Vietnamese-German University

COPYRIGHT WARNING

This paper is protected by copyright. You are advised to print or download **ONE COPY** of this paper for your own private reference, study and research purposes. You are prohibited having acts infringing upon copyright as stipulated in Laws and Regulations of Intellectual Property, including, but not limited to, appropriating, impersonating, publishing, distributing, modifying, altering, mutilating, distorting, reproducing, duplicating, displaying, communicating, disseminating, making derivative work, commercializing and converting to other forms the paper and/or any part of the paper. The acts could be done in actual life and/or via communication networks and by digital means without permission of copyright holders.

The users shall acknowledge and strictly respect to the copyright. The recitation must be reasonable and properly. If the users do not agree to all of these terms, do not use this paper. The users shall be responsible for legal issues if they make any copyright infringements. Failure to comply with this warning may expose you to:

- Disciplinary action by the Vietnamese-German University.
- Legal action for copyright infringement.
- Heavy legal penalties and consequences shall be applied by the competent authorities.

The Vietnamese-German University and the authors reserve all their intellectual property rights.

VIETNAMESE – GERMAN UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Frankfurt University of Applied Sciences

Faculty 2: Computer Science and Engineering

Thesis topic: Crime-scenes Detection from Surveillance videos directly on Jetson edge devices system

Full name: Pham Viet Hoang

Matriculation number: 1321092

First supervisor: Dr. Vo Bich Hien

Second supervisor: Prof. Dr. Huynh Trung Hieu

BACHELOR THESIS

Submitted in partial fulfillment of the requirements for the degree of Bachelor Engineering in study program Computer Science, Vietnamese - German University, 2018

Binh Duong, Viet Nam

Abstract

The current concept of Smart Cities aims to provide modern, secure, and sustainable infrastructure and give residents a decent quality of life. To achieve this goal, video surveillance cameras have been deployed to enhance the safety and well-being of the citizens. However, abnormal event detection in surveillance video systems is challenging and requires exhaustive human efforts despite technical developments in modern science. In this project, end-to-end crime-scene anomaly detection on surveillance camera systems is proposed and implemented directly on edge devices. Although the output only determines the scene to be normal or abnormal, the concept of anomaly events stands for nine types of crimes scene: Arson, Assault, Burglary, explosion, Fighting, Road Accident, Shooting, Stealing, and Vandalism. The project is a deep learning-based method with the help of some edge AI techniques and code-based structure from the Facebook open-source Slowfast project, which is re-implemented to apply to the AI system successfully.

The system collects camera video data and performs anomaly detection directly on Jetson edge devices with the highest inference speed. I searched for the state-of-the-art (SOTA) models based on the UCF-Crime dataset for anomaly detection problems on surveillance videos problem on Paperwithcode [1] and chose weakly-supervised video anomaly detection with robust temporal feature magnitude learning (RTFM) as the final anomaly detection AI solution. The metric for the evaluation process is Area Under the ROC curve (AUC). The AI solutions achieved a 0.8439 AUC score on the UCF-Crime dataset and a 0.884 AUC score on the VNAnomaly dataset collected in Vietnam from Ho Chi Minh City University of Information Technology (UIT). The system was also successfully deployed and tested on Jetson Nano with 1.55 FPS inference speed by using TensorRT edge AI techniques with only 2.61 GB RAM usage total.

Keywords: Anomaly Detection, Jetson, Edge computing, Edge AI, Slowfast

Declaration of Authorship

I, Pham Viet Hoang, hereby declare to the best of my knowledge that this bachelor thesis, which is submitted to the Vietnamese-German University and Frankfurt University of Applied Sciences, is a product of my work unless otherwise referenced. I also claim that all opinions, results, and conclusions are mine. Besides, I declare that this thesis has not been previously or concurrently submitted, in part or whole, at any other universities or institutions.

Pham Viet Hoang
1321092
CSE2018

Acknowledgement

This bachelor's thesis cannot be accomplished without the support and assistance of numerous people. I want to show my gratitude towards each individual associated directly or indirectly during my thesis.

I want to express my deep appreciation to my two supervisors, Dr. Vo Bich Hien and Prof. Dr. Huynh Trung Hieu, for their continuous guidance, constant support, and supervision despite having a hectic schedule.

Lastly, to all my CSE2018 classmates, I would like to show my sincere appreciation for providing lots of support and continuous encouragement while studying at VGU. They have also shared their experiences and advice to help me a lot through researching this topic. I would not accomplish this thesis and completed my study program without your help and guidance. Thank you!

Contents

List of Figures

List of Tables

1 Introduction

With the growing need for protection for people and personal properties, video surveillance has drawn much concern in daily life. Considering these needs have led to the deployment of cameras in almost every corner, video surveillance systems can interpret the scene and automatically recognize abnormal behaviors, which play a vital role in intelligence monitoring. Anomaly detection is an essential task in surveillance video collected from edge devices. However, it poses several challenges, including edge devices' limited computational resources and bandwidth. There are different approaches to the problem, such as deep learning-based and rule-based methods. Each system has its advantages and disadvantages. For example, deep learning-based methods can achieve high accuracy but require large amounts of data and computational resources. Otherwise, rule-based methods are more straightforward but may not be as accurate.

On the other hand, the edge device system or system on the chip (SoC) currently achieves some standout advancements to keep track of the industrial need. Edge computing is a distributed computing paradigm that brings computation and data storage closer to the data sources. This is expected to improve response times and save bandwidth. Edge computing is an architecture rather than a specific technology and a topology - and location-sensitive form of distributed computing [26]. According to [7], spending on edge computing in 2022 is projected to increase 14.8% from the last year, and by 2026, the market is looking to reach $17.8 billion dollars. With the advantages of AI and Edge Computing, exciting ideas and projects are proposed to solve industrial problems.

Artificial Intelligence (AI) is the computer systems simulation of human intelligence processes. These processes include learning (acquiring information and rules for using the data), reasoning (using rules to reach approximate or definite conclusions), and self-correction. AI has been around for decades but has recently gained more attention due to technological advancements and the increasing amount of data available. One field of Artificial Intelligence is Computer vision. It enables computers to interpret and understand the visual world. It has many applications in various industries, such as healthcare, automotive, retail, and more. According to a report by McKinsey Global Institute [15], AI has the potential to create between $3.5 trillion and $5.8 trillion in value annually across nine business functions in 19 industries. Another report by Gartner predicts that by 2025, 75% of enterprise-generated data will be created and processed outside a traditional centralized data center or cloud.

Edge AI is a technology that enables data processing and artificial intelligent prediction to be performed on devices at the edge of the network rather than in the cloud traditionally. Many advantages are provided to improve the company user experiences, such as reduced latency, real-time analytics, higher speeds, reduced bandwidth requirement and cost, enhanced data security, more effortless scalability, etc. [16]

By combining the advancement of the above technologies, an end-to-end crime detection system is constructed, implemented, and showcased in this project. Figure (1) shows the general architecture of the whole system, from edge computing, Edge Artificial Intelligent.

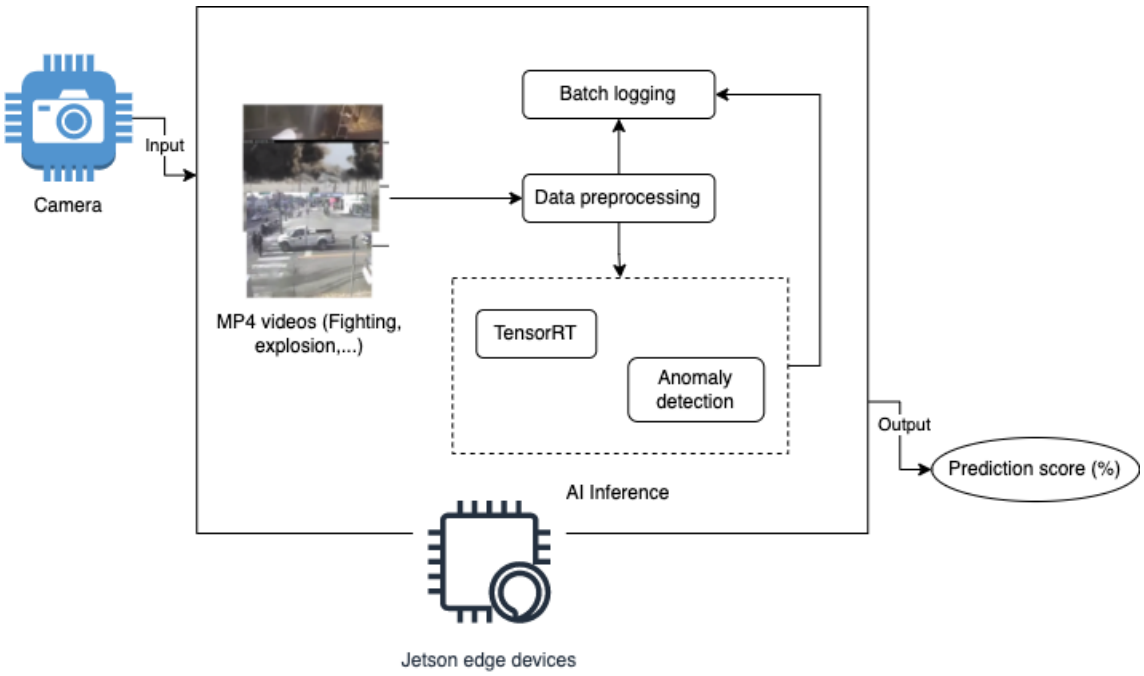

Figure 1: General anomaly detection system on surveillance videos architecture.

A surveillance camera connects with Jetson devices to collect video in real time. The output videos are then inputted into the data processing pipeline to extract the videos' spatial and temporal features. The deep learning-based anomaly detection model consumes these features to detect abnormal events from them. There are totally two deep learning models in this system. The first is the feature extractor which extracts the part directly from MP4 video input. The second one is the anomaly detection model to detect abnormal events from the output features of the extractor. The input video is divided into 16-frame clips each. The extractor processes each clip to construct each feature unit. The extractor should extract all the features from the video input before adding them to the second model. The anomaly detection then refines those features and performs the detection process to output the abnormal prediction score. However, deploying and running the whole AI system on Jetson edge devices requires some essential optimization techniques to boost the inference speed and minimize the RAM usage. Edge AI techniques such as Torch-TensorRT are fully installed and applied to the system to satisfy this requirement.

2 Related Works

Video surveillance cameras have been deployed to enhance the safety and well-being of citizens in smart cities. However, detecting abnormal events in surveillance video systems is challenging and requires exhaustive human efforts. There are various methodologies developed to detect anomalies in intelligent video surveillance. Based on what Devashree R. Patrikar [3] propose, some developed methods are listed to have an overview of the systematic categorization of algorithms for anomaly detection problems on surveillance videos.

Over some time, by following the purpose of anomaly detection, different researchers applied various methods. In the early period, the majority of algorithms were handcrafted that includes Histogram of Oriented Gradient (HOG), Histogram of Optical Flow (HOF)[23], etc.

The development of machine learning allowed for the automatic understanding of spatiotemporal features using neural networks. However, some critical issues, such as gradient vanishing or exploding, still existed and prevented further improvements. Until 2011, a solution to those problems was proposed in the form of the Deep Sparse Rectifier Neural Network [28], which used ReLu as an activation function. This solution was a significant breakthrough in the history of neural networks and led to the development of convolutional neural networks (CNNs) that could automatically learn spatiotemporal features.

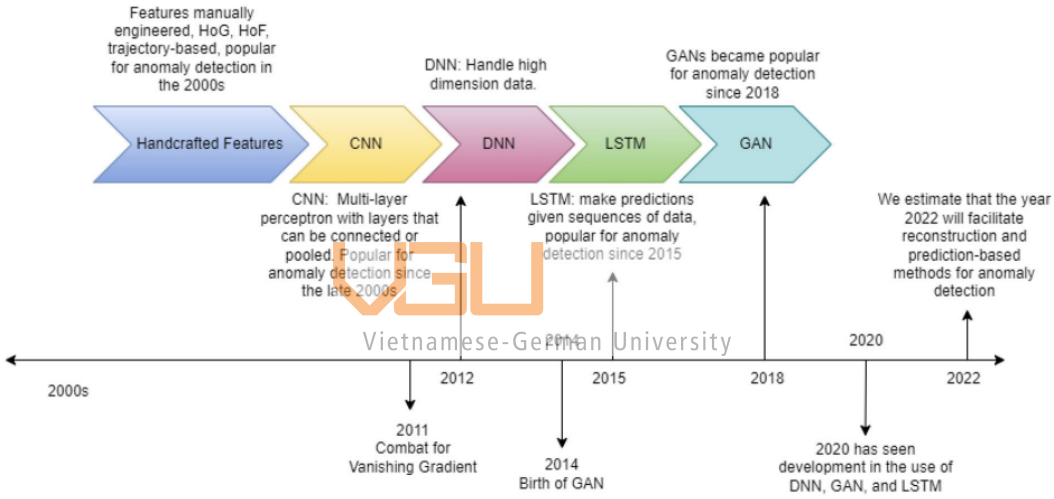

Figure 2: Evolution of anomaly detection techniques timeline from [3].

