Artificial Intelligence Framework for Video Analytics:
Detecting Pushing in Crowds

Ahmed Alia

IAS Series
Band / Volume 61
ISBN 978-3-95806-763-9

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum

Schriften des Forschungszentrums Jülich
IAS Series Band / Volume 61

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Zivile Sicherheitsforschung (IAS-7)

Artificial Intelligence Framework for Video Analytics: Detecting Pushing in Crowds

Ahmed Alia

Schriften des Forschungszentrums Jülich
IAS Series

Band / Volume 61

ISSN 1868-8489

ISBN 978-3-95806-763-9

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte Bibliografische Daten
sind im Internet über http://dnb.d-nb.de abrufbar.

Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 52425 Jülich Tel.: +49 2461 61-5368 Fax: +49 2461 61-6103 zb-publikation@fz-juelich.de www.fz-juelich.de/zb
Umschlaggestaltung:	Grafische Medien, Forschungszentrum Jülich GmbH
Titelbild:	Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2024

Schriften des Forschungszentrums Jülich
IAS Series, Band / Volume 61

D 468 (Diss. Wuppertal, Univ., 2024)

ISSN 1868-8489
ISBN 978-3-95806-763-9

Vollständig frei verfügbar über das Publikationsportal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess.

Main Research Field. This thesis primarily contributes to the field of Artificial Intelligence.

Keywords. Artificial Intelligence, Machine Learning, Deep Learning, Convolutional Neural Networks, Intelligent Data Analytics, Image and Video Processing, Intelligent Systems, Internet of Things, Automatic Pushing Detection, Crowd Dynamics.

Acknowledgements

The journey of completing this thesis would not have been possible without the incredible support from many people. First and foremost, I wish to express my profound gratitude to my supervisor, Prof. Dr. Armin Seyfried. His valuable guidance, relentless support, and encouragement have been the beacon that lit my path throughout this PhD journey. He has not only honed my research skills but has also fostered a spirit of independence and teamwork in me. I would furthermore like to thank Prof. Dr.-Ing Karsten Voss, Prof. Dr. Antoine Tordeux and Prof. Dr. Matthias Ehrhardt for their willingness to be members of my examination committee.

I would like to thank Dr. Mohcine Chraibi, an exceptional leader and friend, for teaching me lessons on dealing with challenges and being strong and confident in work and life. Furthermore, his guidance and feedback have not only enhanced my research and communication skills but also helped in the completion of this thesis. Moreover, I am grateful to Dr. Mohammed Maree for his fruitful discussions, valuable insights, and contributions that significantly benefited my thesis.

I am thankful to Prof. Dr. Anna Sieben, Helena Lügering, and Ezel Üsten for developing the manual rating system and annotating the pushing behavior instances in the video experiments. Their contributions were indispensable and laid the foundational groundwork for this journey. I also want to express my gratitude to Dr. Maik Boltes and Tobias Schrödter for their valuable discussions regarding using PeTrack software and video analysis techniques. Furthermore, I am grateful to Dr. David Haensel, Dr. Adel Taweel, Dr. Juliane Adrian, Dr. Fadi Mohsen and Dr. Anas Toma for several valuable discussions. And special thanks go to Arne Graf for ensuring the availability of the essential tools needed throughout this journey and for efficiently resolving numerous technical issues.

I wish to convey my deepest appreciation to all my colleagues at the Wuppertal University and Institute for Advanced Simulation-7, Forschungszentrum Jülich. Your assistance, no matter how small, has played a pivotal role in the success of this journey, and I deeply appreciate your contributions. Additionally, I am highly thankful to Dr. Carole Babelot and Mrs. Yvonne Thies for their exceptional administrative support, which facilitated the smooth resolution of numerous administrative hurdles. Moreover, I am grateful to the Helmholtz AI consultant team at Forschungszentrum Jülich for their insightful discussions about cutting-edge algorithms in Artificial Intelligence.

I would like to extend my sincere thanks to the German Federal Ministry of Education and Research for funding this PhD journey within the Palestinian-German Science Bridge project framework (BMBF: funding number 01DH16027). I also extend my gratitude to Wuppertal University for giving me the opportunity to undertake my PhD studies. Similarly, I appreciate the Forschungszentrum Jülich for allowing me to write my thesis within the Institute for Advanced Simulation.

All this would not have been possible without the support of my wife, Amal

Aldarawsheh, who has always been by my side. Together, we made this journey successful and enjoyable. I cannot thank her enough. Last but not least, I would like to apologize to my kids, Oday and Sham, for the moments when my work took away from our time together, especially with the big change of moving to Germany. Despite this, they both have been amazing, adjusting well and doing great in their schools. I am so proud of both of them and grateful for the strength they've given me.

Abstract

In the modern era, data has become more complex, posing additional challenges to conventional data analysis methods. This is where Artificial Intelligence comes into play, specifically Deep Learning algorithms. These algorithms can analyze such data automatically, quickly, and accurately. Moreover, they can explore complex relationships between variables and identify non-linear patterns humans may not perceive. Leveraging this potential, Deep Learning has recently become pivotal in analyzing complex data, such as video data, arising from human crowds to enhance safety. Despite considerable advancements, some challenging problems in crowd dynamics still need to be solved efficiently and automatically.

One of the critical challenges is assisting organizers and security forces in mitigating the spread of pushing behavior and its risks in dense crowds. In such crowds, some individuals could start pushing each other to reach their destinations faster, like gaining quicker access to an event. Indeed, such behavior often increases the crowd's density, potentially posing a threat not only to people's comfort but also to their safety. In response to this challenge, this cumulative thesis encompasses four publications, each proposing an innovative, automatic, and intelligent approach to glean vital insights from crowd data. The goal is to provide organizers and security teams with the necessary knowledge to alleviate the pushing behavior and its associated risks, thereby enhancing crowd comfort and preventing potential life-threatening situations.

In publication I, a novel Artificial Intelligence-based approach is proposed to identify the patches or regions containing persons who engage in pushing within video recordings of crowds. The approach aims to assist in understanding the causes and the risks of such behavior on the crowd. This approach combines a Deep Learning optical flow model, and a Convolutional Neural Network to annotate the pushing patches in the input video. While the first proposed approach does not meet real-time detection requirements, publication II introduces a new cloud-based Artificial Intelligence approach to detect pushing patches within live camera streams captured from dense crowds. Such approach enables timely and automatic detection, facilitating early intervention by organizers and security forces to prevent situations from escalating dangerously. For this purpose, the approach utilizes a Convolutional Neural Network, a GPU-accelerated Deep Learning optical flow model, a cloud computing environment, and live camera technology.

The first two approaches focus on identifying pushing behavior at the patch level, while publication III introduces a new Artificial Intelligence-based approach that detects pushing in dense crowd videos at the level of individuals (microscopic level). By examining individual interactions more closely, this approach provides new insights into transferring such behavior to neighbors. The proposed approach combines a novel Voronoi method with a Convolutional Neural Network to localize individuals involved in pushing within the given video footage.

Finally, publication IV proposes an intelligent, online, and low-cost GPS-based data system for real-time visualization of crowd dynamics during open and large-scale events. Such visualization helps to avoid or mitigate dense crowds, thus reducing the likelihood of conditions that could increase pushing behavior and its effect on crowds. For this purpose, the system employs a custom mobile application, smartphone GPS receivers, a web-based platform, and advanced visualization techniques.

Zusammenfassung

In der heutigen Zeit werden Daten immer komplexer, was konventionelle Datenanalysemethoden vor zusätzliche Herausforderungen stellt. An diesem Punkt kommen künstliche Intelligenz und insbesondere Deep Learning Algorithmen ins Spiel. Diese Algorithmen können Daten automatisch, schnell und akkurat analysieren. Darüber hinaus können sie komplexe Strukturen zwischen den Variablen untersuchen und nicht-lineare Muster erkennen, die Menschen eventuell nicht wahrnehmen. Durch die Nutzung dieses Potenzials ist Deep Learning in jüngster Zeit zu einem zentralen Element bei der Analyse komplexer Daten aus Menschenmengen geworden, um die Sicherheit zu verbessern. Trotz des beachtlichen Fortschritts gibt es im Bereich der Gruppendynamik immer noch einige Schwierigkeiten, die effizient und automatisch gelöst werden müssen.

Eine wesentliche Herausforderung besteht darin, Veranstalter und Sicherheitskräfte dabei zu unterstützen, die Ausbreitung von Gedränge und die damit verbundenen Risiken in dichten Menschenmengen einzudämmen. In solchen Menschenmengen kann es vorkommen, dass einige Personen anfangen, sich gegenseitig zu schubsen, um ihr Ziel schneller zu erreichen und sich beispielsweise schnelleren Zugang zu einer Veranstaltung zu verschaffen. Tatsächlich erhöht ein solches Verhalten jedoch oft die Dichte in einer Menschenmenge, was nicht nur das Wohlbefinden der Menschen beinträchtigt, sondern auch ihre Sicherheit bedroht en kann. Als Reaktion auf diese Herausforderung umfasst diese kumulative Promotion vier Publikationen, die jeweils einen innovativen, automatischen und intelligenten Ansatz vorschlagen, um wichtige Erkenntnisse aus Daten von Menschenmengen zu gewinnen. Das Ziel ist, Informationen für Veranstalter und Sicherheitskräfte bereitzustellen, um Drängelverhalten und die damit verbundenen Risiken einzuschränken und so den Komfort für die Besucher zu erhöhen und potenziell lebensbedrohliche Situationen zu verhindern.

In Publikation I wird ein neuer Ansatz basierend auf künstlicher Intelligenz vorgestellt, der in Videoaufnahmen von Menschenmengen Abschnitte oder Bereiche identifiziert, in denen Personen drängeln. Der Ansatz soll Veranstalter dabei unterstützen, zu verstehen, wann, wo und warum Drängeln auftritt, damit sie Menschenmengen angenehmer und sicherer gestalten und steuern können. Dieser Ansatz kombiniert ein auf Deep Learning und ein Convolutional Neural Network, um Bereiche in dem Video zu markieren, die Drängelverhalten enthalten. Da die erste vorgestellte Methode den Anforderungen einer Echtzeit-Erkennung nicht gerecht wird, befasst sich Publikation II mit einem neuen cloudbasierten künstliche Intelligenz-Ansatz, der Bereiche mit Drängelverhalten im Kamera-Live-Stream von dichten Menschenmengen erkennt. Eine solche Methode ermöglicht die rasche und automatische Erkennung sowie ein frühzeitiges Eingreifen von Veranstaltern und Sicherheitskräften und verhindert so eine gefährliche Eskalation der Situation. Zu diesem Zweck nutzt der Ansatz ein Convolutional Neural Network, ein GPU-beschleunigtes Deep Learning optischer Fluss Modell, eine Cloud-Computing-Umgebung und eine Live-Kamera-Technologie.

Die ersten zwei Ansätze zielen darauf ab, Drängelverhalten in definierten Teilbereichen zu identifizieren, während Publikation III einen neuen künstliche Intelligenz-basierten Ansatz einführt, der Drängeln in Videos von dichten Menschenmengen auf Ebene der Individuen erkennt (mikroskopische Ebene). Indem die individuellen Interaktionen genauer untersucht werden, liefert diese Methode neue Erkenntnisse zum Verhalten individueller Personen und deren Wirkung auf ihre nächsten Nachbarn. Der vorgestellte Ansatz kombiniert eine Voronoi Zerlegung des Bildes anhand der Positionen der Personen mit einem Convolutional Neural Network um die Individuen, die drängeln, auf dem Videomaterial zu lokalisieren.

