJing Wang, Ming Tian, and Chun Fei. Real time violence detection based on deep spatio-temporal features. In *Chinese Conference on Biometric Recognition*, pages 157–165. Springer, 2018.

[54] Juheon Yi, Sunghyun Choi, and Youngki Lee. Eagleeye: Wearable camera-based person identification in crowded urban spaces. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[55] Hongkun Yu, Chen Chen, Xianzhi Du, Yeqing Li, Abdullah Rashwan, Le Hou, Pengchong Jin, Fan Yang, Frederick

Liu, Jaeyoun Kim, and Jing Li. Tensorflow model garden. https://github.com/tensorflow/models, 2020.

[56] Wuyang Zhang, Zhezhi He, Luyang Liu, Zhenhua Jia, Yunxin Liu, Marco Gruteser, Dipankar Raychaudhuri, and Yanyong Zhang. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 201–214, 2021.

[57] Peipei Zhou, Qinghai Ding, Haibo Luo, and Xinglin Hou. Violent interaction detection in video based on deep learning. In *Journal of physics: conference series*, volume 844, page 012044. IOP Publishing, 2017.

요약 (국문초록)

최근에는 안전을 위해 CCTV가 곳곳에 설치되어 있으며 범죄 예방 및 수사에 중요한 역할을 하고 있다. 그러나 CCTV 영상들을 실시간으로 감시하거나 녹화된 영상을 재검토하기 위해서는 막대한 인력이 필요하다는 문제점이 있다. 이러한 관점에서 자동으로 폭력을 감지할 수 있는 딥러닝 모델들이 꾸준히 개발되어왔다. 그러나 대부분의 모델은 3D 컨볼루션, LSTM 등의 무거운 영상처리 모델을 사용했기 때문에 CCTV 디바이스 내에서의 추론은 거의 불가능했고, 서버로 영상을 전송하여 처리하는 것을 전제로 한다. 이 경우 막대한 전송 비용이 발생할 뿐만 아니라 사생활 침해 문제가 발생할 소지가 있다. 뿐만 아니라, 폭력은 일반적인 사건에 비해 발생 빈도가 낮다는 점을 고려한다면 CCTV 동작 시간 내내 무거운 폭력 감지 모델을 구동하는 것은 비효율적이라고 할 수 있다.

이러한 문제점들을 해결하고 폭력 감지 시스템의 효율성을 제고하기 위해 본 연구에서는 *WhenToWatch*라는 폭력 감지 시스템을 제안한다. 본 연구의 주요 목적은 다음과 같다. (1) 데이터 전송 비용을 최소화하고 개인정보를 보호하기 위해 감시카메라 장치 내에서 구동 가능한 딥러닝 기반의 폭력 감지 시스템을 제안한다. (2) 감시카메라 장치의 전력 소모량과 데이터 처리 시간을 줄이기 위해 사전 판단 모듈을 도입한다. 이를 통해 사람의 존재 여부를 판단하고 폭력 감지 모델의 실행 여부를 결정함으로써 불필요한 연산량을 줄일 수 있다. (3) 사전 판단 모듈로 인한 추가적인 연산량 부담을 최소화하기 위해 실행속도가 빠른 비 딥러닝 기반의 방법론들을 결합한 시스템을 디자인하고, 이전 상태에 따라 적절한 연산을 실행한다. 최종적으로 *WhenToWatch*는 CCTV와 같이 리소스가 제한된 엣지 디바이스에서 폭력 감지 모델을 효율적으로 구동할 수 있게 한다.

실제 실험 결과, 제안된 사전 판단 모듈을 적용했을 때, 폭력 감지 모델의 실행 횟수는 RWF-2000 [12] 데이터셋에서 약 17% 감소했으며 CCTV-Busan [29] 데이터셋에서는 약 31% 감소하는 것으로 나타났다. 본 논문의

시스템 구조를 통해 보다 효율적인 시스템 운영이 가능함을 확인할 수 있었다. 또한 젯슨 나노 [3]에서 평균 비디오 처리 시간은 310.46초에서 255.60초로 감소하였으며 전력 소모량은 3,303 mW에서 3,100 mW로 감소하여 *WhenToWatch*가 효율적인 온디바이스 시스템 운영에 기여할 수 있음을 보여주었다.

주요어: 폭력 감지, 엣지 컴퓨팅 및 인공 지능, 영상 인식, CCTV
학번: 2021-21846