Since 2012, Deep CNN networks with multiple hidden layers have been used to handle high dimensional data to improve prediction accuracy and extract more features. However, generating realistic images with many features for tasks such as image detection, classification, and reconstruction remains challenging. In 2014, Ian Goodfellow and his colleagues developed a model using generative modeling that can learn any data distribution in an unsupervised way. The model is called Generative Adversarial Network (GAN) [10]. It consists of two sub-network: a generator that creates authentic images and a discriminator that distinguishes between generated and natural images. GANs have become popular in the research community since 2018 and have been successfully implemented for anomaly detection. 2020 has seen the development in the use of DNN, GANs, and Long short-term memory (LSTM) for many anomaly detection methodologies. In 2022, the reconstruction and prediction-based approaches for anomaly detection problems were proposed based on the inspiration from transformer [2]. My final AI solution for the thesis is also the result of transformer inspiration.

One of the crucial issues when deploying anomaly detection systems directly on edge devices is the usage of high RAM storage and computational resources. Mayur R. Parate [14] proposed a modular framework that captures object-level information and tracking. The author uses feature encoding and trajectory associations to handle partial occlusions, posture deformations, and complex scenes. The lightweight anomaly detection framework elements are developed and optimized for CPU-only edge devices to achieve real-time performance. The author tested the system on various datasets, including UCSD Ped-1 and UMN datasets, and showed competitive results compared to the previous methods. Figure 3 shows the general architecture of the anomaly detection deployed on CPU-only IoT frameworks related to my thesis project. Figure 4 shows the hardware system of the anomaly detection deploys on CPU-only IoT frameworks.

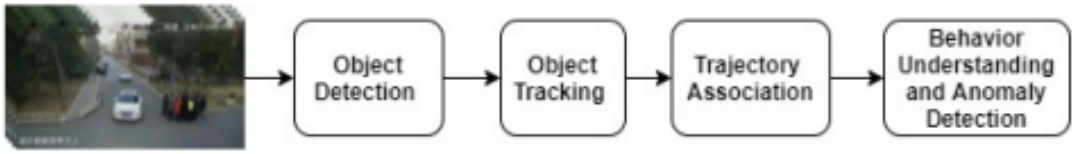

Figure 3: General flow of Anomaly Detection IoT Frameworks from [14]

Figure 4: Hardware system of Anomaly Detection IoT Frameworks from [14].

Muhammad Islam [17] introduces an efficient framework for detecting anomalies in extensive surveillance video data using Artificial Intelligence of Things (AIoT). He pointed out that automated anomaly detection techniques using computer vision can replace human intervention in securing video surveillance applications, making them more efficient and accurate than traditional systems that rely on human involvement. However, automatic detection in real-world scenarios is challenging due to abnormal events' diverse and complex nature. Authors proposed a hybrid model integrating 2D-CNN and ESN [11] for their intelligent surveillance solution. The CNN extracts feature from input videos, which are then refined by an autoencoder and inputted into the ESN for sequence learning and anomaly detection. The methods show competitive results on different datasets, including UCF, Shanghai-Tech, etc. Figure 5 shows the general architecture of the above second related work.

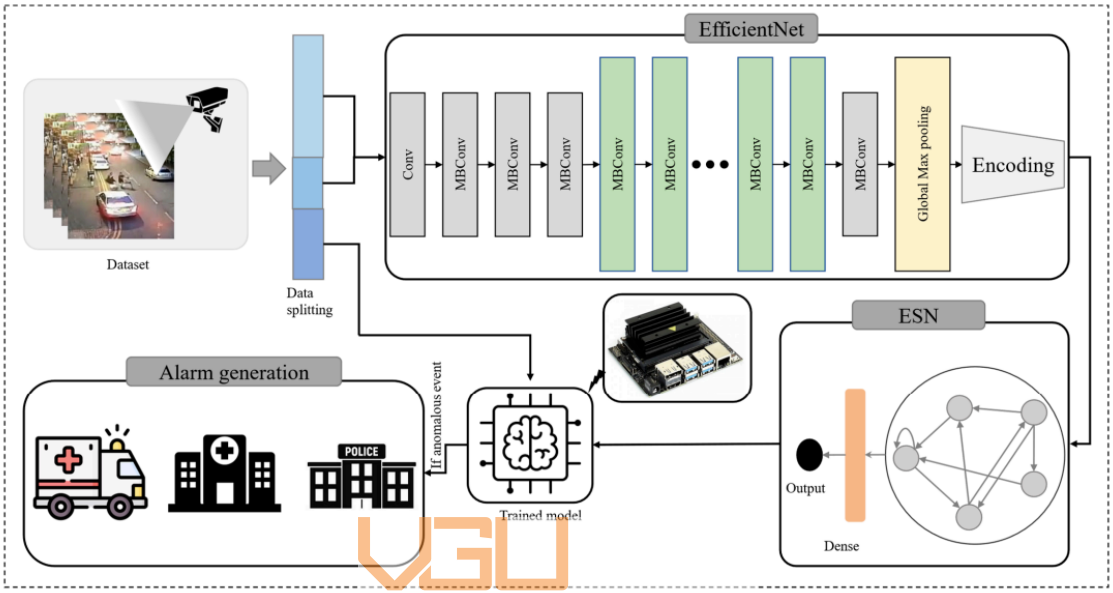

Figure 5: General architecture of the second related work from [17].

3 Methods

3.1 Hardware

NVIDIA Jetson brings accelerated AI performance to the Edge in a power-efficient and compact form factor. Together with NVIDIA JetPack SDK, these Jetson modules open the door for you to develop and deploy innovative products across all industries, including artificial intelligent systems. Jetpack SDK is an end-to-end accelerated AI application comprising Jetson Linux with bootloader, Linux kernel, Ubuntu desktop environment, and a complete set of libraries for accelerating GPU computing. The Jetson family of modules uses the same NVIDIA CUDA-X software and supports cloud-native technologies like containerization and orchestration to build, deploy, and manage AI at the Edge. With Jetson, customers can accelerate all modern AI networks, easily roll out new features, and leverage the same software for different products and applications. In this project, NVIDIA Jetson Nano is used to configure and deploy the AI system via a Docker container.

3.1.1 Jetson Nano developer kit

The AI system is firstly developed, tested, and optimized on an NVIDIA GeForce RTX 2060 6GB VRAM [22] local machine and then tried deploying on a Jetson Nano edge device. NVIDIA GeForce RTX 2060 is a performance-segment graphics card launched on January 7th, 2019, with 1920 CUDA cores and 336GB/s of GDRR6 memory bandwidth. NVIDIA Jetson Nano [4] is a small, powerful computer for embedded AIoT applications. It brings incredible new capabilities to millions of tiny, power-efficient AI systems. Jetson Nano is also the perfect tool to start learning about AI and robotics in real-world settings, with ready-to-try projects and the support of an active and passionate developer community.

	Jetson Nano
AI Performance	472 GFLOPs
GPU	128-core NVIDIA Maxwell™ GPU @921MHz
CPU	Quad-Core Arm® Cortex®-A57 MPCore processor @1.43GHz
Memory	4 GB 64-bit LPDDR4 25.6GB/s
Storage	16 GB eMMC 5.1
Power	5W - 10W

Table 1: Jetson Nano hardware information

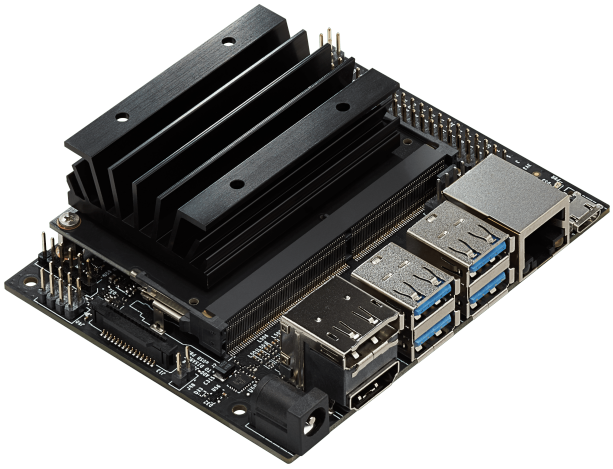

Figure 6: Jetson Nano visualization

Table 1 shows the general hardware information of Jetson Nano, and Figure 6 shows the detailed hardware visualization of Jetson Nano.

3.1.2 Camera

One of the core benefits of Jetson Nano is using the Linux kernel module Video Four Linux (version 2) (V4L2) to interface with a USB camera. V4L2 is part of Linux and is not unique to the Jetsons. This means that Jetson Nano can connect and use various USB cameras. The Logitech C920 webcam is chosen as the video-capturing device in this project. It features a 3Mpx sensor with a maximum resolution of 1080p at 30fps. It has a field of view of 78 degrees. This camera is chosen specifically because it is durable and readily available at my University for borrowing. In the future, the plan to standardize the protocol to work with camera streams could be proposed so that the system can connect with more types of cameras, such as IP or embedded cameras. Figure 7 Visualize the complete real-world illustration of the Logitech C920 webcam

	Area Under the Curve (AUC)
Resolution FPS	Full HD 1080p/30fps; HD 720p/30fps
Diagonal Field of View	78 degree
Autofocus	Yes
Auto Light Correction	RightLight 2
Noise Cancelling Mic(s)	2 omni-directional mics
Connection	USB-A plug-and-play
Cable length	1.5m
Tripod	Yes
Privacy Shutter	Yes

Table 2: Logitec C920 USB Camera specification

Figure 7: Logitec C920 USB camera

3.2 Coding template

Facebook Research is a group within Facebook that is focused on advancing the state of the art in AI. They work on various topics, including computer vision, natural language processing, and machine learning. They also have an active presence on GitHub, where they share their research and open-source projects. One of the novel open-source projects from Facebook is PySlowfast [8]. PySlowfast is a video understanding codebase from Facebook Artificial Intelligence Research (FAIR) that provides state-of-the-art video classification models with efficient training. PySlowfast aims to provide a high-performance, lightweight Pytorch codebase with state-of-the-art video backbones for video understanding research on different tasks (classification, detection, etc.). It is designed to support the rapid implementation and evaluation of novel video research ideas. In my thesis, PySlowFast is used and modified as the code-based structure template to quickly test various anomaly detection deep learning models and build a complete end-to-end AI system from collecting surveillance video inputs to outputting the final anomaly scores. Figure 8 shows an overview of open-source PySlowFast published on GitHub.

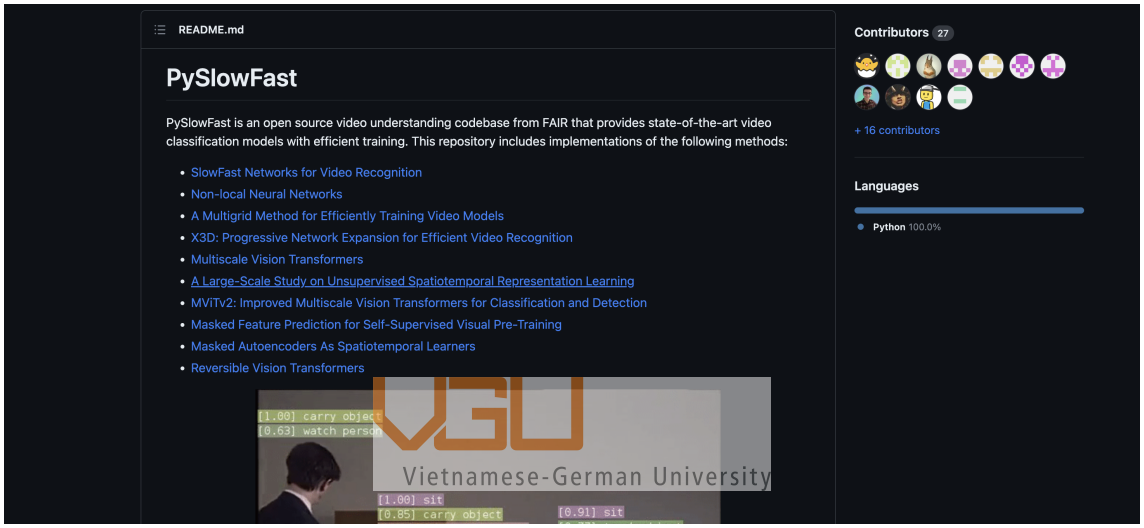

Figure 8: PySlowFast Github overview

PySlowfast coding template provides multiple mechanisms, which include logging, multi-processing, multi-threading, highly efficient data loader, multiple instance learning (MIL), etc. The template also comprises the video understanding X3D model already set up for inference and training. X3D is fine-tuned on the UCF-Crime dataset and performs the detection on some videos benchmark. Unfortunately, X3D did not show the potential result. Afterward, I removed all the video understanding model architecture and adapted the RTFM model into the coding template. The training mechanism of the PySlowfast template was also modified to apply the training code provided by the RTFM model. The other structures are preserved to take all the advantages of Facebook's production coding template. Consequently, the AI solution successfully integrates into the PySlowfast template and does the inference task.