Abschließend wird in Publikation IV ein intelligentes, kostengünstiges, GPS-basiertes Online-Datensystem für die Echtzeit-Visualisierung von Gruppendynamiken für große Veranstaltungen im Freien vorgestellt. Eine solche Visualisierung trägt dazu bei, dichte Menschenmengen zu vermeiden oder zu entschärfen und damit die Wahrscheinlichkeit von Zuständen zu verringern, die Drängelverhalten und dessen Auswirkungen auf Menschenmengen verstärken könnten. Zu diesem kombiniert setzt das System eine selbstständig entwickelte mobile Anwendung, Smartphone GPS-Empfänger, eine webbasierte Plattform und hochentwickelte Visualisierungstechniken.

Contents

Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
RAFT	Recurrent All-Pairs Field Transforms for Optical Flow
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RSE	Research Software Engineering
FAIR	Findable, Accessible, Interoperable, and Reusable
MIM	Motion Information Map
ROI	Region of Interest
FC	Fully Connected layer
GAP	Global Average Pooling2D
ReLU	Rectified Linear Unit
LSTM	Long Short-Term Memory
CSV	Comma-Separated Values
P	Pushing
NP	Non-pushing
TNP	True Non-pushing
FNP	False Non-pushing
FP	False Pushing
TP	True Pushing
YOLO	You Only Look Once
MBConv	Mobile Inverted Residual bottleneck convolution
Fused-MBConv	Fused Mobile Inverted Residual Bottleneck Convolution
ROC	Receiver Operating Characteristics
AUC	Area Under the Curve
TPR	True Pushing Rate
TNPR	True Non-Pushing Rate
GPS	Global Positioning System
PHP	Hypertext Preprocessor
SQl	Structured Query Language
JSON	JavaScript Object Notation
XML	Extensible Markup Language
iOS	iPhone Operating System

List of Publications and Awards

Journal Publications included in this Thesis

I. **Ahmed Alia**, Mohammed Maree, and Mohcine Chraibi. "A Hybrid Deep Learning and Visualization Framework for Pushing Behavior Detection in Pedestrian Dynamics." Sensors 22, no. 11 (2022): 4040. https://doi.org/10.3390/s22114040.

II. **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi, Anas Toma, and Armin Seyfried. "A Cloud-based Deep Learning Framework for Early Detection of Pushing at Crowded Event Entrances." IEEE Access (2023). https://doi.org/10.1109/ACCESS.2023.3273770.

III. Ahmed Alia, Mohammed Maree, Mohcine Chraibi, and Armin Seyfried. "A Novel Voronoi-based Convolutional Neural Network Framework for Pushing Person Detection in Crowd Videos." Complex & Intelligent Systems (2024). https://doi.org/10.1007/s40747-024-01422-2.

IV. **Ahmed Alia**, Mohammed Maree, and Mohcine Chraibi. "On the Exploitation of GPS-based Data for Real-Time Visualisation of Pedestrian Dynamics in Open Environments." Behaviour & Information Technology 41, no. 8 (2022): 1709-1723. https://doi.org/10.1080/0144929X.2021.1896781.

Other Journal Publications

V. **Ahmed Alia**, and Adel Taweel. "Enhanced Binary Cuckoo Search with Frequent Values and Rough Set Theory for Feature Selection." IEEE access 9 (2021): 119430-119453. https://doi.org/10.1109/ACCESS.2021.3107901.

VI. Helena Lügering, **Ahmed Alia**, and Anna Sieben. "Psychological Pushing Propagation in Crowds – Does the Observation of Pushing Behavior Promote Further Intentional Pushing?." (2023). https://doi.org/10.3389/frsps.2023.1263953.

Awards

I. **Research Excellence Award**, Palestinian Islamic Bank Award for Scientific Research, Second Prize in the field of Education, Learning, and Business in the Age of Technology and Artificial Intelligence, Palestine, 2023.

II. **Best Presenter Award** in the Intelligent Image Processing and System Monitoring session at the 3rd International Conference on Computers and Automation (CompAuto 2023), held in Paris, France, from December 7-9, 2023.

Author's Contributions

Publication I
Conceptualization: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
Methodology: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
Software: **Ahmed Alia**
Validation: **Ahmed Alia**
Formal Analysis: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
Visualization: **Ahmed Alia**
Data Curation: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
Writing—Review and Editing **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
GitHub Repository Creation: **Ahmed Alia**

Publication II
Conceptualization: **Ahmed Alia**
Methodology: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi, Armin Seyfried
Software: **Ahmed Alia**
Validation: **Ahmed Alia**
Formal Analysis: **Ahmed Alia**
Visualization: **Ahmed Alia**, Anas Toma
Data Curation: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
Writing—Review and Editing **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi, Anas Toma, Armin Seyfried
GitHub Repository Creation: **Ahmed Alia**

Publication III
Conceptualization: **Ahmed Alia**
Methodology: **Ahmed Alia**, Armin Seyfried
Software: **Ahmed Alia**
Validation: **Ahmed Alia**
Formal Analysis: **Ahmed Alia**
Visualization: **Ahmed Alia**
Data Curation: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
Writing—Review and Editing **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi, Armin Seyfried
GitHub Repository Creation: **Ahmed Alia**

Publication IV
Conceptualization: **Ahmed Alia**, Mohammed Maree
Methodology: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
Software: **Ahmed Alia**
Validation: **Ahmed Alia**

Formal Analysis: **Ahmed Alia**
Visualization: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
Writing—Review and Editing **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
GitHub Repository Creation: **Ahmed Alia**

Chapter 1

Introduction

1.1 Overview

The availability of complex datasets is more widespread than ever, especially that encompassing images and videos. Efficiently analyzing and interpreting this data opens up new knowledge and solving many real-world problems. Such data often involve intricate and non-linear relationships that are difficult to study with traditional analytical methods. Artificial Intelligence (AI), and more specifically, Deep Learning (DL), have revolutionized the field of complex data analytics [1]. These advancements are pivotal in developing intelligent and automatic applications that tackle complex issues in various domains, including finance [2], healthcare [3], transportation [4], agriculture [5], energy [6], education [7], and crowd safety [8]. Within this broad scope, this thesis focuses on developing an innovative and automatic AI-based framework for analyzing video data captured from crowds to help improve crowd safety.

1.2 Background and Motivation

With the rapid development of urbanization, crowded places have become common, such as religious areas, train terminals, concert venues, sports stadiums, shopping centers, and tourist attractions. In such dense crowds, pushing behavior could be a safety risk [9, 10, 11]. People may start pushing for different reasons. 1) Saving their lives in emergencies or tense scenarios. 2) Grabbing a limited resource, such as gaining access to a crowded subway train [12]. 3) Accessing a venue more quickly; for instance, in crowded event entrances, some pedestrians start pushing others to enter the event faster [13]. The focus of this thesis is the pushing that occurs in crowded event entrances due to the availability of public real-world experiments about such entrances. In this context, Lügering et al. [13] defined pushing as a behavior that pedestrians use to reach a target (like accessing an event) faster. This behavior involves pushing others using arms, shoulders, elbows, or the upper body, as well as utilizing gaps among neighboring people to navigate forward quickly.

In fact, such behavior often increases the crowd's density, potentially posing a threat not only to people's comfort but also to their safety [14, 15, 16, 11]. This is particularly true when numerous individuals engage in pushing. Mitigating the spread of pushing behavior and reducing the associated risks helps enhance the comfort and safety of crowds. The following are useful techniques in achieving this purpose [13, 17]:

1. **Understanding pushing dynamics.** Identifying such behavior in crowds can help to know where, when, and why such behavior occurs, as well as

understanding its impact on the crowd. This knowledge can facilitate the development of effective crowd management strategies and improve public space designs, thereby reducing the incidence of pushing and its risks in dense crowds.

2. **Timely pushing identification.** Real-time detection of pushing behavior offers a critical advantage. It allows organizers and security forces to intervene at an early stage before pushing spreads and situations become dangerous. Mainly when unexpected pushing instances occur, and current strategies prove insufficient for the current situation.

3. **Avoidance of high-density areas.** Pushing behavior is more likely to arise in areas of high density, particularly within moving crowds. Managing crowds and reducing areas with high density help mitigate the likelihood of such behavior and the associated risks. Real-time visualization of crowd evolution is crucial for assisting organizers and security teams in successfully managing crowds and reducing overcrowded areas.

Recently, some researchers have attempted to understand pushing by manually observing and identifying such behavior among video recordings of crowded events. For example, Lügering et al. [13] proposed a manual rating system to understand when, where, and why pushing appears. The system utilizes two trained psychologists who manually categorize pedestrian behaviors over time in top-view videos into two main categories: pushing and non-pushing. However, this manual rating procedure is time-consuming, tedious, and prone to errors in some scenarios. Additionally, it requires trained observers, which may not always be feasible, and it does not cope with real-time detection requirements. Therefore, the automatic approaches capable of identifying pushing in video recordings and real-time camera streams are helpful in many respects.

From a computer science perspective, the visual analysis of the crowd falls within the realm of computer vision. DL algorithms, particularly Convolutional Neural Network (CNN) architectures, have demonstrated significant success in various computer vision tasks, including face recognition [18], object detection [19], and abnormal behavior detection [20]. Technically, pushing identification is closely related to the task of abnormal behavior detection in video crowds. Typically, behavior is considered abnormal when seen as unusual under specific contexts. This implies that the definition of abnormal behavior depends on the situation [21]. To illustrate, running inside a bank might be considered abnormal behavior, whereas running at a traffic light could be normal [22]. Several behaviors have been automatically addressed in crowds, including walking in the wrong direction [23], running away [24], sudden people grouping or dispersing [25], human falls [26], suspicious behavior, violent acts [27], abnormal crowds [28], hitting, and kicking [29]. To the best of our knowledge, no previous studies have automatically identified pushing for faster access in human crowds. As a result, this thesis primarily focuses on developing approaches for automatically detecting pushing in video recordings and live streams of dense crowds.

1.3 Challenges of Automatic Pushing Detection

The automatic detection of pushing behavior in videos or live camera streams of dense crowds represents a novel research and a demanding task. The challenges

of automatic pushing detection stem from multiple factors, which are:

1. **Diversity in pushing behavior.** Such behavior has various forms, varying in intensity and nature, not just among different individuals but also within the same person over time. Additionally, the significant similarity and overlap between pushing and non-pushing behaviors complicate the differentiation task.

2. **Crowd density.** In densely crowded scenes such as entrances of large-scale events, multiple people often overlap, leading to frequent occlusions. Consequently, body parts involved in pushing can become invisible. This makes the analysis challenging because extracting individual interactions is required for identifying pushing behavior.

3. **Feature representation.** One of the main challenges is the lack of knowledge about the relevant features that represent pushing behavior. Without accurately determining these features, it would be impossible to create reliable automatic methods for such detection.

Considering the challenges mentioned above, it's clear that the primary obstacle to automating pushing detection is identifying and extracting relevant features of pushing from dense crowds.

1.4 Preliminary Concepts

This section provides an overview of the fundamental concepts utilized in this thesis.

Within the rapidly evolving field of **AI**, which is dedicated to creating systems that can emulate human intelligence functions such as learning, reasoning, and problem-solving [30], subfields have arisen, including **Machine Learning (ML)** and **DL**. ML empowers computers to learn and adapt using data, eliminating the need for explicit programming [31]. In contrast, DL is a specialized branch of ML that employs artificial neural networks with three or more layers to simulate the human brain's structure and function, allowing the learning of complex patterns from large datasets without needing data pre-processing as classical machine learning algorithms [32]. Among the various DL techniques, **CNN**, inspired by the animal visual cortex organization, achieved remarkable success in vision-related tasks [33]. This success is attributed to CNNs' unmatched ability to automatically learn feature representations from data without human supervision [34]. The relationship between AI, ML, DL, and CNN is shown in Fig. 1.

A typical CNN architecture consists of repeated blocks of convolution and pooling layers, followed by one or more fully connected layers. The convolution and pooling layers are responsible for feature extraction. In contrast, the fully connected layers map the extracted features to the final layer, which contains a neuron for each potential category in classification [35]. Recently, many advancements have been made in CNN architectures, among which the **EfficientNet** family [36, 37] stands out. It encompasses various architectures that outperform existing CNNs regarding accuracy and efficiency (smaller and faster) across different classification tasks.