3.3 AI solution

Anomaly detection in surveillance videos is a field of research that uses AI techniques to detect unusual events or behaviors in video footage automatically. One approach to this problem is to learn anomalies by exploiting both normal and abnormal videos. To avoid the time-consuming task of annotating anomalous segments or clips in training videos, researchers have proposed learning peculiarities through the deep multiple instance ranking framework by leveraging weakly labeled training videos called multiple instance learning (MIL). In the weakly labeled training approach, each regular or anomalous video is divided into many small video segments (clip). Each video segment is called an instance. Therefore, each video input is a bag of video segments (clips). A deep learning model is then automatically learned from the anomalous bags and ranked the video segments based on their predicted anomaly score. Below, totally three different anomaly detection models are introduced, and the final chosen AI solution for the system.

3.3.1 AI solutions research methodologies

Paperwithcode [1] is a website created by Meta Facebook that provides a collection of the latest machine learning research papers along with their corresponding code implementations. It covers various topics, including computer vision, natural language processing, medicine, time series, graphs, speech, and more. Paperwithcode allows us to browse the state-of-the-art machine learning paper based on the public dataset or specific problems with the related code. By searching for the UCF-Crime dataset, the primary dataset for the situation we are solving on this platform, we have an overview of all the state-of-the-art AI solutions ranked by the evaluation metric provided in each paper. The figure shows an overview of the Paperwithcode page searched by the UCF-Crime dataset.

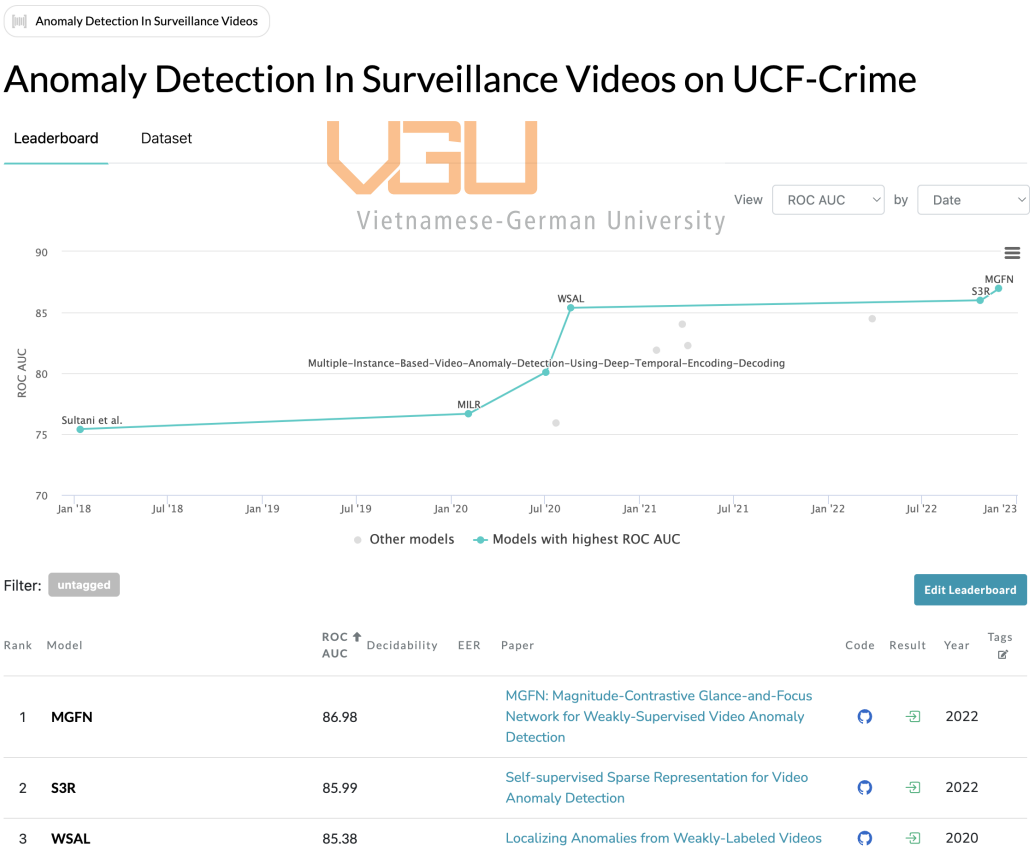

Figure 9: Overview of Paperwithcode platform searched by UCF-Crime dataset

3.3.2 Rethinking Spatio Temporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification (S3D)

S3D is a model resulting from replacing 3D convolutions of the state-of-the-art model I3D [cite] with 2D and 1D convolutions. This new model is claimed to have the best trade-off between complexity and accuracy. Staining Xie [20] findings show the efficient performance of a wide range of datasets. An efficient video classification system is invented by combining the system with other cost-effective designs, such as separable spatial/temporal convolution and feature gating.

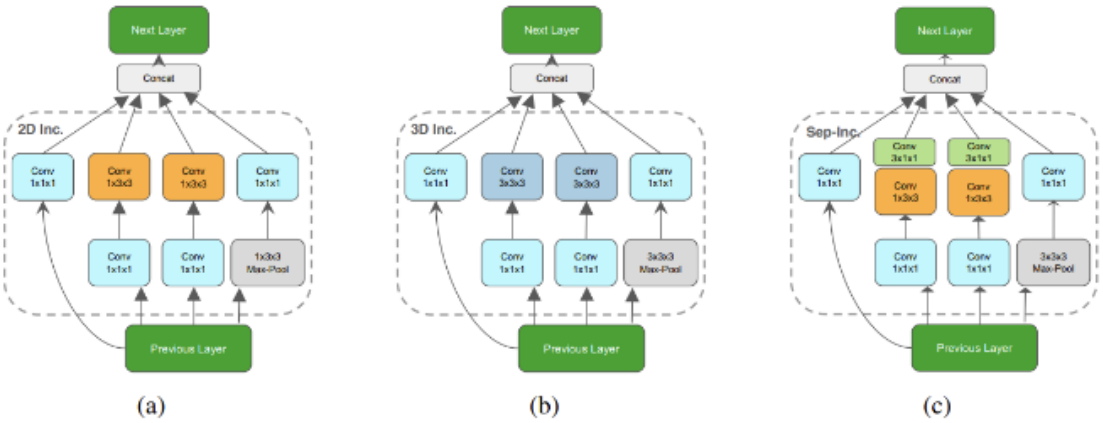

Figure 10: (a) 2D Inception block; (b) 3D Inception block; (c) 3D temporal separable Inception block used in S3D networks from [20]

The proposed model shows a massive drop in complexity and better performance. This system has achieved competitive results on several action classification benchmarks (Kinetics, Something-something, UCF101, and HMDB) and two action detection benchmarks (JHMDB and UCF101-24). S3D-G reaches outstanding results on the UCF-101 dataset with 96.8% accuracy compared to 95.6% from I3D, 85.8% from Res3D, and only 82.3% from C3D methodology. The present method also achieves the highest result on the HMDB-51 dataset as well with 75.9% accuracy. However, due to the lack of training code from the authors, it is time-consuming to reproduce the results.

Model	Inputs	Pre-train	UCF-101	HMDB-51
C3D	RGB	Sports-1M	82.3	51.6
Res3D	RGB	Sports-1M	85.8	54.9
I3D	RGB	ImNet+Kinetics	95.6	74.8
S3D-G	RGB	ImNet+Kinetics	96.8	75.9

Table 3: Results of various methods on anomaly action classification on the UCF-101 and HMDB-51 datasets from [20]

3.3.3 Expanding Architectures for Efficient Video Recognition (X3D)

Christoph Feichtenhofer [6] proposes a novel video recognition architecture called X3D, which stands for Expanding Architectures for Efficient Video Recognition. The motivation of X3D is to improve the efficiency of video recognition models while maintaining accuracy. X3D is based on expanding the receptive field of a video recognition model by using a cascade of dilated convolutions. This allows X3D to capture long-range dependencies in videos, which is vital for tasks such as action recognition and video classification. X3D is also designed to be more efficient than 3D CNNs by using a smaller number of parameters and a simpler architecture. The authors also propose a new training method for X3D based on distillation. Distillation is a technique that can be used to transfer knowledge from a large, complex model to a smaller, simpler model. Moreover, Stochastic depth is used to regularize X3D and prevent overfitting. X3D was evaluated on the Kinetics-400 dataset and the UCF-101 dataset.

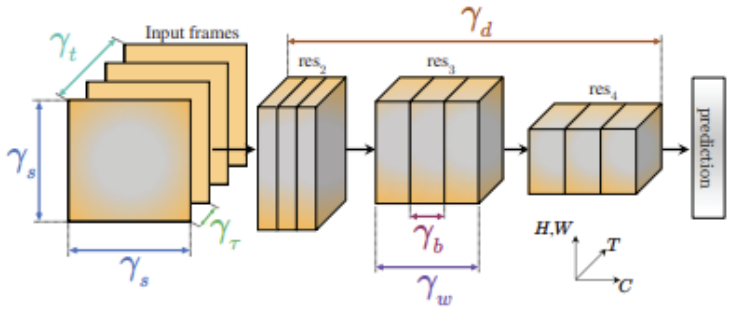

Figure 11: X3D networks progressively expand a 2D network across the following axes: Temporal duration, frame rate, spatial resolution, width, bottleneck width, and depth

On the Kinetics-400 dataset and UCF-101 dataset, X3D outperformed the state-of-the-art methods by a significant margin. For example, on the Kinetics-400 dataset, X3D achieved an accuracy of 85.1%, which is 2.2% higher than the previous best result. Unfortunately, based on the author's findings, X3D is designed for the action recognition task of a specific object, and it uses fully connected as the last layer for classification. While anomaly detection requires a comprehensive observance of different objects, including the main and background ones, to successfully perform the task. Therefore, X3D can not be the final choice for the AI solution. Figure 10 is the general architecture of X3D taken from [6]

Model	pre	top-1	top-5	test	GFLOPs x Views	Param
I3D	ImageNet	71.1	90.3	80.2	108 × N/A	12M
Two-Stream I3D	ImageNet	75.5	92.0	82.8	216 × N/A	25M
Nonlocal R50	ImageNet	76.5	92.6		282 × 30	35.3M
SlowFast 4×16, R50	ImageNet	75.6	92.1		36.1 × 30	34.4M
X3D-M	ImageNet	76.0	92.3	82.9	6.2 × 30	3.8M
X3D-L	ImageNet	77.5	92.9	83.8	24.8 × 30	6.1M
X3D-XL	ImageNet	79.1	93.9	85.3	48.4 × 30	11.0M
X3D-XXL	ImageNet	80.4	94.6	86.7	194.1 × 30	20.3M

Table 4: Comparison of X3D to the state-of-the-art on K400-val test from [6]

3.3.4 Weakly-supervised video anomaly detection with robust temporal feature magnitude learning (RTFM)

Yu Tian [27] proposes a weakly-supervised video anomaly detection method called Robust Temporal Feature Magnitude Learning (RTFM). RTFM learns to detect anomalies by robustly measuring the magnitude of temporal features. The technique can learn from weakly-labeled videos, making it more practical than previous methods that require fully-labeled videos. The paper "Robust Temporal Feature Magnitude Learning" (RTFM) aims to address the problem of temporal feature magnitude estimation in the presence of noise. The authors argue that existing temporal feature magnitude estimation methods are not robust to noise and can lead to inaccurate results. They propose a new approach, RTFM, designed to be more robust to noise. RTFM is based on using a temporal filter to smooth the temporal features. The authors show that RTFM is more accurate than existing methods for temporal feature magnitude estimation in the presence of noise.

Supervision	Method	Feature	AUC (%)
Unsupervised	SVM Baseline	-	50.00
	Conv-AE [15]	-	50.60
	Sohrab et al. [56]	-	58.50
	Lu et al. [30]	C3D RGB	65.51
	BODS [64]	I3D RGB	68.26
	GODS [64]	I3D RGB	70.46
Weakly Supervised	Sultani et al. [57]	C3D RGB	75.41
	Sultani et al.* [57]	I3D RGB	77.92
	Zhang et al. [75]	C3D RGB	78.66
	Motion-Aware [81]	PWC Flow	79.00
	GCN-Anomaly [79]	C3D RGB	81.08
	GCN-Anomaly [79]	TSN Flow	78.08
	GCN-Anomaly [79]	TSN RGB	82.12
	Wu et al. [67]	I3D RGB	82.44
	Ours	C3D RGB	**83.28**
	Ours	I3D RGB	**84.30**

Figure 12: RTFM frame-level evaluation compared other algorithms based on Area under the curve (AUC) metric

The authors evaluate their method on several benchmark datasets and show that it outperforms existing methods for weakly-supervised video anomaly detection. Specifically, their way achieves state-of-the-art performance on the ShanghaiTech and UCF-Crime datasets. They also show that their approach is more efficient than existing weakly-supervised video anomaly detection methods. Therefore, the RTFM anomaly detection model is the final AI solution choice for the AI system in my thesis.

3.3.5 Final chosen AI solution

The final AI solution is based mainly on two methodologies: ResNet 50 I3D feature extractor and RTFM anomaly detector. ResNet 50 Inception 3D (I3D) was invented by Joao Carreira [12] and was initially designed for human action recognition tasks. The model is pre-trained on Kinetics 400 dataset, and some modifications, such as removing head architecture, are applied to construct an efficient video feature extractor. Weakly-supervised Video Anomaly Detection with Robust Temporal Feature Magnitude Learning (RTFM) is the final choice for the anomaly detection model in the system for two main reasons. Firstly, Yu Tian [27] provides the codes to reproduce the result reported by the authors. The code achieves 267 stars on Github, the highest star compared to other models, and successfully performs the inference and training processes. Secondly, based on the paperwithcode reference, RTFM ranks highly on anomaly-related datasets such as UCF-Crime, UCSD Peds2, and ShanghaiTech Weakly Supervised datasets. Figure 13 shows the step-by-step of the whole AI pipeline in this system.