Recurrent All-Pairs Field Transforms for Optical Flow (RAFT) is one of the most powerful DL-based methods for dense optical flow estimation, a task of

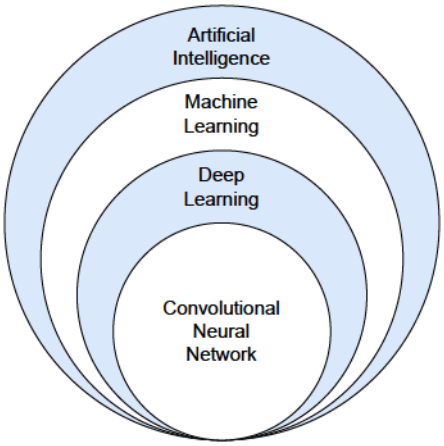

Figure 1: Relationship between AI, ML, DL and CNN.

estimating per-pixel motion between video frames. RAFT, a composition of CNN and recurrent neural network (a class of DL) architectures, has been trained on the SINTEL dataset [38], resulting in an efficient and general model to estimate dense optical flow efficiently. This pre-trained model is a promising for applications in dense crowds due to its ability to reduce the impact of occlusions on optical flow estimation [39].

1.5 State-of-the-Art CNN-based Approaches

As previously discussed, automatic identification of pushing behavior in videos falls under the domain of computer vision, specifically in the task of abnormal behavior detection. The literature presents various automatic approaches to detecting abnormal behavior in videos, categorized into traditional-based and learning-based methods [40, 41]. The main idea of the first group is to use some measures to distinguish between normal and abnormal behaviors. For example, a high value in the entropy [42] or the speed [43] may indicate the presence of abnormal behavior in crowds. However, these approaches are inefficient for complex behaviors due to the difficulty of describing them accurately by rules [41]. To address these limitations, learning-based approaches, especially those harnessing the power of CNNs, have been developed. These algorithms learn directly from data, avoiding manual rule-setting, and have shown notable advancements in the area.

A customized CNN-based method to identify abnormal activities in videos was presented by Tay et al [21]. The authors trained a customized CNN for feature extraction and labeling using normal and abnormal samples. In another study, Alafif et al. [44] proposed two methods of identifying abnormal behaviors in small and large-scale crowd videos. The first method employs a combination of a CNN model and a random forest classifier to detect anomaly behaviors at the object level in a small-scale crowd. In contrast, the second method utilizes two classifiers to recognize abnormal behaviors in a large-scale crowd. The initial model, finds the frames containing abnormal behaviors, while the second classifier, You Only Look Once (YOLOv2), processes those frames to identify abnormal behaviors exhibited by individuals. The effectiveness of these techniques relies heavily on utilizing CNNs to learn features from labeled datasets containing both normal and

abnormal behaviors. A large training dataset of normal and abnormal behaviors is necessary to create an accurate and adaptable CNN model. However, obtaining such a dataset is often unattainable for various abnormal behaviors, including pushing behavior. This challenge arises because labeling data is exceptionally time-consuming and may require specialized knowledge. Furthermore, securing a significant number of videos that exhibit these behaviors can be difficult [22].

In order to overcome the shortage of large datasets comprising normal and abnormal behaviors, some researchers have utilized one-class classifiers with datasets consisting only of normal behaviors. It is easier to obtain or create a dataset that contains only normal behaviors than a dataset that includes both normal and abnormal behaviors [45, 46]. The fundamental concept behind the one-class classifier is to exclusively learn from normal behaviors, thereby establishing a class boundary between normal and undefined (abnormal) classes. For example, Sabokrou et al. [45] employed a pre-trained CNN for extracting motion and appearance information from crowded scenes. Subsequently, they utilized a one-class Gaussian distribution to construct the classifier using datasets comprised of normal behavior. Similarly, in [46, 47], the authors developed one-class classifiers by utilizing a dataset of normal samples. In [46], Xu et al. employed a convolutional variational autoencoder to extract features, followed by the use of multiple Gaussian models to detect abnormal behavior. Meanwhile, in [47], a pre-trained CNN model was utilized for feature extraction, with one-class support vector machines being used to identify abnormal behavior. Another study by Ilyas et al. [48] utilized a pre-trained CNN and a gradient sum of the frame difference to extract significant features. Following this, three support vector machines were trained on normal behavior to detect abnormal behaviors. Generally, the one-class classifier is commonly used when the target behavior class or abnormal behavior is infrequent or poorly defined [49]. However, pushing behavior is not rare, particularly in high-density and competitive situations. Furthermore, this type of classifier regards new normal behavior as abnormal.

To overcome the limitations of CNN-based and one-class classifier approaches, several studies have combined multi-class CNN with one or more handcrafted feature descriptors [48, 20]. As an example, Duman et al. [22] utilized the traditional Farnebäck optical flow method in conjunction with CNN to detect anomalous behavior. They extracted direction and speed information using the optical flow method, and CNN and then utilized a convolutional long short-term memory network to construct the classifier. Similarly, Hu et al. [50] employed a combination of the histogram of gradient and CNN for feature extraction, while a least-squares support vector was used for classification. Almazroey et al. [51] focused on utilizing the Lucas-Kanade optical flow method, pre-trained CNN, and feature selection method (neighborhood component analysis) to extract relevant features. They then used a support vector machine to generate a trained classifier. In a different study [52], Zhou et al. introduced a CNN-based method to identify and locate abnormal activities. This approach integrated traditional optical flow method with CNN for feature extraction and utilized a CNN for classification. Direkoglu [20] utilized the Lucas-Kanade optical flow method and CNN to extract relevant features and identify "escape and panic behaviors[1]".

In summary, hybrid-based approaches could be more suitable for automatically detecting pushing behavior due to the limited availability of labeled pushing data.

[1]For additional details regarding the term "panic behavior", please refer to Ref. [53].

Nevertheless, most of the reviewed hybrid-based approaches for abnormal behavior detection may be inefficient for detecting pushing, since 1) The descriptors used in these approaches can only extract limited essential data from high-density crowds to represent pushing behavior. 2) Some CNN architectures commonly utilized in these approaches may not be effective in dealing with the increased variations within pushing behavior (intra-class variance) and the substantial resemblance between pushing and non-pushing behaviors (high inter-class similarity), which can potentially result in misclassification.

1.6 Objectives

Given the challenges and motivations previously discussed, this thesis primarily aims to develop an automatic AI-based framework for detecting pushing behavior in video recordings and live camera streams of dense crowds. This framework is based on cutting-edge technologies, including supervised deep learning algorithms, visualization techniques, cloud computing, and the Internet of Things. The development of the framework consists of three phases:

1. The first phase introduces an approach combining deep learning (DL) and visualization to identify the regions or patches where pushing occurs in crowd videos (publication I). Notably, each region covers an area between 1 and 2 square meters on the ground.

2. While the initial approach isn't designed for real-time detection, the second phase (publication II) introduces a cloud-based DL approach for real-time detection. This approach aims to annotate regions containing pushing behavior in live camera streams of dense crowds at an early stage.

3. While the initial phases focus on region-based detection, the third phase (publication III) introduces a Voronoi-based CNN approach to identify individuals engaged in pushing within crowds (microscopic level). The approach leverages video recordings of crowds and pedestrian trajectory data to provide new insights into the interactions between pedestrians in the crowd.

In addition to the primary objective, this thesis also presents an intelligent, real-time, online, and low-cost GPS-based data system (publication IV). This system offers real-time visualization of crowd dynamics in open environments.

1.7 Contributions

This thesis is based on four publications, where the contributions of each are outlined below:

Publication I:

1. Introducing the first AI-based approach, to our knowledge, dedicated to automatically identifying pushing regions in video recordings of crowds.
2. Integrating EfficientNetV1B0-based CNN [36], RAFT [39], and color wheel visualization [54, 55] within a unique approach for pushing behavior detection.
3. Proposing a patch-based method to augment data and address the class imbalance issue in video datasets.
4. Generating the first publicly available dataset in this research domain, consisting of pushing and non-pushing patches.
5. Introducing a false reduction algorithm to enhance the accuracy of the approach.

Publication II:

1. Proposing the first AI-based approach, to our knowledge, for automatic and early detection of pushing regions in live camera streams of crowds.
2. Integrating an adapted EfficentNetV2B0 [37], a GPU-accelerated pre-trained RAFT model [39], the color wheel method [54, 55], a cloud computing environment, and live camera technology to develop an approach that capable for early detection of pushing regions in crowds.
3. Creating a novel patch-based pushing dataset that encompasses more complex scenarios.
4. Conducting a comprehensive performance comparison across fifteen CNN architectures using the generated dataset.

Publication III:

1. Introducing the first approach, to our knowledge, for automatically identifying pushing in video recordings of crowds at the microscopic level.
2. Presenting a novel feature extraction technique to characterize microscopic behaviors, especially pushing, in crowd videos.
3. Developing a new dataset with microscopic pushing and non-pushing samples.

Publication IV:

1. Introducing a system that collects pedestrian trajectories every second, estimating horizontal accuracy to depict the maps accurately.
2. Presenting a novel data processing method with temporal storage of current pedestrian positions to improve the data visualization performance.
3. Providing access to online and real-time visualized maps depicting pedestrian movements and timely heat maps for spotting crowded areas.
4. Offering an archive of collected GPS data in various formats (SQL, JSON, CSV) for interested researchers.

1.8 Ethical Considerations

1. **The videos of experiments utilized for dataset generation:** The experiments were conducted according to the guidelines of the Declaration of Helsinki, and approved by the ethics board at the University of Wuppertal, Germany. Informed consent was obtained from all subjects involved in the experiments.

2. **A cloud-based AI approach for early detection of pushing at human crowds:** To protect individuals' privacy, the approach applies a blurring technique to the annotated frames before storing them in the cloud.

3. **An online, intelligent, and low-cost GPS-based data system:** To safeguard users' privacy, the mobile application collects only the smartphone's latitude and longitude coordinates, associating them with an anonymous ID.

1.9 Thesis Structure

The thesis is organized into two parts, as shown in Fig. 2:

1. **First part:** consists of three chapters. Chapter 1 provides the background and motivation, challenges in automatic pushing detection, state-of-the-art CNN-based approaches, main goal and objectives, ethical considerations, and contributions. Then, Chapter 2 provides an overview of each publication in the thesis. Subsequently, Chapter 3 discusses the limitations of the works presented in this thesis and highlights potential areas for further research directions in the future.