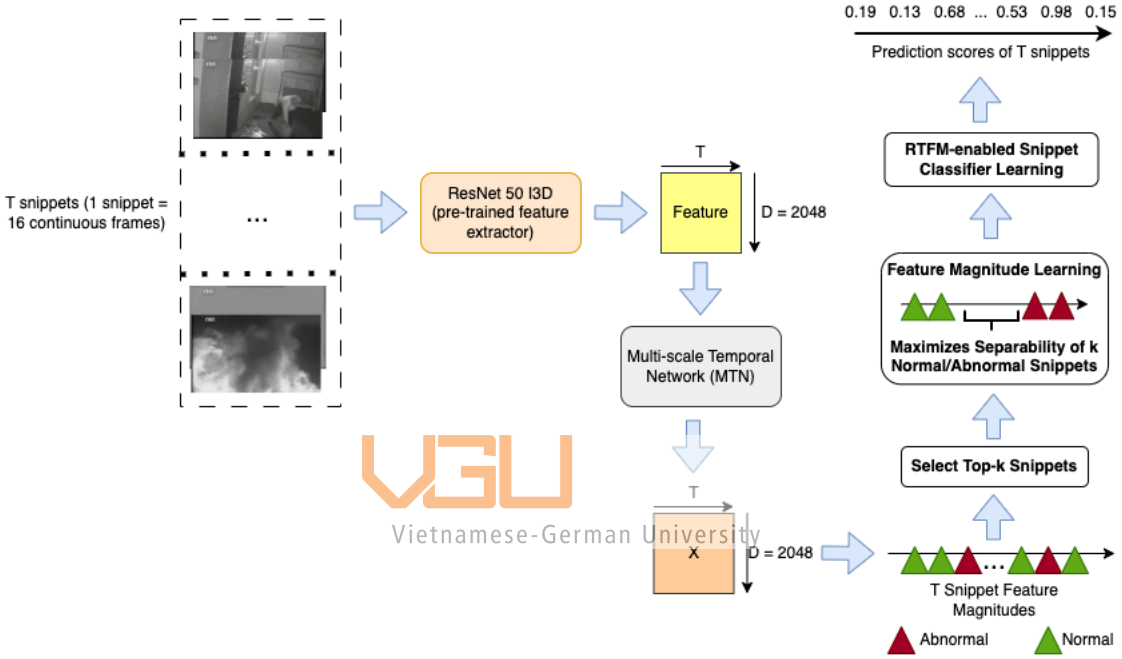

Figure 13: Final AI solution for system

From the camera frames stream, the system divides those stream into 10s video each and input them into the feature extractor with feature shape output as 2048. More importantly, it is impossible to input each 16-frame video segment (clip) into the system and perform the detection because RTFM utilizes the top-k Multiple Instance Learning (MIL), which requires a bag of many continuous 16-frame video segments together to complete the detection of the abnormal event successfully. After receiving the output features, they are inputted into a Multi-scale Temporal Network (MTN) to capture the multi-resolution local and global temporal dependencies between video segments. Traditionally, dilated convolution is applied in the spatial domain to expand the receptive field without losing resolution. Similarly, MTN uses a pyramid of dilated convolution over the temporal dimension. It is crucial to capture the multi-scale temporal dependencies of neighboring video segments for anomaly detection [27]. The top k video segments with the highest MTN result are pushed into a feature magnitude learning algorithm. The algorithm is trained to maximize the separability between normal and abnormal segments. Finally, a binary cross entropy-based classifier loss function proposed in RTFM-enabled Snippet Classifier Learning is put on the table to output the final anomaly prediction probability for each video segment.

3.4 Datasets for anomaly detection from Surveillance Videos problem

There are several datasets available for anomaly detection on surveillance cameras. For example, UCF-Crime is the largest available dataset for automatic visual analysis of anomalies and consists of real-world crime scenes of various categories. A subset of this dataset, HR-Crime, is suitable for human-related anomaly detection tasks. The ShanghaiTech dataset is a large-scale crowd-counting dataset. It consists of 1198 annotated crowd images with nearly 330,165 annotated people. This project uses UCF-Crime and UIT VNAnomal datasets for the testing, fine-tuning, and evaluation.

3.4.1 UCF-Crime dataset

The UCF-Crime dataset is proposed in "Real-World Anomaly Detection in Surveillance Videos" 2018 by Waqas Sultani [25]. It consists of long untrimmed surveillance videos which cover 13 real-world anomalies, including Abuse, Arrest, Arson, Assault, Road Accident, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing, Shoplifting, and Vandalism. The dataset includes nearly 1900 normal and abnormal Youtube videos, which stand for 128 hours in total. The table below showcases the comparison of UCF-Crime with other datasets with the same purpose.

	# of videos	Average frames	Dataset length	Example anomalies
UCSD Ped1 [27]	70	201	5 min	Bikers, small carts, walking across walkways
UCSD Ped2 [27]	28	163	5 min	Bikers, small carts, walking across walkways
Subway Entrance [3]	1	121,749	1.5 hours	Wrong direction, No payment
Subwa Exit [3]	1	64,901	1.5 hours	Wrong direction, No payment
Avenue [28]	37	839	30 min	Run, throw, new object
UMN [2]	5	1290	5 min	Run
BOSS [1]	12	4052	27 min	Harass, Disease, Panic
Ours	**1900**	**7247**	**128 hours**	**Abuse, arrest, arson, assault, accident, burglary, fighting, robbery**

Figure 14: A comparison of anomaly datasets. The dataset contains a larger number of longer surveillance videos with more realistic anomalies from [25]

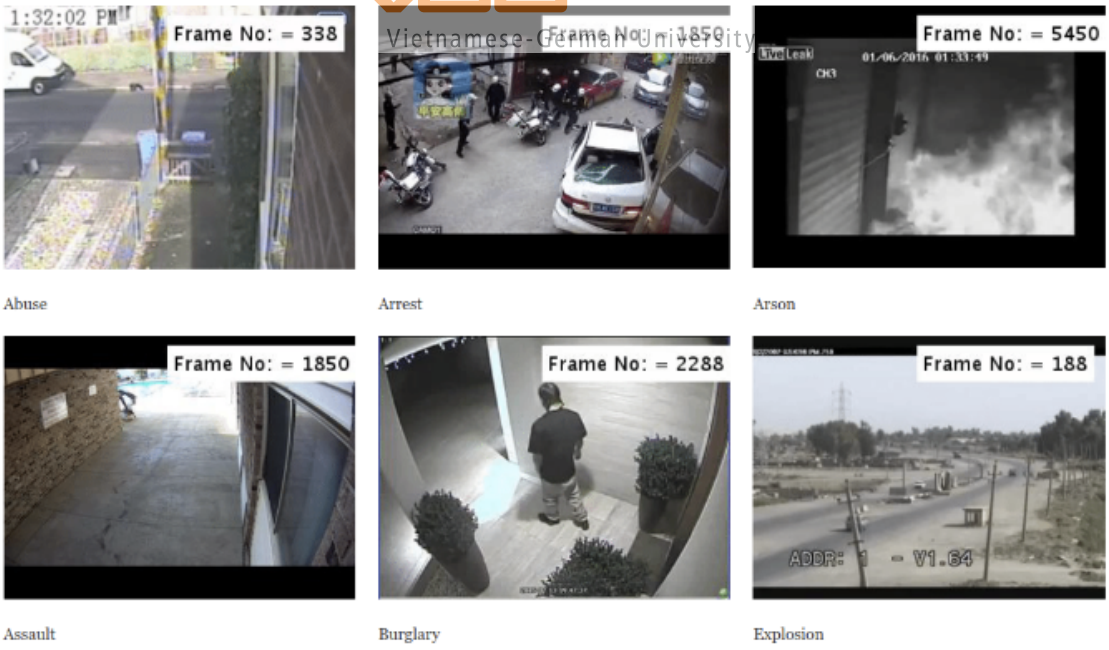

Figure 15: Video samples of anomalous events from UCF-Crime dataset.

3.4.2 UIT VNAnomaly dataset in Viet Nam

The UIT-Anomaly dataset is a set of anomalous videos collected in Vietnam by Dung T.T. Vo [24] from the University of Information Technology research group. The dataset is used for the Anomaly detection task, which identifies patterns in data that do not conform to expected behavior. The total duration of the dataset is 200 minutes. It consists of 89 training videos and 96 evaluating videos, including real-world anomalies in Vietnamese streets. The anomaly types contain six common human-related abnormalities in the road of Vietnam, including fighting, assault, vandalism, robbery, dog thief, and traffic accident. Figure 16 visualizes some video samples from the VNAnomaly dataset with different types of anomaly scenes.

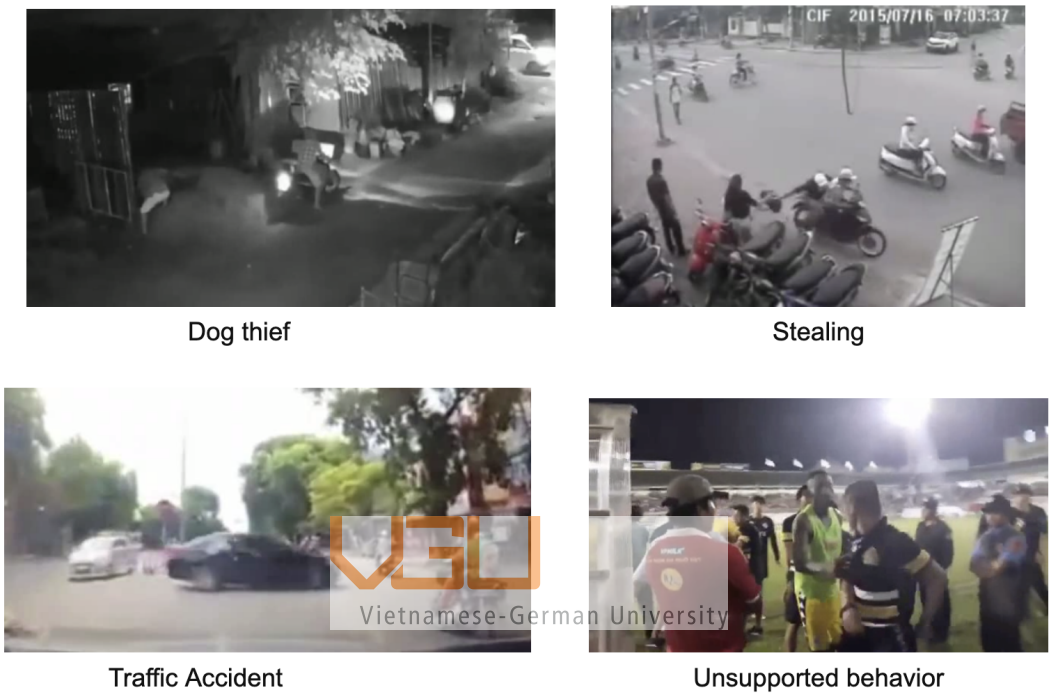

Figure 16: Video samples of anomalous events from the VNAnomaly dataset.

The dataset surpasses existing unsupervised datasets from the following three aspects

- VNAnomaly is one of the first unsupervised datasets captured in a Vietnamese scene.
- Larger data volume: VNAnomaly has 578,609 training frames and 75,214 evaluating frames, bigger than existing unsupervised benchmark datasets.
- Higher scene diversity: it contains 36 scenes with different aspects such as camera angles and times of the day.

3.5 Edge AI Optimization Techniques

In recent years, AI has appeared in every technology field, from home electronics to complex simulation experiments for protein structure. However, high-performance machine learning or deep learning requires a significant computational capability to handle difficult training and inference methodologies and large datasets. Applying cloud servers can provide powerful computational capabilities to meet this demand for computation. While cloud-based systems have proven to be a suitable platform for some AI applications, they have limitations that may prevent their adoption for all AI applications. For example, in autonomous driving, the robustness and latency of the connection to the server can seriously impact vehicle safety due to the time of the collision. Uploading personal information or street-view videos to the cloud also raises privacy concerns. Additionally, internet connectivity is not always available everywhere. These issues have led to the need for AI computation to be performed on edge devices (Edge AI).