2. **Second part:** lists the four research articles in this dissertation's context. The publications have been reformatted for consistency with the thesis format. Importantly, this thesis not only puts forward various novel approaches but also strongly advocates for Research Software Engineering (RSE) principles, emphasizing open-source development, reproducibility, and the FAIR (Findable, Accessible, Interoperable, and Reusable) principles. All methodologies developed in this work, alongside the accompanying source codes used for building, training, and assessing the models, the trained models, and the datasets, are openly accessible on GitHub at the following links:

   Publication I: `https://github.com/PedestrianDynamics/DL4PuDe`
   Publication II: `https://github.com/PedestrianDynamics/CloudFast-DL4PuDe`
   Publication III: `https://github.com/PedestrianDynamics/VCNN4PuDe`
   Publication IV: `https://github.com/PedestrianDynamics/GPSdataColVis`

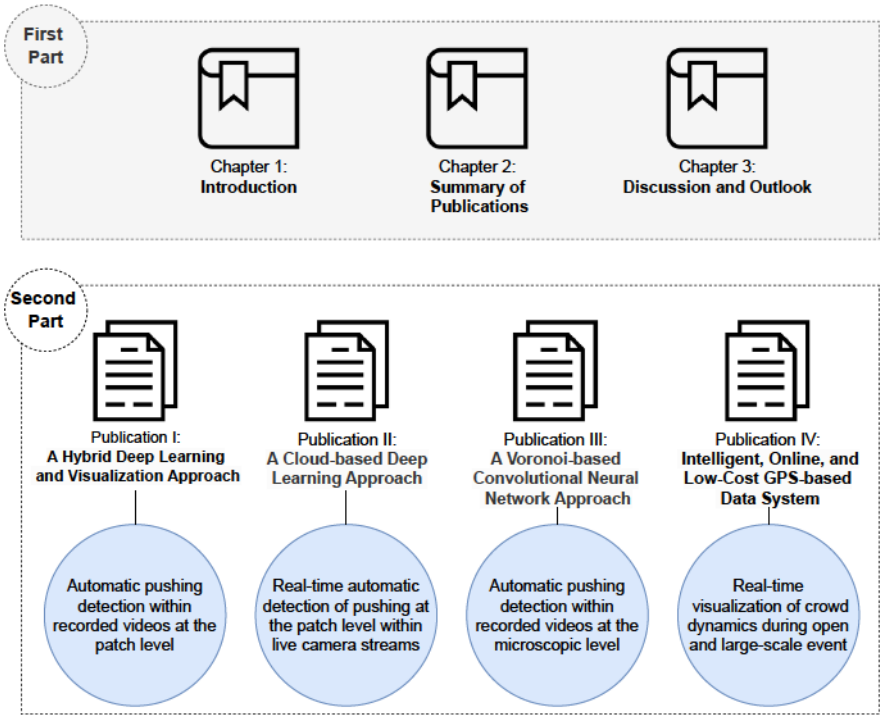

Figure 2: An outline of the thesis structure.

Chapter 2

Summary of Publications

In the context of this thesis, which focuses on AI and Video Analytics, four articles have been published. This chapter offers a summary of the methodologies, objectives, and results delineated in each publication. The full texts of the publications can be found in subsequent chapters at the end of this thesis.

2.1 Publication I:

A Hybrid Deep Learning and Visualization Framework for Pushing Behavior Detection in Pedestrian Dynamics

Within crowded event entrances, pushing behavior can easily arise, potentially posing a threat not only to people's comfort but also to their safety, especially when many people start pushing each other to reach their goals faster [11, 15, 10, 56, 9]. By identifying instances of pushing in crowd videos, a deeper understanding of when, where, and why such behavior occurs can be achieved [13]. This knowledge is essential to creating effective crowd management strategies and better designing public spaces, thus enhancing crowd safety. However, manually identifying pushing in crowd videos is challenging because it is time-consuming, tedious, prone to errors, and needs trained experts [13]. Therefore, this publication proposed a hybrid deep learning and visualization framework that aims to assist researchers in automatically identifying pushing behavior in video recordings of crowds.

The proposed framework comprises two main components: Motion Information Extraction and Pushing Patch Annotation. The first component combines the color wheel method [54, 55] and one of the most efficient deep optical flow models for crowds. Such a combination is responsible for estimating and visualizing the crowd motion from the input video. The deep optical flow estimator relies on the pre-trained CPU-based RAFT model [39] to calculate the optical flow vectors for all pixels between two frames. Then, the color wheel method is used to infer and visualize the motion information from the estimated optical flow vectors. Specifically, the color wheel firstly calculates each vector's magnitude and direction at each pixel. After that, it visualizes the calculated information to generate motion information maps. In these maps, color indicates the direction of pixel movement, while the color's intensity denotes the magnitude or speed of that movement. Furthermore, each map's pixel position mirrors movement locations within the crowds, and each map represents a specific time segment from the input video. Next, each map is divided into several patches to help the framework identify the regions that contain at least one person to engage in pushing (at the patch level).

The second component, Pushing Patch Annotation, is responsible for localizing

the patches that contain pushing persons, annotating the patches, and generating the annotated videos. This component modifies and trains EfficientNetV1B0-based CNN [36] to build a robust classifier. This classifier identifies relevant features from the patches and categorizes them into pushing and non-pushing labels. Besides the classifier, a false reduction algorithm was designed in the Pushing Patch Annotation component to enhance the classifier's predictions. Finally, the component outputs the annotated video showing the patches containing at least one person engaged in pushing.

In order to train and evaluate the classifier and the framework, five videos of real-world experiments, along with their ground truth data of pushing, have been used to prepare the final dataset. The five videos were selected from the data archive hosted by the Forschungszentrum Jülich [57]. These videos are identified by the experiment numbers 110, 150, 170, 270, and 280. Each experiment's setup mimics a straight entrance with a single gate. Firstly, maps are extracted from the videos. Each map is then divided into several equal patches, and each patch dimensions on the ground ranged between 1.2 and 1.7 meters. After that, based on the ground truth data, the patches that contain at least one person engaged in pushing are classified as pushing; otherwise, they are labeled as non-pushing. The final dataset includes 2364 pushing patches and 1722 non-pushing patches of maps.

Finally, experimental results show that the proposed framework achieved 88 % accuracy.

2.2 Publication II:

A Cloud-based Deep Learning Framework for Early Detection of Pushing at Crowded Event Entrances

While the earlier framework was designed to identify pushing patches in video recordings of human crowds, publication II introduced a new cloud-based DL framework for early identifying pushing patches within such crowds. Automatic and timely identification of such behavior would help organizers and security forces intervene at an early stage before situations become dangerous. Furthermore, such timely identification can aid in assessing the efficiency of implemented plans and strategies, allowing for early detection of vulnerabilities and potential improvement points.

The architecture of the cloud-based DL framework comprises three major components: Preprocessing, Motion Descriptor, and Pushing Detection and Annotation. The first component aims to capture a live camera stream of crowds, process it, and then display it on a web client in real-time. At the same time, it collects only the data required for detection purposes from the live stream. Simultaneously, the second component, Motion Descriptor, aims to extract the crowds motion characteristics at the patch level from the collected data. More specifically, this component estimates the motion direction, magnitude, and associated spatio-temporal information from the crowds and accordingly visualizes them as motion maps. This component integrates a robust deep optical flow model (GPU-based pre-trained RAFT model [39]) with the color wheel method to accurately and rapidly generate the map patches every two seconds from the live stream. Finally, the Pushing Detection and Annotation component utilizes the adapted and trained EfficientNetV2B0-based CNN model [37] to analyze the maps and

detect pushing patches accurately. Notably, it directly annotates the regions that contain pushing behavior on the live stream on the web client. Simultaneously, the component stores the annotated data in the cloud storage, where the stored data is blurred, ensuring the privacy of individuals is maintained. Moreover, the framework uses live capturing technology and a cloud environment to provide more powerful computational resources, help collect video streams of crowds in real-time, and provide early-stage results.

A new dataset has been created to train and evaluate the classifier and the overall framework under various scenarios. This dataset was generated using a new video called Entrance_2 [58], in addition to the previously utilized videos labeled 110, 150, 270, and 280 [57]. Notably, Entrance_2 has a setup with a 90° corner and two gates. Like the prior framework, this dataset contains map patches categorized as pushing and non-pushing. The final dataset comprises 2,257 pushing map patches and 1,684 non-pushing map patches. Additionally, to evaluate the overall quality of the proposed framework, measuring its computational time is also required. For this purpose, to simulate acquiring the actual inputs, we created a live video stream of the crowded event entrances using video recordings of entrances and a virtual camera on a web client. In this context, we changed the camera's input to the video recordings. Then, the cloud-based framework was executed to display the live camera stream on the web client and directly annotate pushing patches on the live stream. Simultaneously, the computational time has been calculated for each component in the framework.

The experimental findings indicate that the proposed framework detected pushing patches from the live camera stream of crowded event entrances with 87 % accuracy rate within a reasonable time delay.

2.3 Publication III:

A Novel Voronoi-based Convolutional Neural Network Framework for Pushing Person Detection in Crowd Videos

While the first two frameworks focused on identifying pushing behavior at the patch level, publication III introduced a novel framework for detecting such behavior within video recordings of crowds at the microscopic level. Analyzing the dynamics of pushing behavior at the microscopic level can yield more precise insights into crowd behavior and interactions, such as knowledge into transferring such behavior to neighbors. This accurate understanding helps create more effective crowd management strategies and improves the design of public spaces and events, leading to safer and more comfortable crowds. Additionally, it can be asserted that patch-level detection is not facilitative for modeling purposes. For a model to accurately replicate pushing behavior, it is imperative to comprehend the phenomena at the individual level. This approach allows for the development of more reliable and accurate models.

The new framework employs a combination of DL algorithm and Voronoi Diagram technique to address the complex task of microscopic-level detection of pushing behavior in crowd videos. Additionally, it uses pedestrian trajectory data as an auxiliary input source. The framework comprises two main components: Feature Extraction and Labeling. The first component aims to extract deep features from each individual's behavior, which the Labeling component can use to classify pedestrians as pushing and non-pushing. The Feature Extraction

component consists of two modules: Voronoi [59]-based local region extraction and EfficientNetV1B0-based deep feature extraction. In the first module, a new Voronoi-based method was developed for determining the local regions associated with each person in the input video over time. Subsequently, these regions are fed into EfficientNetV1B0-based CNN in the second module to extract the deep features of each person. Each local region encapsulates the crowd dynamics around a single person, reflecting potential interactions between that person and his neighbors. This region plays the primary role in guiding the proposed framework to focus on microscopic behavior. On the other hand, the second component employs a fully connected layer with a Sigmoid activation function to analyze the extracted deep features and annotate the individuals involved in pushing within the video.

The framework was trained and evaluated on a new dataset created for this purpose. The dataset consists of a training set, a validation set for the learning process, and two test sets for the evaluation process. The samples of these sets are either pushing or non-pushing local regions. In this context, the pushing region means its pedestrian contributes to pushing, while the non-pushing sample indicates that its individual follows the social norm or queuing. The dataset was created from six real-world videos selected from the data archive hosted by Forschungszentrum Jülich [57, 58]. The training, validation, and the first test set use five specific videos of experiments (named 110, 150, 270, 280, and Entrance_2) for generating their samples. Moreover, the second test set employs a video of experiment number 50 [57] for its samples. The final dataset contains 3,384 pushing local regions and 8,994 non-pushing local regions.

The experimental findings demonstrate that the proposed framework achieved an accuracy of 85 % on the first test set and 82 % on the second test set.

2.4 Publication IV:

On the Exploitation of GPS-based Data for Real-time Visualization of Pedestrian Dynamics in Open Environments

In addition to the above studies, publication IV introduced an intelligent and low-cost GPS-based system for online and real-time visualization of pedestrian dynamics at open environments using GPS data. This system aims to mitigate pushing behavior and associated risks by helping reduce dense areas that can easily lead to pushing, especially in large and open events. It provides real-time visualized maps that capture ongoing pedestrian movement, as well as heat maps to track crowded areas. These maps empower event organizers and security teams to regulate crowd flow effectively, thereby maintaining low-density conditions. Furthermore, it enables attendees to find less crowded areas, further alleviating crowd density.

Publication IV introduced the system based on a custom mobile application, a built-in GPS sensor in smartphones, a web-based system, and visualization techniques. Notably, the organizers, security teams, and attendees can conveniently view the maps online via a web browser. The system comprises five main components:

1. **Mobile-based Data Acquisition.** A custom mobile application installed on client smartphones identifies their location using built-in GPS sensors in their smartphones. The application then transmits this GPS data to a

dedicated web server in real-time, prioritizing user privacy by only collecting location data (latitude and longitude) within the event area.

2. **Pre-Processing of GPS Data.** This component filters incoming location data based on two criteria: whether the data falls within the event's geographical bounds and its accuracy within a five-meter.

3. **Data Repository Component.** The received data is stored in two separate database tables (archive and live). The archive table serves as a comprehensive repository for all location data, while the live table maintains only the current locations for real-time visualization.

4. **Post-Processing of GPS Data.** This component keeps the live table up-to-date by storing only the current locations of active attendees. It also prepares the data in JSON format for the final component.

5. **Real-time Visualization.** This last component visualizes the real-time location data on a web browser using point maps and heat maps. Point maps display the individual current locations of attendees, while heat maps categorize regions within the event based on crowd density: high, medium, and low.

To assess the proposed system, a prototype was developed and hosted on a web server, along with a mobile application developed for Android OS. The prototype was then tested using both real-world and simulation experiments. The real-world experiment was conducted in an open area of 37,000 square meters, employing multiple smartphones with the installed application to collect and transmit real-time pedestrian location data to a web server. The main objectives of this experiment are to: 1) Evaluate the system's ability to automatically collect pedestrians' position data. 2) Measure the accuracy of the collected GPS-based data in open areas. 3) Calculate the computational time of each component within the prototype. On the other hand, a series of simulation experiments were carried out to measure the system prototype's computational efficiency across a range of position counts, from 100 to 5,000.

The experimental results revealed that the prototype: 1) Successfully captured 98.8 % of the pedestrians' current positions at one-second intervals. 2) Achieved real-time data collection in an open environment within four meters of horizontal accuracy, paired with real-time visualization capabilities.

Remarkably, the system offers real-time visualization of crowds in open areas without requiring specialized equipment or sensors, making it cost-effective and highly scalable for large venues. This means that the system is an effective solution for helping manage crowds and minimize pushing behavior and its risks in large open areas where camera coverage is challenging due to size or obstacles.

Chapter 3

Discussion and Outlook

Within the scope of this thesis, a cutting-edge AI-based framework has been developed to analyze video recordings and live streams of crowds, explicitly targeting the detection of pushing behavior. The progress made in this thesis not only highlighted certain limitations of the current work but also introduced open topics and suggestions for further research directions in the future. This chapter discusses some of these limitations, topics, and suggested solutions.

The current deep learning models used in the first three publications have shown encouraging results. However, their training and evaluation relied on limited datasets collected from a maximum of seven real-world video experiments captured by static and top-view cameras, potentially leading to an overfitting issue. To enhance the models' generality, employing transfer learning techniques and expanding the datasets with additional scenarios could be useful.

In publication III, the proposed Voronoi-based CNN approach currently relies on pedestrian trajectory data to identify individuals involved in pushing behavior within the input video. However, such data is often unavailable in real-life scenarios. To address this limitation, we plan to integrate YOLOv8 into the existing approach. This integration requires an annotated dataset for pedestrian heads in dense crowds to train the YOLOv8 model. Such a model aims to help extract each individual's local region within the video, eliminating the need for trajectory data.

At the same line, the approach presented in publication III currently processes one frame per second to identify pushing behavior at the microscopic level. Utilizing a sequence of frames has the potential to capture more comprehensive feature sets relevant to pushing behavior compared to a single-frame approach. To achieve this, we plan to integrate YOLOv8, CNN, and Long Short-Term Memory (LSTM) models into a unified approach. This enhanced approach aims to extract relevant features from a sequence of frames in the input video, thereby improving the detection of pushing behavior at the microscopic level. In such a suggested approach, CNN and LSTM can be trained using the current labeled pushing data, while YOLOv8 requires an annotated dataset for pedestrian heads in dense crowds.

Deep Learning (DL) algorithms, particularly CNN models, have successfully detected pushing behavior in crowd videos. However, this detection capability relies on labeled datasets required for training the models. Manually establishing ground truths from real-world video experiments for pushing datasets is challenging. As a solution, we recommend equipping each experiment participant with specialized sensors to gather data that can aid in generating the ground truths accurately.

The proposed GPS-based data system in publication IV was designed to work exclusively with Android-based smartphones. To extend the system's reach to a

broader audience using diverse mobile platforms, developing an iOS-based mobile application is required. Additionally, further experiments are essential to evaluate the system's performance. Furthermore, the system has limitations in indoor settings due to the inherent constraints of GPS technology. However, by integrating this system with AI and video surveillance, we potentially automate and enhance the identification and visualization of crowd levels in outdoor and indoor areas, along with pushing detection in those areas.

In general, timely and accurately detecting unmarked pedestrian heads in dense crowd videos is crucial for practical visual analysis. Therefore, developing pretrained, adaptable models for real-life scenarios involving dense crowds is essential.

Real-world experiments of crowds are often conducted to study and understand crowd behaviors, where it is possible to extract valuable information from crowds under specific experimental conditions. However, these studies have limitations: 1) The results achieved under experimental conditions might differ from real-life scenarios. 2) Some real-life scenarios are challenging to carry out experimentally. 3) Real-world experiments can be costly. Given these constraints and the availability of real-life crowd videos, developing automated methods to glean insights from real-life crowds that help understand their dynamics would be beneficial. Additionally, these methods must prioritize and address privacy considerations.

References

[1] K. Saravanan and A. Z. Kouzani, "Advancements in on-device deep neural networks," *Information*, vol. 14, no. 8, p. 470, 2023.

[2] H. Pallathadka, E. H. Ramirez-Asis, T. P. Loli-Poma, K. Kaliyaperumal, R. J. M. Ventayen, and M. Naved, "Applications of artificial intelligence in business management, e-commerce and finance," *Materials Today: Proceedings*, vol. 80, pp. 2610–2613, 2023.

[3] M. Tsuneki, "Deep learning models in medical image analysis," *Journal of Oral Biosciences*, vol. 64, no. 3, pp. 312–320, 2022.

[4] E. Dilek and M. Dener, "Computer vision applications in intelligent transportation systems: a survey," *Sensors*, vol. 23, no. 6, p. 2938, 2023.

[5] M. Javaid, A. Haleem, I. H. Khan, and R. Suman, "Understanding the potential applications of artificial intelligence in agriculture sector," *Advanced Agrochem*, vol. 2, no. 1, pp. 15–30, 2023.

[6] A. Entezari, A. Aslani, R. Zahedi, and Y. Noorollahi, "Artificial intelligence and machine learning in energy systems: A bibliographic perspective," *Energy Strategy Reviews*, vol. 45, p. 101017, 2023.

[7] A. Rachha and M. Seyam, "Explainable ai in education: Current trends, challenges, and opportunities," *SoutheastCon 2023*, pp. 232–239, 2023.

[8] Y. Zhu, K. Ni, X. Li, A. Zaman, X. Liu, and Y. Bai, "Artificial intelligence aided crowd analytics in rail transit station," *Transportation Research Record*, p. 03611981231175156, 2023.

[9] C. Wang and W. Weng, "Study on the collision dynamics and the transmission pattern between pedestrians along the queue," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2018, no. 7, p. 073406, 2018.

[10] A. Sieben and A. Seyfried, "Inside a life-threatening crowd: Analysis of the love parade disaster from the perspective of eyewitnesses," *arXiv preprint arXiv:2303.03977*, 2023.

[11] S. Feldmann and J. Adrian, "Forward propagation of a push through a row of people," *Safety science*, vol. 164, p. 106173, 2023.

[12] R. E. Ocejo and S. Tonnelat, "Subway diaries: How people experience and practice riding the train," *Ethnography*, vol. 15, no. 4, pp. 493–515, 2014.

[13] E. Üsten, H. Lügering, and A. Sieben, "Pushing and non-pushing forward motion in crowds: A systematic psychological observation method for rating

individual behavior in pedestrian dynamics," *Collective Dynamics*, vol. 7, pp. 1–16, 2022.

[14] V. Filingeri, K. Eason, P. Waterson, and R. Haslam, "Factors influencing experience in crowds–the participant perspective," *Applied ergonomics*, vol. 59, pp. 431–441, 2017.

[15] X. Li, X. Xu, J. Zhang, K. Jiang, W. Liu, R. Yi, and W. Song, "Experimental study on the movement characteristics of pedestrians under sudden contact forces," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 6, p. 063406, 2021.

[16] C. Wang, L. Shen, and W. Weng, "Experimental study on individual risk in crowds based on exerted force and human perceptions," *Ergonomics*, vol. 63, no. 7, pp. 789–803, 2020.

[17] T. Metivet, L. Pastorello, and P. Peyla, "How to push one's way through a dense crowd," *Europhysics Letters*, vol. 121, no. 5, p. 54003, 2018.

[18] A. Budiman, R. A. Yaputera, S. Achmad, A. Kurniawan *et al.*, "Student attendance with face recognition (lbph or cnn): Systematic literature review," *Procedia Computer Science*, vol. 216, pp. 31–38, 2023.

[19] W. Lu, C. Lan, C. Niu, W. Liu, L. Lyu, Q. Shi, and S. Wang, "A cnn-transformer hybrid model based on cswin transformer for uav image object detection," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2023.

[20] C. Direkoglu, "Abnormal crowd behavior detection using motion information images and convolutional neural networks," *IEEE Access*, vol. 8, pp. 80 408–80 416, 2020.

[21] N. C. Tay, T. Connie, T. S. Ong, K. O. M. Goh, and P. S. Teh, "A robust abnormal behavior detection method using convolutional neural network," in *Computational Science and Technology*. Springer, 2019, pp. 37–47.

[22] E. Duman and O. A. Erdem, "Anomaly detection in videos using optical flow and convolutional autoencoder," *IEEE Access*, vol. 7, pp. 183 914–183 923, 2019.

[23] M. J. Roshtkhari and M. D. Levine, "An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions," *Computer vision and image understanding*, vol. 117, no. 10, pp. 1436–1452, 2013.

[24] G. Singh, A. Khosla, and R. Kapoor, "Crowd escape event detection via pooling features of optical flow for intelligent video surveillance systems," *International Journal of Image, Graphics and Signal Processing*, vol. 10, no. 10, p. 40, 2019.

[25] M. George, C. Bijitha, and B. R. Jose, "Crowd panic detection using autoencoder with non-uniform feature extraction," in *2018 8th International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2018, pp. 11–15.

[26] G. L. Santos, P. T. Endo, K. H. d. C. Monteiro, E. d. S. Rocha, I. Silva, and T. Lynn, "Accelerometer-based human fall detection using convolutional neural networks," *Sensors*, vol. 19, no. 7, p. 1644, 2019.

[27] A. Mehmood, "Lightanomalynet: A lightweight framework for efficient abnormal behavior detection," *Sensors*, vol. 21, no. 24, p. 8501, 2021.

[28] X. Zhang, Q. Zhang, S. Hu, C. Guo, and H. Yu, "Energy level-based abnormal crowd behavior detection," *Sensors*, vol. 18, no. 2, p. 423, 2018.

[29] J. F. Kooij, M. C. Liem, J. D. Krijnders, T. C. Andringa, and D. M. Gavrila, "Multi-modal human aggression detection," *Computer Vision and Image Understanding*, vol. 144, pp. 106–120, 2016.

[30] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI magazine*, vol. 27, no. 4, pp. 12–12, 2006.

[31] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[32] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[33] H. J. Aerts, E. R. Velazquez, R. T. Leijenaar, C. Parmar, P. Grossmann, S. Carvalho, J. Bussink, R. Monshouwer, B. Haibe-Kains, D. Rietveld *et al.*, "Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach," *Nature communications*, vol. 5, no. 1, pp. 1–9, 2014.

[34] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[35] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[36] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[37] ——, "Efficientnetv2: Smaller models and faster training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 096–10 106.

[38] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

[39] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision*. Springer, 2020, pp. 402–419.

[40] H. Ammar and A. Cherif, "Deeprod: a deep learning approach for real-time and online detection of a panic behavior in human crowds," *Machine Vision and Applications*, vol. 32, no. 3, pp. 1–15, 2021.

[41] A. Bahamid and A. Mohd Ibrahim, "A review on crowd analysis of evacuation and abnormality detection based on machine learning systems," *Neural Computing and Applications*, pp. 1–15, 2022.