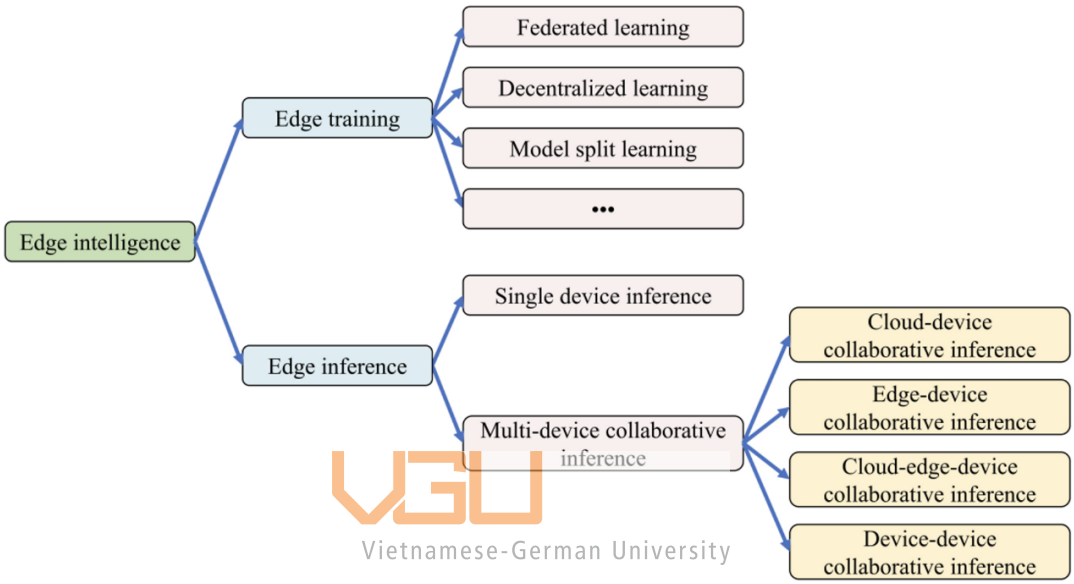

Figure 17: Current Egde AI Methods

Researchers following the edge AI direction have proposed various optimization algorithms regarding inference speed, storage usage reduction, and energy conservation. Some popular methods should be listed, such as Knowledge Distillation [9], Model Quantization, Federated Learning, Model Pruning, etc. In this project, the TensorRT platform created by NVIDIA is applied.

3.5.1 TensorRT

TensorRT [5] is an SDK provided by NVIDIA that allows models to run efficiently on NVIDIA hardware, which meets this project's requirements. It makes models lighter and faster by using optimization techniques, namely precision calibration, layer and tensor fusion, kernel auto-tuning, dynamic tensor memory, and multiple stream execution.

TensorRT provides precision calibration, which converts parameters and activation from high precision (FP32) to low precision (FP16, INT8). This helps reduce the latency and model size. Layer and tensor fusion merges nodes in a kernel vertically or horizontally(or both), which reduces the overhead and the cost of reading and writing the tensor data for each layer. Kernel auto-tuning selects the best layers, algorithms, and optimal batch size based on the target GPU platform. Dynamic tensor memory allocates memory to the tensor only for the duration of its usage. Multiple stream execution allows numerous input streams to be processed in parallel.

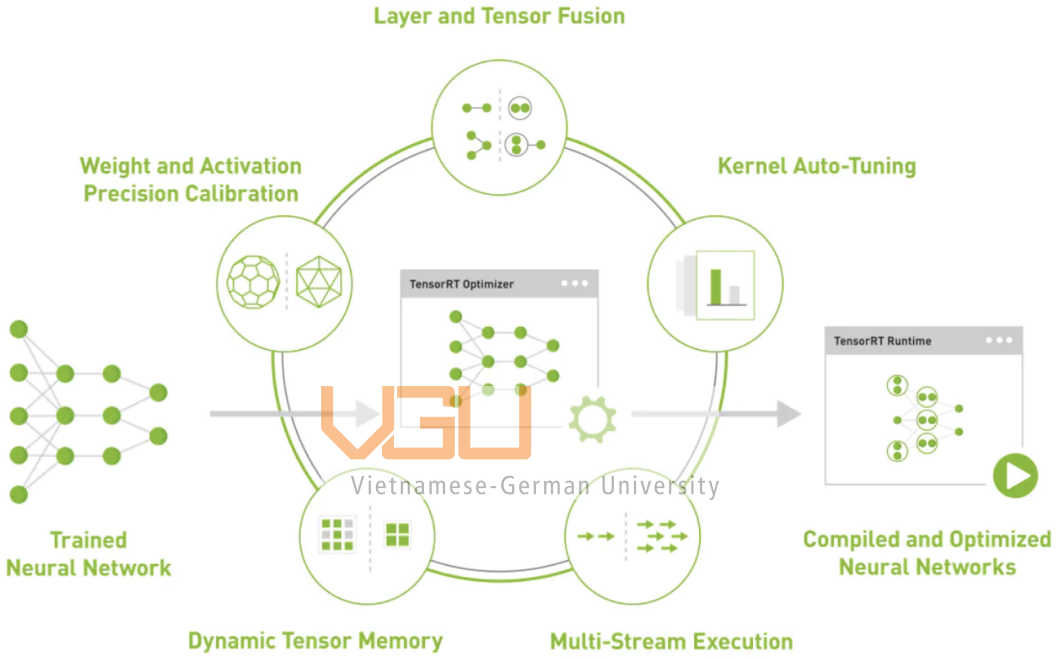

Figure 18: TensorRT Platform from [5]

Based on the functionalities the TensorRT platform provides, various benefits relating to inference speed, easier deployment, and energy conservation are brought to the table. Firstly, NVIDIA TensorRT-based application performs up to 36X faster than CPU-only platforms during the inference, enabling the optimization of neural network models trained on all major frameworks. Secondly, TensorRT, built on NVIDIA CUDA parallel computing model, enables optimization using quantization, layer and tensor fusion, kernel tuning, and others on NVIDIA GPUs. Thirdly, it accelerates every workload by utilizing quantization-aware training INT8, post-training quantization, and floating point 16 (FP16) optimization to deploy deep learning inference applications. Reduced-precision inference significantly reduces latency, essential for many real-time services and autonomous and embedded applications.

3.6 Deployment on Edge Devices

Deci.ai is a Deep Learning Development Platform company that simplifies and accelerates the development of computer vision and NLP applications with advanced tools to build, optimize, and deploy highly accurate and efficient models. The company aims to empower AI developers with powerful tools for creating innovative AI-based solutions. Deci was founded in 2019 by world-recognized experts in AI, and their team consists of top AI researchers with advanced academic degrees from prestigious universities. In this project, I take advantage of the best practices of step-by-step deployment created by Deci.ai to deploy the AI system into Jetson hardware edge devices.

3.6.1 Deci.ai best-practices AI model deployments on Edge Devices

Based on an engineering post about the best practices for Deep Learning Deployment on NVIDIA Jetson Devices [21], many valuable tips with step-by-step Jetson device configurations are shown to optimize the deployment process. Firstly, the Jetson devices should be configured by modifying Jetson mode (nvpmodel) and Jetson clocks to enable the Jetson's full potential. Secondly, some edge AI techniques, such as TensorRT FP16, are applied to optimize the inference runtime of the end-to-end AI system on Jetson devices. Thirdly, after having the result from running the whole system, production parameters such as buffer size are calibrated to obtain the optimal result. Finally, the NGC NVIDIA l4t Pytorch docker image should be used as a base image for the code system to run directly on Jetson devices.

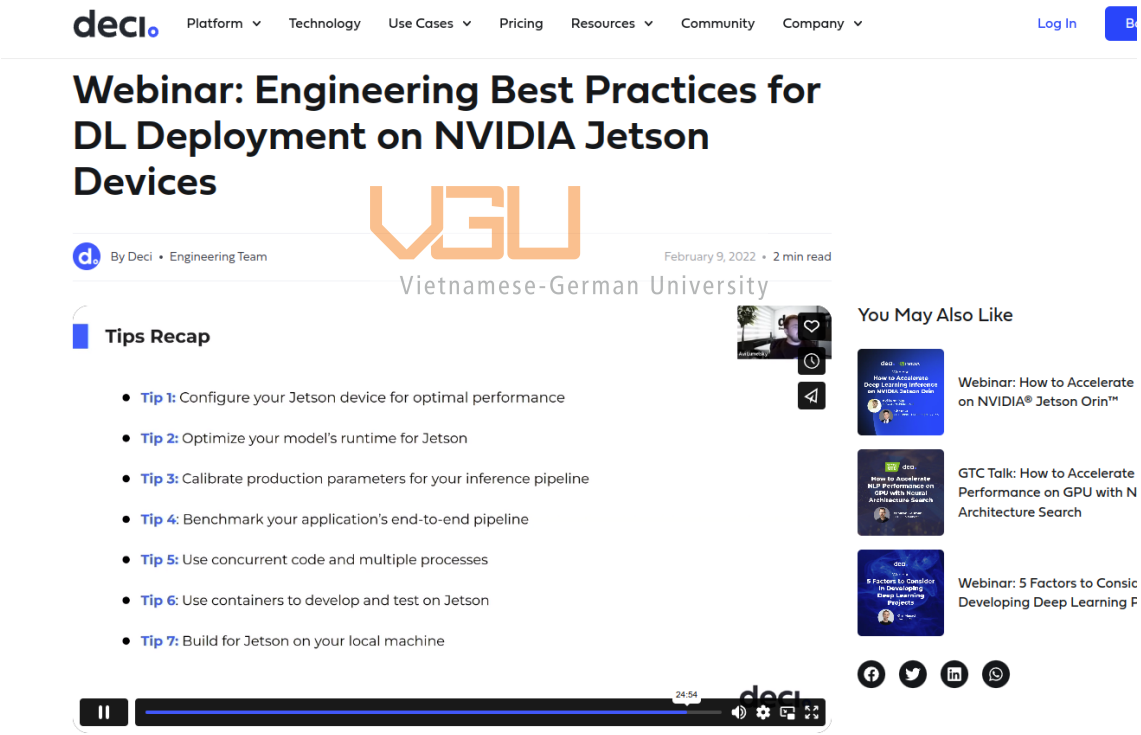

Figure 19: Overview of Deci.ai deployment

3.6.2 Application of Docker in Jetson Edge Devices

Docker is an open platform for developing, shipping and running applications. It enables us to separate the applications from the infrastructure so I can deliver software quickly. With Docker, I can manage your infrastructure like I manage the applications. Docker provides the ability to package and run an application in a loosely isolated environment called a container. The containers save users the hassle of troubleshooting possible compatibility issues between systems. The isolation and security allow us to run many containers simultaneously on a given host.

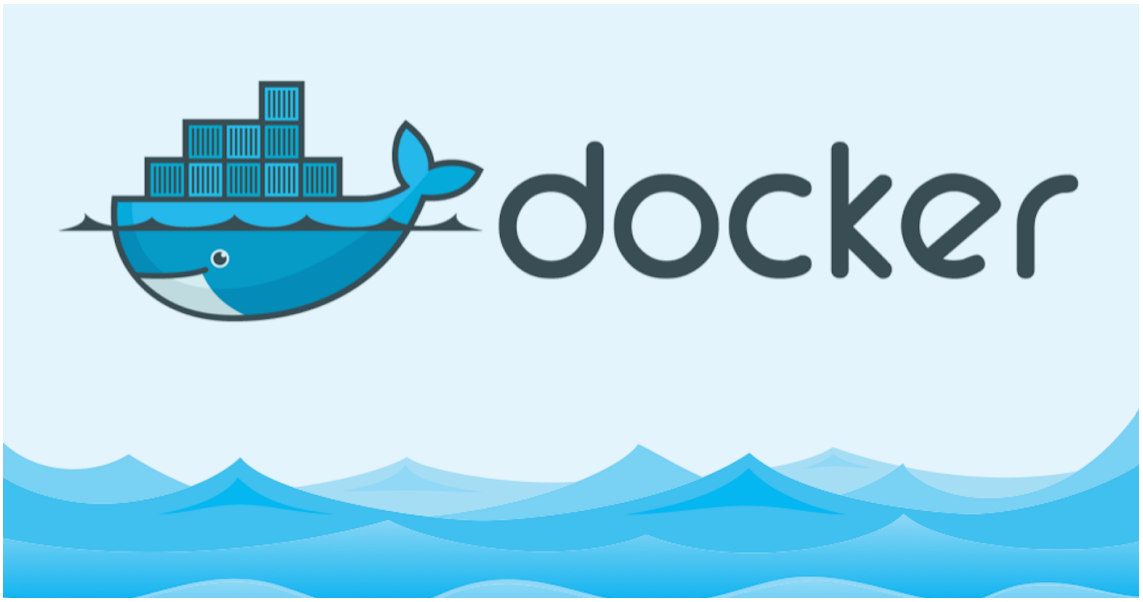

Figure 20: Docker

In edge computing, Docker can deploy applications on edge devices close to the data source, such as sensors or IoT devices. This project's Docker code is written based on the built-in Nvidia l4t images explicitly created to run the Pytorch CUDA framework on a Jetson edge device. By deploying the AI system on edge devices using Docker, multiple advantages of containerization technology are listed below.

- Reduced Latency: Docker containers running on edge devices can provide near-real-time processing of data
- Improved Security: Provide a high degree of isolation which help to improve security
- Portability: Docker containers can be run on any machine that supports Docker
- Resource Efficiency: Docker containers are lightweight, which makes them an ideal choice for edge computing environments

3.6.3 NGC NVIDIA l4t Docker image for Jetson Edge devices

In this project, the NGC l4t Pytorch [18] docker image created by NVIDIA is used as the base image for the deployment process. NVIDIA L4T PyTorch is a docker image that contains PyTorch, Torchvision, and Torchaudio already installed in a Python 3 environment. The containers support the following releases of Jetpack for Jetson Nano, TX1/TX2, Xavier NX, AGX Xavier, AGX Orin, and Orin NX. These images are designed, tested, and developed mainly for Jetson edge devices to advance the AI deployment progress.