[42] X. Zhang, X. Shu, and Z. He, "Crowd panic state detection using entropy of the distribution of enthalpy," *Physica A: Statistical Mechanics and its Applications*, vol. 525, pp. 935–945, 2019.

[43] H. Fradi and J.-L. Dugelay, "Towards crowd density-aware video surveillance applications," *Information Fusion*, vol. 24, pp. 3–15, 2015.

[44] T. Alafif, A. Hadi, M. Allahyani, B. Alzahrani, A. Alhothali, R. Alotaibi, and A. Barnawi, "Hybrid classifiers for spatio-temporal abnormal behavior detection, tracking, and recognition in massive hajj crowds," *Electronics*, vol. 12, no. 5, p. 1165, 2023.

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

[46] M. Xu, X. Yu, D. Chen, C. Wu, and Y. Jiang, "An efficient anomaly detection system for crowded scenes using variational autoencoders," *Applied Sciences*, vol. 9, no. 16, p. 3337, 2019.

[47] S. Smeureanu, R. T. Ionescu, M. Popescu, and B. Alexe, "Deep appearance features for abnormal behavior detection in video," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 779–789.

[48] Z. Ilyas, Z. Aziz, T. Qasim, N. Bhatti, and M. F. Hayat, "A hybrid deep network based approach for crowd anomaly detection," *Multimedia Tools and Applications*, pp. 1–15, 2021.

[49] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.

[50] Y. Hu, "Design and implementation of abnormal behavior detection based on deep intelligent analysis algorithms in massive video surveillance," *Journal of Grid Computing*, vol. 18, no. 2, pp. 227–237, 2020.

[51] A. A. Almazroey and S. K. Jarraya, "Abnormal events and behavior detection in crowd scenes based on deep learning and neighborhood component analysis feature selection," in *Joint European-US Workshop on Applications of Invariance in Computer Vision*. Springer, 2020, pp. 258–267.

[52] S. Zhou, W. Shen, D. Zeng, M. Fang, Y. Wei, and Z. Zhang, "Spatial–temporal convolutional neural networks for anomaly detection and localization in crowded scenes," *Signal Processing: Image Communication*, vol. 47, pp. 358–368, 2016.

[53] H. Lügering, D. Tepeli, and A. Sieben, "It's (not) just a matter of terminology: Everyday understanding of "mass panic" and alternative terms," *Safety science*, vol. 163, p. 106123, 2023.

[54] D. F. Tom Runia, "Optical flow visualization," 2020. [Online]. Available: https://github.com/tomrunia/OpticalFlow_Visualization

[55] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International journal of computer vision*, vol. 92, no. 1, pp. 1–31, 2011.

[56] J. Adrian, A. Seyfried, and A. Sieben, "Crowds in front of bottlenecks at entrances from the perspective of physics and social psychology," *Journal of the Royal Society Interface*, vol. 17, no. 165, p. 20190871, 2020.

[57] "Crowds in front of bottlenecks from the perspective of physics and social psychology," http://doi.org/10.34735/ped.2018.1, Pedestrian Dynamics Data Archive hosted by the Forschungszentrum Jülich, 2018.

[58] "Entrance 2, entry with guiding barriers (corridor setup)," http://doi.org/10.34735/ped.2013.1, Pedestrian Dynamics Data Archive hosted by the Forschungszentrum Jülich, 2013.

[59] P. J. Green and R. Sibson, "Computing dirichlet tessellations in the plane," *The computer journal*, vol. 21, no. 2, pp. 168–173, 1978.

Publication I

A Hybrid Deep Learning and Visualization Framework for Pushing Behavior Detection in Pedestrian Dynamics

This article has been published as Alia, Ahmed, Mohammed Maree, and Mohcine Chraibi, 2022. "A hybrid deep learning and visualization framework for pushing behavior detection in pedestrian dynamics." Sensors, 22(11), p.4040.

Author's Contributions

Conceptualization: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
Methodology: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
Software: **Ahmed Alia**
Validation: **Ahmed Alia**
Formal Analysis: **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
Visualization: **Ahmed Alia**
Data Curation: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
writing—review and editing **Ahmed Alia**, Mohammed Maree, Mohcine Chraibi
GitHub Repository Creation: **Ahmed Alia**

GitHub repository: `https://github.com/PedestrianDynamics/DL4PuDe`

Publication DOI: `https://doi.org/10.3390/s22114040`

Note: The EfficientNet-B0 in this publication is equivalent to the EfficientNetV1B0 in other parts of the thesis.

A Hybrid Deep Learning and Visualization Framework for Pushing Behavior Detection in Pedestrian Dynamics

Ahmed Alia[1,2,3], Mohammed Maree[4*], Mohcine Chraibi[1*]

[1]Institute for Advanced Simulation, Forschungszentrum Jülich, 52425 Jülich, Germany.

[2] Computer Simulation for Fire Protection and Pedestrian Traffic, University of Wuppertal, 42285 Wuppertal, Germany.

[3] Department of Management Information Systems, Faculty of Engineering and Information Technology, An-Najah National University, Nablus, Palestine.

[4] Department of Information Technology, Faculty of Engineering and Information Technology, Arab American University, Jenin, Palestine.

Corresponding authors: mohammed.maree@aaup.edu, m.chraibi@fz-juelich.de

Abstract Crowded event entrances could threaten the comfort and safety of pedestrians, especially when some pedestrians push others or use gaps in crowds to gain faster access to an event. Studying and understanding pushing dynamics leads to designing and building more comfortable and safe entrances. Researchers—to understand pushing dynamics—observe and analyze recorded videos to manually identify when and where pushing behavior occurs. Despite the accuracy of the manual method, it can still be time-consuming, tedious, and hard to identify pushing behavior in some scenarios. In this article, we propose a hybrid deep learning and visualization framework that aims to assist researchers in automatically identifying pushing behavior in videos. The proposed framework comprises two main components: (i) deep optical flow and wheel visualization; to generate motion information maps. (ii) A combination of an EfficientNet-B0-based classifier and a false reduction algorithm for detecting pushing behavior at the video patch level. In addition to the framework, we present a new patch-based approach to enlarge the data and alleviate the class imbalance problem in small-scale pushing behavior datasets. Experimental results (using real-world ground truth of pushing behavior videos) demonstrate that the proposed framework achieves an 86% accuracy rate. Moreover, the EfficientNet-B0-based classifier outperforms baseline CNN-based classifiers in terms of accuracy.

Keywords: Deep Learning; Convolutional Neural Network, EfficientNet-B0-based Classifier, Image Classification, Crowd Behavior Analysis, Pushing Behavior Detection, Motion Information Maps, Deep Optical Flow

1 Introduction

In entrances of large-scale events, pedestrians either follow the social norm of queuing or force some pushing behavior to gain faster access to the events [1]. Pushing behavior in this context is an unfair strategy that some pedestrians use to move quickly and enter an event faster. This behavior involves pushing others and moving forward quickly by using one's arms, shoulders, elbows, or upper body, as well as using gaps among crowds to overtake and gain faster access [2, 3]. Pushing behavior, as opposed to queuing behavior, can increase the density of crowds [4].

Consequently, such behavior may lead to threatening the comfort and safety of pedestrians, resulting in dangerous situations [5]. Thus, understanding pushing behavior, what causes it, and the consequences are crucial, especially when designing and constructing comfortable and safe entrances [6, 1]. Conventionally, researchers have attempted to study pushing behavior manually by observing and identifying pushing cases among video recordings of crowded events. For instance, Lügering et al. [3] proposed a rating system on forward motions in crowds to understand when, where, and why pushing behavior appears. The system relies on two trained observers to classify the behaviors of pedestrians over time in a video (the behavior is classified into either pushing or non-pushing categories). In this context, each category includes two gradations: mild and strong for pushing, and falling behind and just walking for non-pushing. For more details on this system, we refer the reader to [3]. To carry out their tasks, the observers analyzed top-view video recordings using pedestrian trajectory data and PeTrack software [7]. However, this manual rating procedure is time-consuming, tedious, and requires a lot of effort by observers, making it hard to identify pushing behavior, specifically when the number of videos and pedestrians in each video increase [3]. Consequently, there is a pressing demand to develop an automatic and reliable framework to identify when and where pushing behavior appears in videos. This article's main motivation is to help social psychologists and event managers identify pushing behavior in videos. However, automatic pushing behavior detection is highly challenging due to several factors, including diversity in pushing behavior, the high similarity and overlap between pushing and non-pushing behaviors, and the high density of crowds at event entrances.

According to a computer vision perspective, automatic pushing behavior detection belongs to the video-based abnormal human behavior detection field [8]. Several human behaviors have been addressed, including walking in the wrong direction [9], running away [10], sudden people grouping or dispersing [11], human falls [12], suspicious behavior, violent acts [13], abnormal crowds [14], hitting, pushing, and kicking [15]. It is worth highlighting that pushing as defined in [15] is different from the "pushing behavior" term in this article. In [15], pushing is a strategy used for fighting, and the scene contains only up to four persons. To the best of our knowledge, no previous studies have automatically identified pushing behavior for faster access from videos.

With the rapid development in deep learning, CNN has achieved remarkable performance in animal [16, 17] and human [13, 18] behavior detection. The main advantage of CNN is that it directly learns the useful features and classification from data without any human effort [19]. However, CNN requires a large training dataset to build an accurate classifier [20, 21]. Unfortunately, this requirement is unavailable in most human behaviors. To alleviate this limitation, several studies have used a combination of CNN and handcrafted feature descriptors [22, 23]. The hybrid-based approaches use descriptors to extract valuable information. Then, CNN automatically models abnormal behavior from the extracted information [24, 25]. Since labeled data for pushing behavior are scarce, the hybrid-based approaches could be more suitable for automatic pushing behavior detection. Unfortunately, the existing approaches are inefficient for pushing behavior detection [22]. Their main limitations are: (1) their descriptors do not work well to extract accurate information from dense crowds due to occlusions, or they cannot extract the needed information for pushing behavior representation [22, 26]; (2) Some used CNN architectures are not efficient enough to deal with

the high similarity between pushing and non-pushing behaviors (high inter-class similarity) and the increased diversity in pushing behavior (intra-class variance), leading to misclassification [25, 26].

To address the above limitations, we propose a hybrid deep learning and visualization framework for automatically detecting pushing behavior at the patch level in videos. The proposed framework exploits video recordings of crowded entrances captured by a top-view static camera, and comprises two main components: (1) motion information extraction aims to generate motion information maps (MIMs) from the input video. A MIM is an image that contains useful information for pushing behavior representation. This component divides each MIM into several MIM patches, making it easier to see where pedestrians are pushing. For this purpose, recurrent all-pairs field transforms (RAFT) [27] (one of the newest and most promising deep optical flow methods) and the wheel visualization method [28, 29] are combined; (2) The pushing patch annotation adapts the EfficientNet-B0-based CNN architecture (the EfficientNet-B0-based CNN [30] is an effective and simple architecture in the EfficientNet family proposed by Google in 2019, achieving the highest accuracy in the ImageNet dataset [31]) to build a robust classifier, which aims to select the relevant features from the MIM patches and label them into pushing and non-pushing categories. We utilized a false reduction algorithm to enhance the classifier's predictions. Finally, the component outputs pushing the annotated video showed when and where the pushing behaviors appeared.

We summarize the main contributions of this article as follows:

1. To the best of our knowledge, we proposed the first framework dedicated to automatically detecting when and where pushing occurs in videos.
2. An integrated EfficientNet-B0-based CNN, RAFT, and wheel visualization within a unique framework for pushing behavior detection.
3. A new patch-based approach to enlarge the data and alleviate the class imbalance problem in the used video recording datasets.
4. To the best of our knowledge, we created the first publicly available dataset to serve this field of research.
5. A false reduction algorithm to improve the accuracy of the proposed framework.