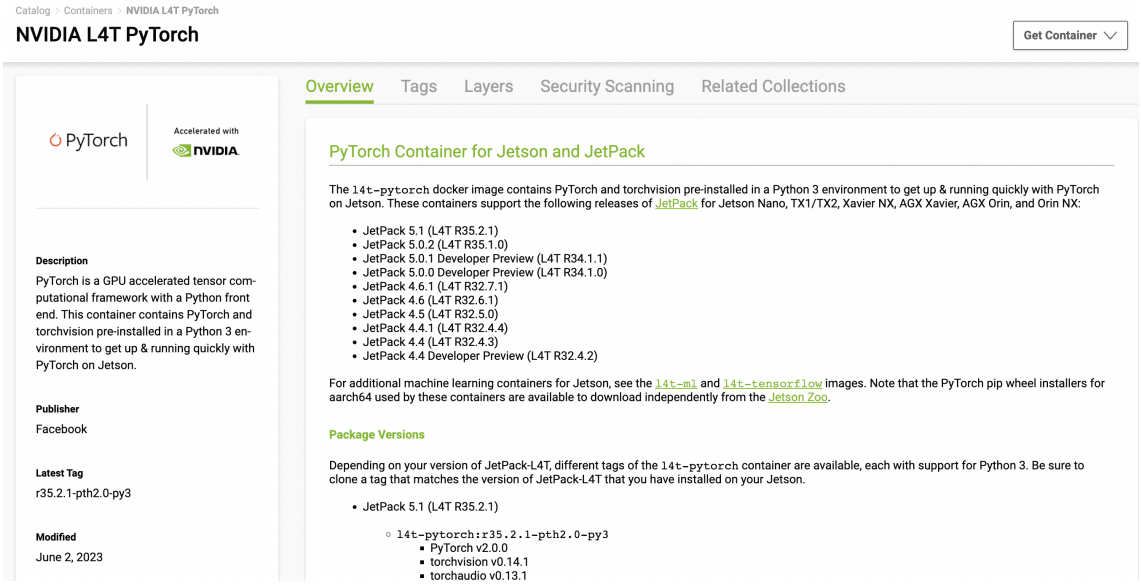

Figure 21: NVIDIA L4T Pytorch docker images for Jetson edge devices

The NVIDIA l4t Docker image supports the CUDA Pytorch framework, which can activate and utilize the GPU on Jetson edge devices. GPU is critical to run a computer vision deep learning model directly at edge devices. It enables the parallel computing machine with the purpose of processing high dimensional arrays simultaneously instead of sequential execution traditionally. The l4t image also includes essential libraries such as onnx, TensorRT, and OpenCV, which are indispensable for video collection from attached cameras or some edge AI techniques such as TensorRT.

4 Experiments

4.1 Data processing pipeline

Dataset UCF Crime and UIT Vietnam anomaly, after being downloaded, are resized firstly to scale 256 height. Because the anomaly detection model input shape is scale 256 height, it would skip the resize step in data processing and exponentially increase the training and evaluation speed. Then, based on the CSV format metadata requirement from the anomaly detection model, a Python code collects all the paths to training and evaluating videos with the associated labels. Finally, I would have a dataset including UCF-Crime and UIT Vietnam anomaly with the correct format for the following testing and fine-tuning process.

Regarding the data process pipeline, RTFM's chosen anomaly detection model requires strict step-by-step pre-processing techniques with the correct order and input parameter specification. The pipeline converts the input data into torch Tensor datatype, then resizes those tensors to test shape 256. After that, it applies a 10-crop data augmentation technique which is quite popular in computer vision. The method involves randomly taking ten different crops of an image and using them as separate inputs to the model. The crops are taken from the four corners, center, and their mirrored versions. Finally, those crop images are stacked together, and the normalization algorithm with provided mean and standard deviation is applied to finalize the data processing pipeline.

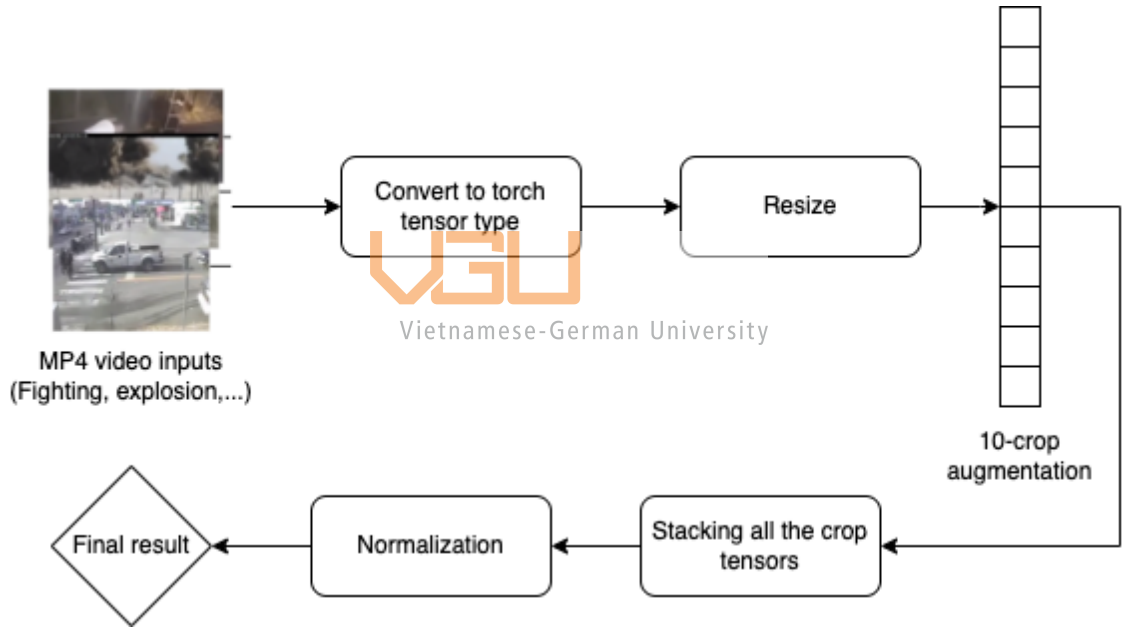

Figure 22: Data processing pipeline of the AI system

```
# NOTE: Should apply transform in each 16-frames image group (as 1 clip)
self.spatial_transform = transforms.Compose([
    # Convert to Tensor
    transforms.ToTensor(),
    # Resize
    transforms.Resize(self.cfg.DATA.TEST_CROP_SIZE),
    # 10-crop augmentation
    transforms.TenCrop(224),
    # Convert to Tensor
    lambda crops: torch.stack([crop * 255 for crop in crops]),
    # Normalization
    transforms.Normalize(self.cfg.DATA.MEAN, self.cfg.DATA.STD),
])
```

Figure 23: Data processing pipeline Pytorch coding

4.2 AI experiments

Experiments on different Artificial Intelligent solutions and datasets preparation take much time and effort. Each AI model requires a specific dataset input format to perform the inference and training process successfully. Because each model comes from different open-source project code with different data loaders and code structures from world-class AI developers. For example, the code from the X3D model comes from Slowfast open-source project created and developed by Facebook. The code includes the iterator and the design patterns, such as the factory pattern, to organize the tasks in the code more efficiently. Moreover, it also consists of the code to support multi-threading and multi-processing to perform some proper functions simultaneously. Understanding and modifying those world-class coding projects is challenging but still benefits me as a developer. The Area Under the ROC Curve (AUC) is the metric for the evaluation process in this project. The metric measures the ability of a binary classifier to distinguish between classes and is used as a summary of the ROC curve1. The higher the AUC, the better the model's performance at distinguishing between the positive and negative classes

After downloading from the internet, the datasets are organized into the required folder structure to match the training design from the provided code. The AI solution utilizes the feature extractor's weight from the open-source Github repository with 2048 as the feature shape. The extractor is pre-trained on the Kinetic 400 dataset. Therefore, non of the training activities are required for the feature extraction part. Regarding the anomaly detection deep learning model, the fine-tuning process is applied to both UCF-Crime and VNAnomaly datasets. Fine-tuning is a process that takes a model that has been trained for one given task and then tunes or tweaks it to make it perform a second similar task. There is a slight modification on the node number of the X3D model head, as it is a fully-connected layer. There is no modification to the RTFM model because the model is designed for crime-scene anomaly detection solely. I tuned the anomaly detection model with its paper's provided loss function and optimization techniques. The learning rate is set to 0.001 with 100 epochs. Figure 24 opens a visualization overview of fine-tuning process.

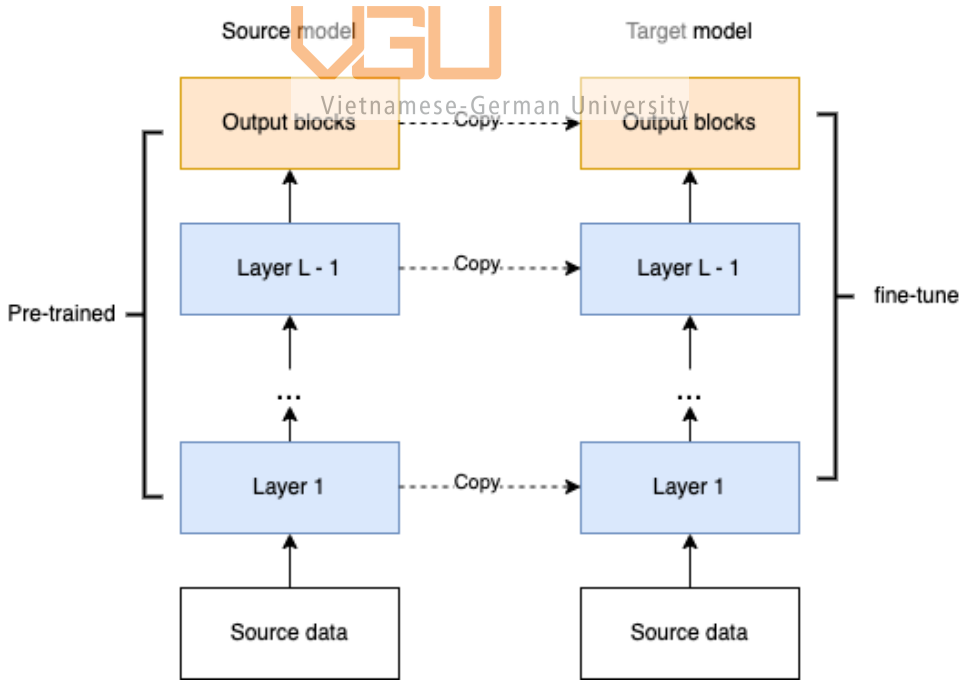

Figure 24: Fine-tuning process visualization

4.2.1 S3D test experiments

Although on paper, S3D shows remarkable performance and accuracy. However, the code of the S3D paper provided by the author has the inference code only and without a training process. Therefore, reproducing the result from the paper should take tremendous time and effort, while the risk of getting unsatisfied results is huge. As a result, S3D is abandoned, and I should continue to the next chosen AI solutions

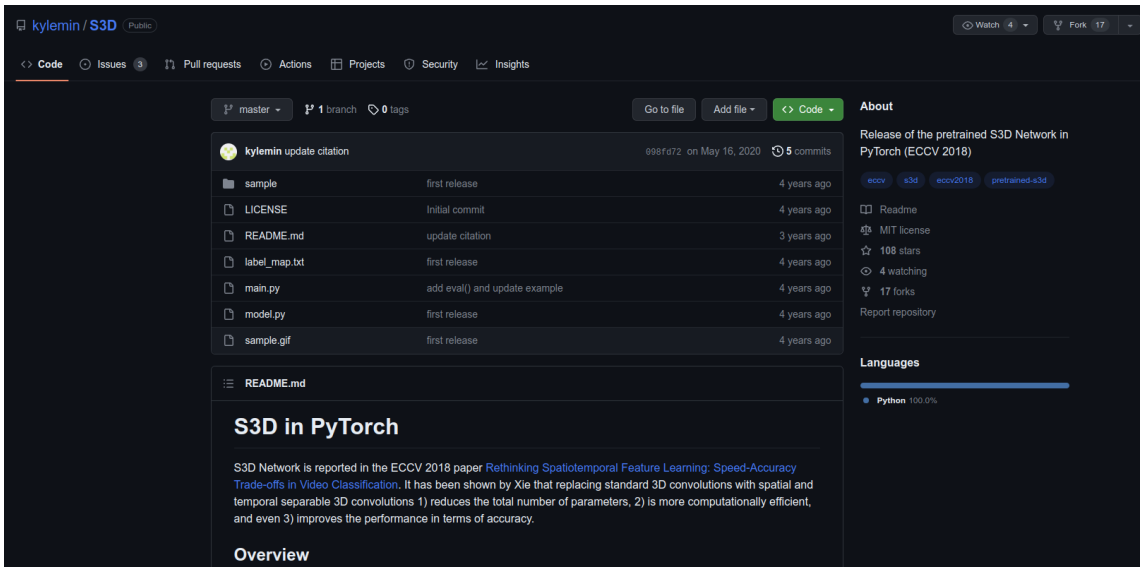

Figure 25: Overview of S3D Github

4.2.2 X3D test experiments

Expanding architecture for efficient Video recognition (X3D) is an exciting direction to test. X3D is a family of efficient video networks that progressively extend a tiny 2D image classification architecture along multiple network axes in space, time, width, and depth. The model achieves the state-of-the-art result on the Kinetic 400 [13] dataset regarding the accuracy and inference speed. I chose five videos across different crime scene types as the benchmark to test the X3D. The UCF Crime dataset is also used to fine-tune the X3D model. Unfortunately, after a fine-tuning step and test afterward, the model failed to detect and separate the anomaly frame from the other normal frames. Hypothetically, based on the demo provided by the author, X3D is designed to recognize the actions of multiple specific objects or humans instead of understanding the movement between humans and objects or humans with humans. Therefore, X3D is not chosen as the final AI solution.