The rest of this paper is organized as follows: Section 2 reviews the related work of video-based abnormal human behavior detection. In Section 3, we introduce the proposed framework. A detailed description of dataset preparation is given in Section 4. Section 5 discusses experimental results and comparisons. Finally, the conclusion and future work are summarized in Section 6.

2 Related Work

Existing video-based abnormal human behavior detection methods can be generally classified into object-based and holistic-based approaches [25, 26]. Object-based methods consider the crowd as an aggregation of several pedestrians and rely on detecting and tracking each pedestrian to define abnormal behavior [32]. Due to occlusions, these approaches face difficulties in dense crowds [33, 34]. Alternatively, holistic-based approaches deal with crowds as single entities. Thus,

they analyze the crowd itself to extract useful information and detect abnormal behaviors [25, 34, 24]. In this section, we briefly review some holistic-based approaches related to the context of this research. Specifically, the approaches are based on CNN or a hybrid of CNN and handcrafted feature descriptors.

Tay et al. [35] presented a CNN-based approach to detect abnormal actions from videos. The authors trained the CNN on normal and abnormal behaviors to learn the features and classification. As mentioned before, this type of approach requires a large dataset with normal and abnormal behaviors. To address the lack of large datasets with normal and abnormal behaviors, some researchers applied a one-class classifier using datasets of normal behaviors. Obtaining or preparing a dataset with only normal behaviors is easier than a dataset with normal and abnormal behaviors [36, 34]. The main idea of the one-class classifier is to learn from the normal behaviors only; to define a class boundary between the normal and not defined (abnormal) classes. Sabokrou et al. [36] utilized a new pre-trained CNN to extract the motion and appearance information from crowded scenes. Then, they used a one-class Gaussian distribution to build the classifier from datasets with normal behaviors. In the same way, the authors of [34, 37] used datasets of normal behaviors to develop their one-class classifiers. Xu et al. used a convolutional variational autoencoder to extract features in [34]. Then, multiple Gaussian models were employed to predict abnormal behavior. Ref. [37] adopted a pre-trained CNN model for feature extraction and a one-class support vector machines to predict abnormal behavior. In another work, Ilyas et al. [24] used pre-trained CNN along with a gradient sum of the frame difference to extract relevant features. Afterward, three support vector machines were trained on normal behavior to detect abnormal behavior. In general, the one-class classifier is popular when the abnormal behavior or target behavior class is rare or not well-defined [38]. In contrast, the pushing behavior is well-defined and not rare, especially in high-density and competitive scenarios. Moreover, this type of classifier considers the new normal behavior as abnormal.

In order to overcome the drawback of CNN-based approaches and one-class classifier approaches, several studies used a hybrid-based approach with a multi-class classifier. Duman et al. [22] employed the classical Farnebäck optical flow method [23] and CNN to identify abnormal behavior. The authors used Farnebäck and CNN to extract the direction and speed information. Then, they applied a convolutional long short-term memory network for building the classifier. In [39], the authors used a histogram of gradient and CNN to extract the relevant features, while a least-square support vector was employed for classification. In a similar line of the hybrid approaches, Direkoglu [25] combined the Lucas–Kanade optical flow method and CNN to extract the relevant features and detect "escape and panic behaviors". Almazroey et al. [26] employed mainly a Lucas–Kanade optical flow, pre-trained CNN, and feature selection (neighborhood component analysis) methods to select the relevant features. The authors then applied a support vector machine to generate a trained classifier. Zhou et al. [40] presented a CNN method for detecting and localizing anomalous activities. The study integrated optical flow with a CNN for feature extraction and it used a CNN for the classification task.

In summary, hybrid-based approaches have shown better accuracy than CNN-based approaches on small datasets [41]. Unfortunately, the reviewed hybrid-based approaches are inefficient for dense crowds and pushing behavior detection due to (1) their feature extraction parts being inefficient for dense crowds; (2) The

reviewed approaches cannot extract all of the required information for pushing behavior representation; (3) Their classifiers are not efficient enough toward pushing behavior detection. Hence, the proposed framework combines the power of supervised EfficientNet-B0-based CNN, RAFT, and wheel visualization methods to solve the above limitations. The RAFT method works well for estimating optical flow vectors from dense crowds. Moreover, the integration of RAFT and wheel visualization helps to simultaneously extract the needed information for pushing behavior representation. Finally, the adapted EfficientNet-B0-based binary classifier detects distinct features from the extracted information and identifies pushing behavior at the patch level.

3 The Proposed Framework

This section describes the proposed framework for automatic pushing behavior detection at the video patch level. As shown in Figure 1, there are two main components: motion information extraction and pushing patches annotation. The first component extracts motion information from input video recordings, which is further exploited by the pushing patch annotation component to detect and localize pushing behavior, producing pushing annotated video. The following subsections discuss both components in more detail.

3.1 Motion Information Extraction

This component employs RAFT and wheel visualization to estimate and visualize the crowd motion from the input video at the patch level. The component has two modules, a deep optical flow estimator and a MIM patch generator.

The deep optical flow estimator relies on RAFT to calculate the optical flow vectors for all pixels between two frames. RAFT was introduced in 2020; it is a promising approach for dense crowds because it reduces the effect of occlusions on optical flow estimation [27]. RAFT is based on a composition of CNN and recurrent neural network architectures. Moreover, RAFT has strong cross-dataset generalization and its pre-trained weights are publicly available. For additional information about RAFT, we refer the reader to [27]. This module is based on the RAFT architecture with its pre-trained weights along with three inputs, which are a video of crowded event entrances, the rotation angle of the input video, and the region of interest (ROI) coordinates. To apply RAFT, firstly, we determine the bounding box of the entrance area (ROI) in the input video V. This process is based on user-defined left–top and bottom–right coordinates of the ROI in the pixel unit. Then, we extract the frame sequence $F = \{f_t \,|\, t = 1, 2, 3, \ldots, T\}$ with ROI only from V, where $f_t \in \mathbb{R}^{w \times h \times 3}$, w and h are the f_t width and height, respectively, 3 is the number of channels, t is the order of the frame f in V, and T is the total number of frames in V. After that, we rotate the frames (based on the user-defined *angle*) in F to meet the baseline direction of the crowd flow that is used in the classifier, which is from left to right. The rotation process is essential to improve the classifier accuracy because the classifier will be built by training the adapted EfficientNet-B0 on the crowd flow from left to right. Next, we construct from F the sequence of clips $C = \{c_i \,|\, i = 1, 2, 3, \ldots\}$ and c_i is defined as

$$c_i = \{f_{(i-1)\times(s-1)+1}, f_{(i-1)\times(s-1)+2}, \cdots, f_{(i-1)\times(s-1)+s}\}, \tag{1}$$

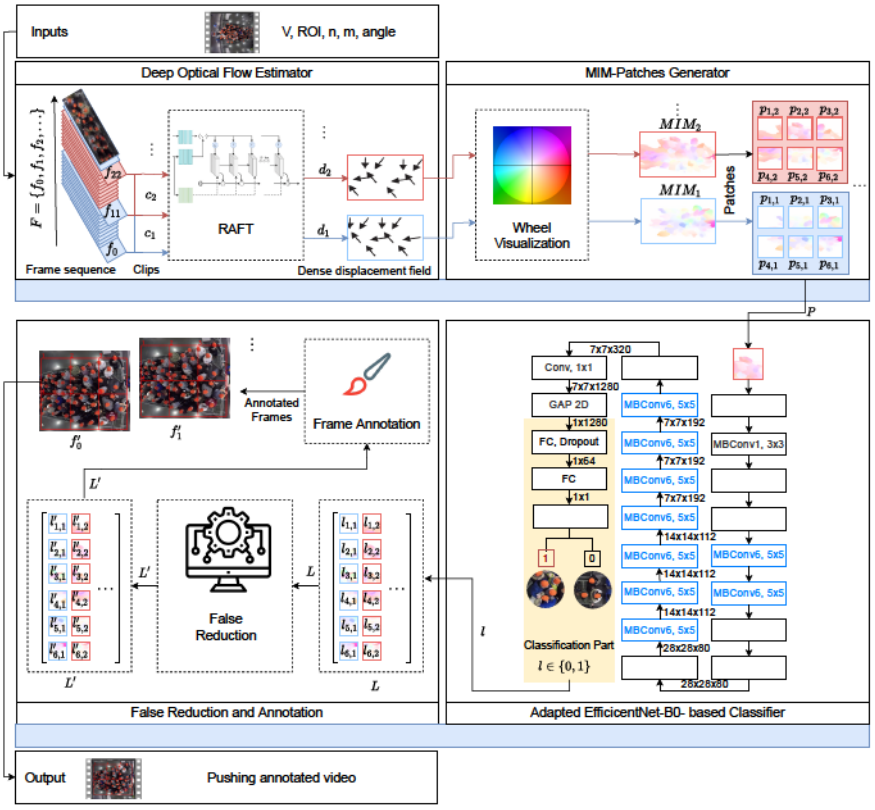

Figure 1: The architecture of the proposed automatic deep learning framework. n and m are two rows and three columns, respectively, for patching. Clip size s is 12 frames. MIM: motion information map. P: patch sequence. L: a matrix of all patches labels. L': an updated L by false reduction algorithm. V: the input video. ROI: region of interest (entrance area). $angle$: the rotation angle of the input video.

where s is the clip size. Finally, RAFT is applied on c_i, to calculate the dense displacement field d_i between $f_{(i-1)\times(s-1)+1}$ and $f_{(i-1)\times(s-1)+s}$. The output of RAFT of each pixel location $\langle x, y\rangle$ in c_i is a vector, as shown in.

$$\langle u_{\langle x,y\rangle}, v_{\langle x,y\rangle}\rangle_{c_i} = RAFT(\langle x, y\rangle_{c_i}), \tag{2}$$

where u and v are horizontal and vertical displacements of a pixel at the $\langle x, y\rangle$ location in c_i, respectively. This means d_i is a matrix of the vector values for the entire c_i, as described in

$$d_i = \left\{ \langle u_{\langle x,y\rangle}, v_{\langle x,y\rangle}\rangle_{c_i} \right\}_{(x,y)=(1,1)}^{(w,h)} \tag{3}$$

In summary, d_i is the output of this module and will act as the input of the MIM patch generator module.

The second module, the MIM patch generator, employs the wheel visualization to infer the motion information from each d_i. Firstly, the wheel visualization calculates the magnitude and the direction of each motion vector at each pixel $\langle x, y\rangle$ in d_i. Equations (3) and (4) are used to calculate the motion direction and magnitude, respectively. Then, from the calculated information, the wheel visualization generates MIM_i, where $MIM_i \in \mathbb{R}^{w\times h\times 3}$. In MIM, the color refers to the

motion direction and the intensity of the color represents the motion magnitude or speed. Figure 2 shows the color wheel scheme (b) and an example of MIM (MIM_{37}) (c) that is generated from c_{37}, whose first and last frames are f_{397} and f_{408}, respectively (a). c_{37} is taken from the experiment 270 [42].

$$\theta(\langle x,y\rangle)_{c_i} = \pi^{-1}\arctan(\frac{v_{\langle x,y\rangle}}{u_{\langle x,y\rangle}}) \tag{4}$$

$$mag(\langle x,y\rangle)_{c_i} = \sqrt{u^2_{\langle x,y\rangle} + v^2_{\langle x,y\rangle}} \tag{5}$$

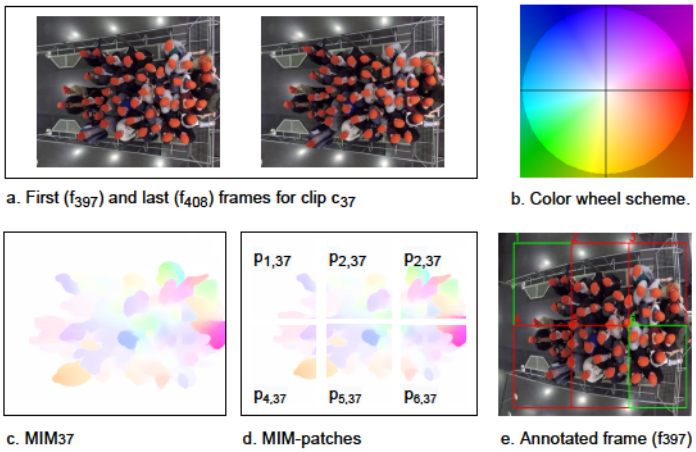

Figure 2: An illustration of two frames (experiment 270 [42]), color wheel scheme [29], MIM, MIM patches, and annotated frame. In sub-figure (e), red boxes refer to pushing patches, while green boxes represent non-pushing patches.