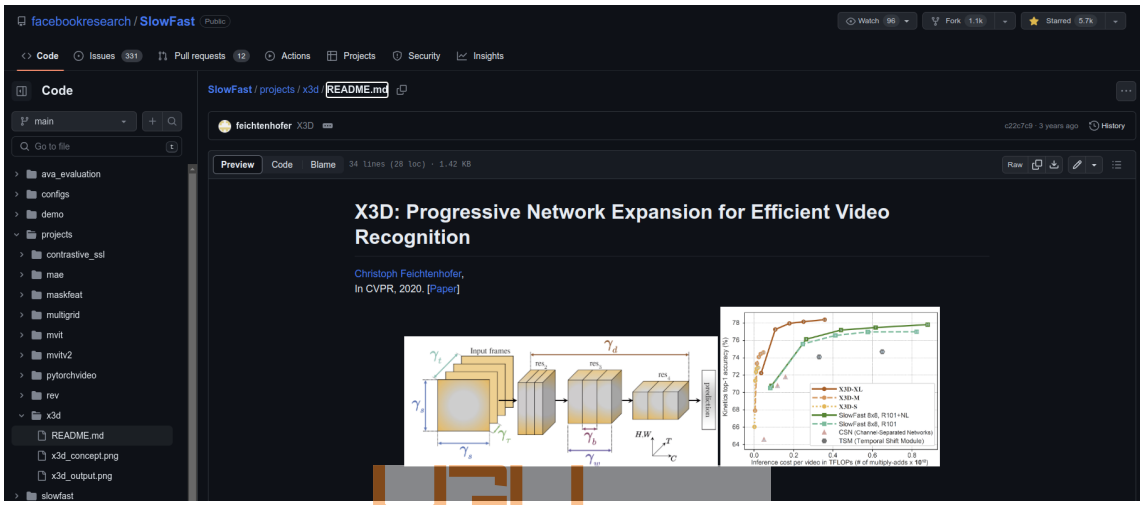

Figure 26: Overview of X3D Github

	Area Under the Curve (AUC)
X3D-L before fine-tune UCF-Crime	0.4892
X3D-L after fine-tune UCF-Crime	0.6132
X3D-L before fine-tune UIT VNAnomaly	0.3241
X3D-L after fine-tune UIT VNAnomaly	0.4695

Table 5: X3D-L evaluation result on UCF-Crime and UIT VNAnomaly dataset

X3D is the most lightweight and fastest model in terms of action recognition tasks and is being announced to deploy and applied widely on Samsung smartphones successfully. Unfortunately, the model shows a disappointing result after fine-tuning on both UCF-Crime and VNAnomaly datasets and quickly becomes vulnerable when testing on the video benchmarks.

4.2.3 RTFM test experiments

Weakly-supervised Video Anomaly Detection with Robust Temporal Feature Magnitude Learning (RTFM) is the most promising solution I can consider now. According to [27], RTFM achieves state-of-the-art results on multiple datasets, especially reaching the 3rd rank on the UCF-Crime dataset. Moreover, RTFM Github, provided by the author, gained 262 stars, which showcases a vast novelty and compliments from the AI developer community. For these reasons, I decided to test and fine-tune the RTFM on UCF-Crime to reproduce the result from the paper. Table 2 shows the evaluation result of the RTFM model before and after fine-tuning the UCF-Crime dataset using Area Under the ROC Curve (AUC) metric.

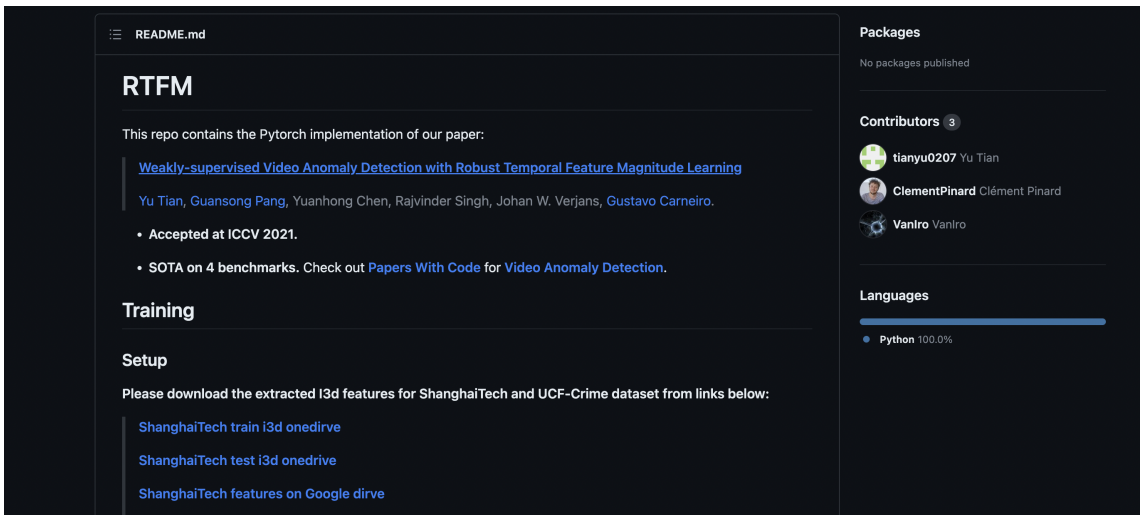

Figure 27: Overview of RTFM Github

	Area Under the Curve (AUC)
RTFM before fine-tuning UCF-Crime	0.8313
RTFM after fine-tuning UCF-Crime	0.8439
RTFM before fine-tuning UIT VNAnomaly	0.826
RTFM after fine-tuning UIT VNAnomaly	0.884

Table 6: RTFM evaluation result on UCF-Crime and UIT VNAnomaly dataset

Based on the evaluation result of RTFM on the UCF-Crime and UIT datasets, RTFM has already scored a significant impact on both datasets. Note that data processing is vital to enable the full potential and reproduce the RTFM result in the author's declaration. After the fine-tuning process, only a slight improvement resulted according to the AUC score. The area under the curve (AUC) is the evaluation metric representing the probability that a randomly selected positive example will be ranked higher than a randomly chosen negative example. The AUC ranges from 0 to 1, with an AUC of 0.5 indicating that the model is no better than random guessing and an AUC of 1 showing perfect classification.

4.3 Torch-TensorRT experiments

Edge AI algorithms are still in the early stage of development and have a lot of room for exciting experiments and challenges. One of the main challenges I have met when doing the experiments for applying Torch-TensorRT is the unsupported combination between different versions and edge devices. Moreover, because of the early improvement state, the coding structure of these techniques is varied. For example, federated learning techniques should be started from individual projects when non of the open-source repository developed by a famous organization or laboratory provides a coding template. However, TensorRT was created and developed by NVIDIA cooperation, and it is an open-source project that fully supports the Pytorch framework, which is the main framework in the project. Therefore, applying TensorRT and doing the experiments to compare with the traditional building direction is compelling.

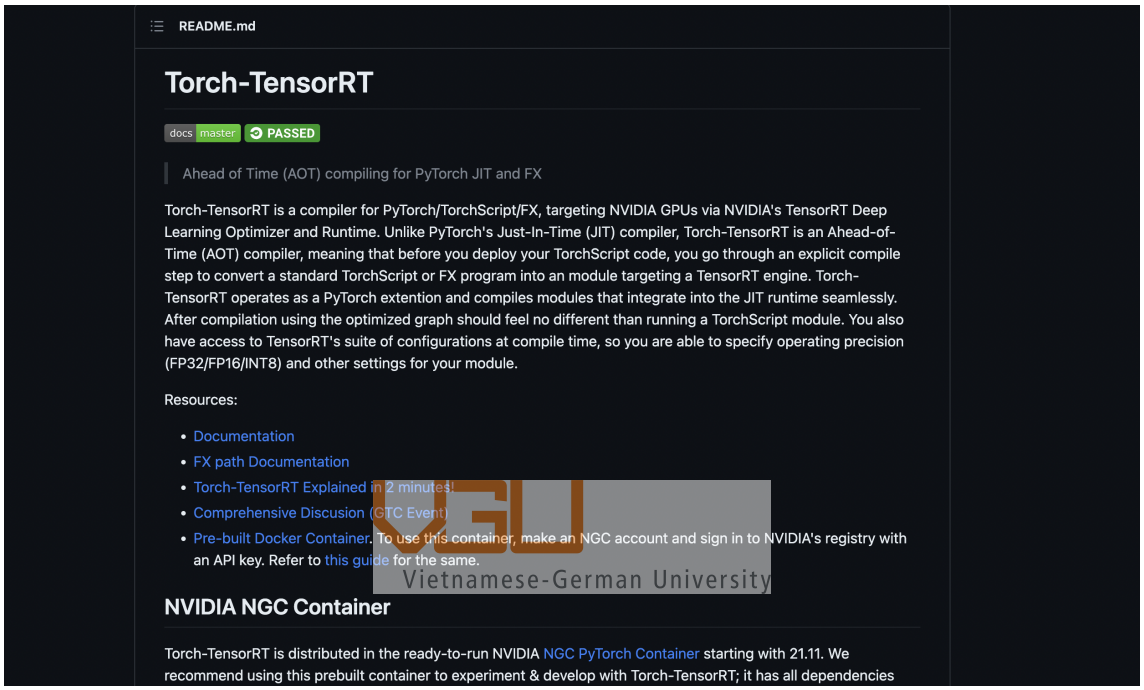

Figure 28: Overview of Torch-TensorRT Github

	RAM usage	Frame Per Second (FPS)
Before Torch-TensorRT	5.29 GB	46.91 fps
After Torch-TensorRT	3.33 GB	61.93 fps

Table 7: Torch-TensorRT application result on RTX GPU 2060 6GB VRAM

Torch-TensorRT [19] shows the most promising and exciting results and experiments on the system. Jetson Nano is a small edge computer designed for embedded applications and AI IoT, which provides GPU enable and 4GB RAM for AI performance. As we can see, the system before applying Torch-TensorRT required nearly 5.3GB RAM to run the entire system. Fortunately, after discovering and testing the TensorRT, the technique reduces the RAM usage (CPU and GPU) to only 3.33GB and increases the FPS exponentially. Therefore, it allows us to deploy the AI system onto the Jetson Nano edge device; the testing results are shown below. The inference speed increases from 46.91 frames being processed per second to nearly 62 FPS tested on GPU RTX 2060 6GB VRAM.

4.4 Docker Test Experiments

The Dockerfile is built based on NVIDIA L4T Pytorch docker images used for Jetson edge devices especially. Docker image NVIDIA L4T Pytorch for Jetpack 4.6 is deployed successfully on Jetson Nano. The Dockerfile adds all the folders and the files of the current working project from my local computer to the Docker image and executes all the initial commands such as update, upgrade, pip-python installing, etc. As a result, the docker container, after being created, is ready to install further required dependencies and runs the system end-to-end. One of the headaches and interesting findings when building the Docker image is the differentiation between the local computer architecture (x86) and the Jetson architecture (arm/v8). Docker extensively provides a "docker buildx build" feature with specified destination computer architecture. Consequently, I successfully built docker images on x86 architecture, which can be run on arm64/v8 Jetson computer architecture. Below is the final Docker build command. Figure 30 shows an overview of the Dockerfile code, which creates an end-to-end Docker image environment for the whole AI system.

```
sudo docker buildx build --platform linux/arm64/v8 -f Dockerfile -t hoangphamviet/
rtfm_i3d-nonlocal_jetson:r35.1.0-pth1.13-py3_v2 --push .
```

On the other hand, the AI system requires a connection to a USB camera. Combining USB cameras and docker containers on Jetson Edge devices requires research and modification to install successfully. I should allow the docker container to access the camera via the xhost access control program on Linux. In the command to run the docker images, the camera devices should also be determined and specified in the X11 volume environment. Below is the Docker command to run and create a Docker container environment.

```
sudo docker run -i -t --runtime nvidia --network host --device /dev/video0:/dev/
video1 --env="DISPLAY" --env="QT_X11_NO_MITSHM=1" --volume="/tmp/.X11-unit:/tmp/.X11
-unix:rw" hoangphamviet/rtfm_i3d-nonlocal_jetson:r35.1.0-pth1.11-py3 bash
```



```
12
13  # L4T Pytorch 1.
14  FROM nvcr.io/nvidia/l4t-pytorch:r35.1.0-pth1.11-py3
15  RUN echo "l4t-pytorch"
16
17  ARG project_dir=/app
18  ARG DEBIAN_FRONTEND=noninteractive
19  WORKDIR $project_dir
20  # ENTRYPOINT ["#!/bin/sh"]  # Comment when install on Jetson AGX X
21
22  ADD main.py $project_dir
23  ADD Dockerfile $project_dir
24  ADD py2trt.py $project_dir
25  ADD temp.py $project_dir
26
27  ADD ckpt ${project_dir}/ckpt
28  ADD configs ${project_dir}/configs
29  ADD data ${project_dir}/data
30  ADD output ${project_dir}/output
31  ADD run_scripts ${project_dir}/run_scripts
32  ADD slowfast ${project_dir}/slowfast
33
34  #RUN rm /etc/apt/sources.list.d/cuda.list
35  #RUN rm /etc/apt/sources.list.d/nvidia-ml.list
36  RUN set -xe \
37      && apt-get update \
38      && apt-get install -y python3-pip
39  # RUN pip3 uninstall -y TensorFlow \
40  #     JupyterLab \
```

Figure 29: Overview of Dockerfile code

4.5 Dependencies installation

Successful dependencies installation requires strong efforts and various skills relating to Docker, Jetson Device, and AI systems. One of the biggest struggles when deploying is the lack of support from the old Jetpack version. JetPack is a software development kit (SDK) from NVIDIA that provides a complete development environment for hardware-accelerated AI-at-the-edge development on NVIDIA Jetson modules. Jetson Nano strictly requires Jetpack Version 4.6 with the operating system Ubuntu 18.04, released on August 4th, 2021. The dependencies support for Torch-TensorRT on Jetson Nano is too old to install and run the whole AI system directly on it. Jetpack 4.6 only support Torch-TensorRT v1.0.0, while the AI system requires at least version v1.1.1 to complete the process. Therefore, I should analyze the errors and look for the parts in v1.1.1 needed for the system while not in v1.0.0. Then, I should compare the difference of the Torch-TensorRT code between v1.0.0 and v1.1.1 based on those parts and transfer the necessary code from v1.1.1 to v1.0.0. Fortunately, the installation process magically runs and finishes automatically. Figure 30 is the overview of Torch-TensorRT v1.0.0 Github.

Figure 30: Overview of torch-tensorRT v1.0.0 Github

On the other hand, connecting the camera attached to Jetson devices with the AI system raise another challenge. The system runs on a Docker container, considered the virtual machine separated from the Jetson edge device host system. Therefore, initially, it was impossible to integrate the camera into the system because the camera had access to the Jetson edge devices instead of the Docker container. After doing some research on the internet, using Xhost to allow the connection of the Docker container to the camera is the most promising solution. Xhost is a server access program that adds and deletes hostnames or usernames to the list allowed to connect to the X server. The x server is the camera in this case. The implementation for this solution is expressed in the command of creating Docker containers.

4.6 Deploy on Jetson Nano test experiments

One of the most hurting points of the deployment process is the Torch-TensorRT installation. Even if Torch-TensorRT announces to support Jetson Nano, its library dependencies version should be strictly followed. On Jetson Nano, versions of Pytorch on L4T docker image do not match with the torch-tensorRT dependencies, combining with the AI solution contains some layers which are not supported by the successfully installed version of torch-tensorRT (v1.0.0). Therefore, I should look for the difference between the currently installed version (v1.0.0) and the new versions (v1.1.1) to look for the supporting codes from the more recent version and add them manually on the current version installed in Jetson Nano. It is an intensive task with only a slight hope for success, and the luck is smile to me!

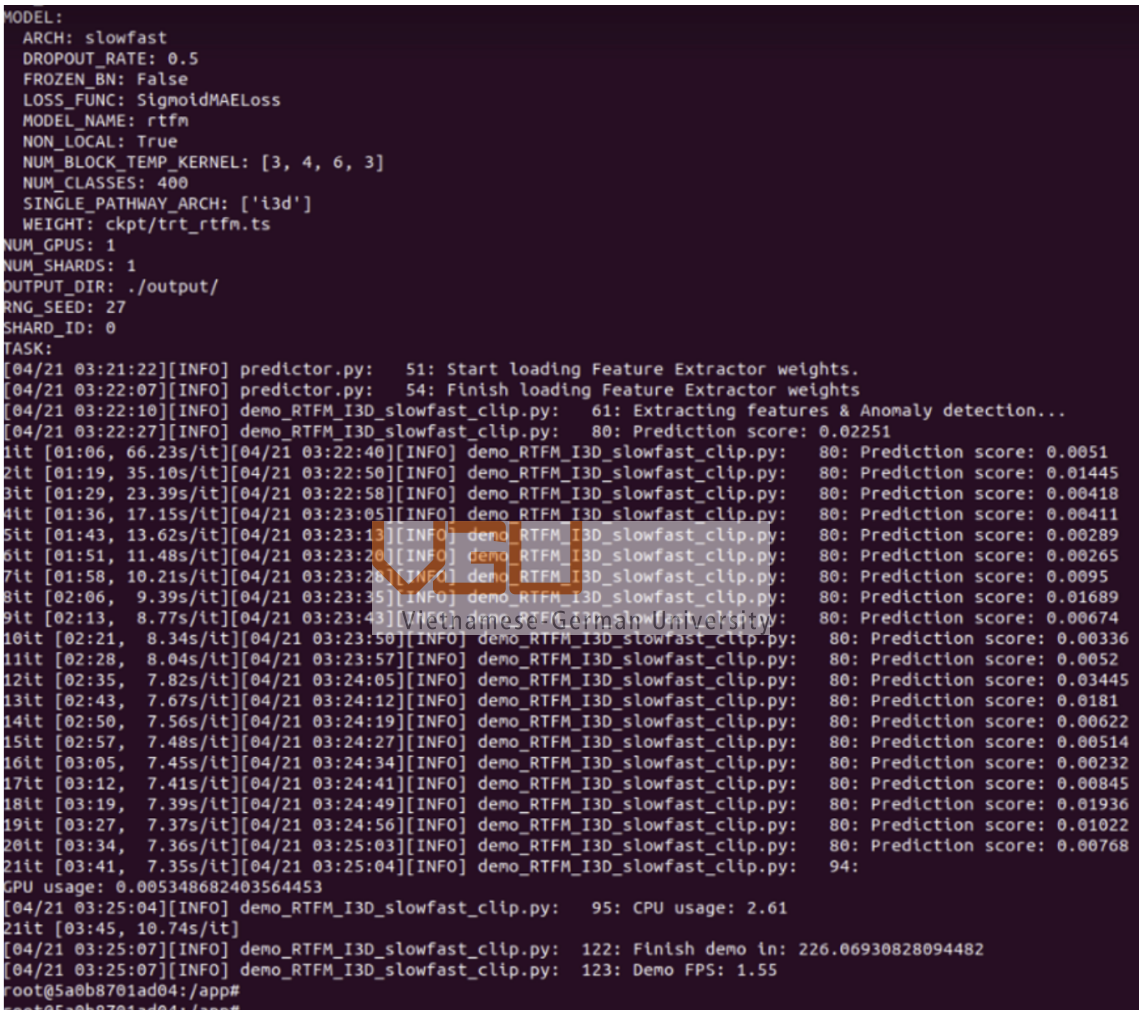

Figure 31: The terminal system log on Jetson Nano

	RAM usage	Frame Per Second (FPS)
Jetson Nano deployment	2.611GB	1.55 fps

Table 8: AI system performance on Jetson Nano

5 Conclusion

In conclusion, a trending problem about anomaly detection related to crime-scene events from surveillance videos is solved by an end-to-end AI system. The system takes the MP4 video inputs from cameras attached to Jetson Edge devices and performs the detection directly at the edge with the help of the TensorRT edge AI method. The solution is built, modified, and adapted based on the open-source Slowfast project first invented by Facebook. This code-based structure is the production-level code with various design pattern applications to make it more robust and reusable. The AI solution for the problem is chosen based on the evaluation ranking on the UCF-Crime dataset and the number of stars on the GitHub model solution code. The higher the number of Github stars, the more reliable the result reproducing. The successful AI solution finding, code-based structure build, optimization, deployment on Jetson Nano, and report for the result evaluation is the main contribution of my thesis. In the future, some general considerations to make the system more applicable are listed as follows.
(1) Some further edge AI optimization techniques can be tested to improve the system's overall performance.
(2) Other Jetson edge devices such as Jetson AGX Xavier, Jetson NX Xavier, and Jetson Orin Nano are great to test and do the experiments.

References

[1] Meta AI. Paperwithcode. *https://paperswithcode.com/sota/anomaly-detection-in-surveillance-videos-on*, 2018.

[2] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. *https://dl.acm.org/doi/10.5555/3295222.3295349*, 2017.

[3] Mayur Rajaram Parate Devashree R. Patrikar. Anomaly detection using edge computing in video surveillance system: review. *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8963404/*, 2022.

[4] NVIDIA Developer. Nvidia jetson nano. *https://developer.nvidia.com/embedded/jetson-nano*, 2019.

[5] NVIDIA Developer. Nvidia tensorrt. *https://developer.nvidia.com/tensorrt*, 2019.

[6] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. *https://openaccess.thecvf.com/content$_C$VPR$_2$020/html/Feichtenhofer$_X$3D$_E$xpanding$_A$rchitectures$_f$or$_E$fficient$_V$*

[7] Joe Fizor. The future of edge computing: Five trends to watch. *https://www.spiceworks.com/tech/edge-computing/guest-article/the-future-of-edge-computing-trends-to-watch/*, 2022.

[8] Github. Pyslowfast. *https://github.com/facebookresearch/SlowFast*, 2019.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015.

[10] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial nets. *https://dl.acm.org/doi/10.5555/2969033.2969125*, 2014.

[11] Ronald G. Harley Jing Dai, Ganesh K. Venayagamoorthy. An introduction to the echo state network and its applications in power system. *https://ieeexplore.ieee.org/document/5352913*, 2009.

[12] Andrew Zisserman Joao Carreira. Kinetics dataset, 2018.

[13] Chloe Hillier Andrew Zisserman Joao Carreira, Eric Noland. Kinetics dataset, 2017.

[14] Ashwin G. Kothari Mayur R. Parate, Kishor M. Bhurchandi. Anomaly detection in residential video surveillance on edge devices in iot framework. *https://deepai.org/publication/anomaly-detection-in-residential-video-surveillance-on-edge-devices-in-iot-framework*, 2021.

[15] McKinsey. Artificial intelligence in business: Separating the real from the hype. *https://www.mckinsey.com/capabilities/quantumblack/our-insights/artificial-intelligence-in-business-separating-the-real-from-the-hype*, 2017.

[16] MicroAI. Edge ai – what is it and how does it work? *tinyurl.com/8n4fpzm9*, 2022.

[17] Saleh Alyahya Shabana Habib Muhammad Islam, Abdulsalam S. Dukyil. An iot enable anomaly detection system for smart city surveillance. *https://www.mdpi.com/1424-8220/23/4/2358*, 2023.

[18] NVIDIA NGC. Nvidia l4t pytorch. *https://catalog.ngc.nvidia.com/orgs/nvidia/containers/l4t-pytorch*, 2019.

[19] Open-Source Project. Torch-tensorrt. *https://github.com/pytorch/TensorRT*, 2020.

[20] Jonathan Huang Zhuowen Tu Kevin Murphy Saining Xie, Chen Sun. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. *https://dl.acm.org/doi/10.1007/978-3-030-01267-0*$_1$9, 2018.

[21] Deci Engineering Team. Webinar: Engineering best practices for dl deployment on nvidia jetson devices. *https://deci.ai/resources/engineering-best-practices-deep-learning-nvidia-jetson/*, 2022.

[22] TECHPOWERUP. Nvidia geforce rtx 2060. *https://www.techpowerup.com/gpu-specs/geforce-rtx-2060.c3310*, 2019.

[23] Ioannis Kompatsiaris Leontios J. Hadjileontiadis Michael Gerasimos Strintzis Vagia Kaltsa, Alexia Briassouli. Swarm intelligence for detecting interesting events in crowded environments. *https://ieeexplore.ieee.org/document/7055905*, 2022.

[24] Tu N Vu, Toan T Dinh, Nguyen D Vo, Tung Minh Tran, and Khang Nguyen. Vnanomaly: A novel vietnam surveillance video dataset for anomaly detection. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 266–271. IEEE, 2021.

[25] Mubarak Shah Waqas Sultani, Chen Chen. Real-world anomaly detection in surveillance videos. *https://ieeexplore.ieee.org/abstract/document/8578776*, 2018.

[26] Wikipedia. Edge computing. *https://en.wikipedia.org/wiki/Edge$_c$omputing*, 2006.

[27] Joongkyu Kim Wonjoon Song, Jonghyun Kim. Weakly supervised video anomaly detection with temporal attention module. *https://ieeexplore.ieee.org/document/9894934*, 2022.

[28] Yoshua Bengio Xavier Glorot, Antoine Bordes. Deep sparse rectifier neural networks. *https://ieeexplore.ieee.org/document/7055905*, 2011.