To detect pushing behavior at the patch level, the MIM patch generator divides each MIM_i into several patches. A user-defined row (n) and column (m) are used to split MIM_i into patches $\{p_{k,i} \in \mathbb{R}^{(w/m)\times(h/n)\times 3} \mid k = 1, 2, \cdots, n \times m\}$, where k is the order of the patch in MIM_i. Afterward, each $p_{k,i}$ is resized to a dimension of 224 × 224 × 3, which is the input size of the second component of the framework. For example, MIM_{37} in Figure 2c represents an entrance with dimensions 5 × 3.4 m on the ground, and it is divided into 2 × 3 patches $\{p_{k,37} \mid k \leq 6\}$ as given in Figure 2d. These patches are equal in pixels, whereas the area that is covered by them is not necessarily equal. The far patches from the camera cover a larger viewing area compared to close patches; because the far-away object has fewer pixels per m than a close object [43]. In Figure 2d, the average width and height of the $p_{k,37}$ are approximately 1.67 × 1.7 m.

In summary, the output of the motion information extraction component can be described as $P = \{p_{k,i} \in \mathbb{R}^{224\times224\times3} \mid k \leq n \times m \;\&\; i \leq |C|\}$, and will serve as input for the second component of the framework.

3.2 Pushing Patches Annotation

This component localizes the pushing patches in $c_i \in C$, annotates the patches in the first frame ($f_{(i-1)\times(s-1)+1}$) of each c_i, and stacks the annotated frame sequence $F' = \{f'_i \mid i = 1, 2, \ldots, |C|\}$ as a video. The Adapted EfficientNet-B0-based classifier and false reduction algorithm are the main modules of this component. In the following, we provide a detailed description.

The main purpose of the first module is to classify each $p_{k,i} \in P$ as pushing or non-pushing. The module is based on EfficientNet-B0 and real-world ground truth of pushing behavior videos. Unfortunately, the existing effective and simple EfficientNet-B0 is unsuitable for detecting pushing behavior because its classification is not binary. However, binary classification is required in our scenario. Therefore, we modify the classification part in EfficientNet-B0 to support a binary classification. The module in Figure 1 shows the architecture of the adapted EfficientNet-B0. Firstly, it executes a 3×3 convolution operation on the input image with dimensions of $224 \times 224 \times 3$. Afterwards, the next 16 mobile inverted bottleneck convolutions are used to extract the feature maps. The final stacked feature maps $\in \mathbb{R}^{7 \times 7 \times 1280}$, where 7 and 7 are the dimensions of each feature map, and 1280 is the number of feature maps. The following global average pooling2D (GAP) layer reduces the dimensions of the stacked feature maps into $1 \times 1 \times 1280$. For the binary classification, we employed a fully connected (FC) layer with a ReLU activation function and a dropout rate of 0.5 [44] before the final FC. The final layer operates as output with a sigmoid activation function to find the probability δ of the class of each $p_{k,i} \in P$.

In order to generate the trained classifier, we trained the adapted EfficientNet-B0 with pushing and non-pushing MIM patches. The labeled MIM patches were extracted from a real-world ground truth of pushing behavior videos, where the ground truth was manually created. In Sections 4 and 5.1, we show how to prepare the labeled MIM patches and train the classifier, respectively. Overall, after several empirical experiments (Section 5.2), the trained classifier on MIM patches of 12 frames produces the best accuracy results. Therefore, our framework uses 12 frames for the clip size (s). Moreover, the classifier uses the threshold for determining the label $l_{k,i}$ of the input $p_{k,i}$ as:

$$l_{k,i} = \begin{cases} 1 \text{ (pushing class)} & \text{if } \delta \geq 0.5 \\ 0 \text{ (non-pushing class)} & \text{if } \delta < 0.5 \end{cases} \tag{6}$$

Finally, the output of this module can be described as $L = \{l_{k,i} \in 0, 1 \mid k \leq n \times m \;\&\; i \leq |C|\}$ and will perform as the input of the next module.

In the second module, the false reduction algorithm aims to reduce the number of false predictions in L, which improves the overall accuracy of the proposed framework. Comparing the predictions (L) with the ground truth pushing, we notice that the time interval of the same behavior of each patch region could help improve the accuracy of the framework. We assume a threshold value of $\frac{34}{25}$ second. This value is based on visual inspection.

The example in Figure 3 visualizes the $\{l_{k,i} \mid k \leq 3 \;\&\; i \leq 4\}$ on the first frame of c_1, c_2, c_3, and c_4 in the video. Each c_i represents $\frac{12}{25}$ second. c_1 (Figure 3a) contains one false non-pushing, $p_{2,1}$, while the same region of the patch in $\{c_2, c_3, c_4\}$ is true pushing (Figure 3b–d). This means, we have two time intervals for $\{p_{2,i} \mid i \leq 4\}$. The first has one clip (c_1) (Figure 3a) with a duration of $\frac{12}{25}$ second, which is lesser than the defined threshold. The second time interval contains three clips ($\{c_2, c_3, c_4\}$), with durations equal to the threshold. Then the algorithm changes the prediction of $p_{2,1}$ to "pushing", while it confirms the predictions of $p_{2,2}, p_{2,3}$, and $p_{2,4}$. Algorithm 1 presents the pseudocode of the false reduction algorithm. Lines 2–8 show how to reduce the false predictions of the patches in $\{c_i \mid i \leq |c| - 2\}$. Then, lines 9–16 recheck the first two clips (c_1, c_2) to discover the false predictions that are not discovered by lines 2–8. After that, lines

Algorithm 1 False Reduction

Input:
$matrix[N, M] \leftarrow L$
Output:
L'

for $i \leftarrow 0, 1, \ldots, N$ **do**
  for $j \leftarrow 0, 1, \ldots, M-2$ **do**
    ▷ Excepting the last two clips
    if matrix[i, j] $\neq$ matrix[i, $j+1$] **then**
      if count(matrix[i, j] in matrix[i, $j+2$ to $j+4$]) > 1 **then**
        matrix[i, $j+1$]$\leftarrow$ not matrix[i, $j+1$]
      end if
    end if
  end for
  ▷ Recheck the first two clips
  if matrix[i, 0 to 2] is not identical **then**
    if matrix[i, 1] is not in matrix[i, 2 to 4] **then**
      matrix[i, 1]$\leftarrow$ not matrix[i, 1]
    end if
    if matrix[i, 0] not in matrix[i, 1 to 3] **then**
      matrix[i, 0]$\leftarrow$ not matrix[i, 0]
    end if
  end if
  ▷ For the last two clips
  if matrix[i, $M-1$] $\neq$ matrix[i, $M-2$] **then**
    if matrix[i, $M-1$] $\neq$ matrix[i, $M-3$] **then**
      matrix[i, $M-1$]$\leftarrow$ not matrix[i, $M-1$]
    end if
  end if
  if matrix[i,$M-1$] $\neq$ matrix[i,$M-2$] **then**
    if matrix[i,$M-1$] $=$ matrix[i,$M-3$] **then**
      matrix[i,$M-2$]$\leftarrow$ not matrix[i,$M-2$]
    end if
  end if
  if matrix[i, $M-1$] $=$ matrix[i, $M-2$] **then**
    if matrix[i, $M-1$] not in matrix[i, $M-5$ to $M-3$] **then**
      matrix[i, $M-1$]$\leftarrow$ not matrix[i, $M-1$]
      matrix[i ,$M-2$]$\leftarrow$ not matrix[i, $M-2$]
    end if
  end if
end for
$L' \leftarrow matrix$

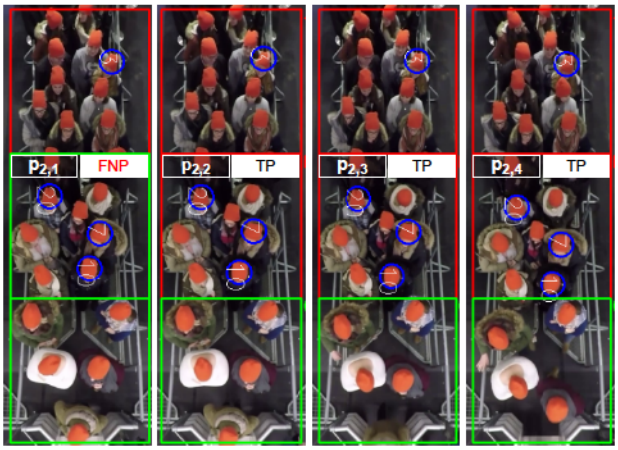

Figure 3: Examples of the visualized classifier predictions with ground truth pushing. The images represent the first frames $\{f_1, f_{12}, f_{23}, f_{34}\}$ of $\{c_1, c_2, c_3, c_4\}$ in a video, respectively; the video is for experiment 110 [42]. Red boxes: pushing patches. Green boxes: non-pushing patches. Blue circles: ground truth pushing. FNP: false non-pushing. TP: true pushing.

17–32 focus on the last two clips $\{c_{|C|-1}, c_{|C|}\}$. Finally, the updated L is stored in L', which can be described as $L' = \{l'_{k,i} \in 0, 1 \mid k \leq n \times m \;\&\; i \leq |C|\}$.

After applying the false reduction algorithm, the pushing patch annotation component based on L' identifies the regions of pushing patches on the first frame for each c_i to generate the annotated frame sequence F'. Finally, all annotated frames are stacked as a video, which is the final output of the proposed framework.

4 Datasets Preparation

This section prepares the required datasets for training and evaluating our classifier. In the following, firstly, four MIM-based datasets are prepared. Then, we present a new patch-based approach for enlarging the data and alleviating the class imbalance problem in the MIM-based datasets. Finally, the patch-based approach is applied to the datasets.

4.1 MIM-based Datasets Preparation

In this section, we prepare four MIM-based datasets using two clip sizes, Farnebäck and RAFT optical flow methods. Two clip sizes (12 and 25 frames) are used to study the impact of the period of motion on the classifier accuracy. Selecting a small clip size (s) for the MIM sequence (MIM^{Q_s}) leads to redundant and irrelevant information, while a large size leads to a few samples. Consequently, we chose 12 and 25 frames as the two clip sizes. The four datasets can be described as RAFT-$\text{MIM}^{Q_{12}}$, RAFT-$\text{MIM}^{Q_{25}}$, Farnebäck-$\text{MIM}^{Q_{12}}$, and Farnebäck-$\text{MIM}^{Q_{25}}$. For more clarity, the "RAFT-$\text{MIM}^{Q_{12}}$" term means that a combination of RAFT and wheel visualization is used to generate the $\text{MIM}^{Q_{12}}$. As mentioned before, the EfficientNet-B0 learns from MIM sequences generated based on RAFT. Therefore, RAFT-$\text{MIM}^{Q_{12}}$-based and RAFT-$\text{MIM}^{Q_{25}}$-based datasets play the primary role in training and evaluating the proposed classifier. Moreover, we create Farnebäck-$\text{MIM}^{Q_{12}}$-based and Farnebäck-$\text{MIM}^{Q_{25}}$-based datasets to evaluate the impact of RAFT on the classifier accuracy. The pipeline for preparing the datasets (Figure 4) is illustrated below.

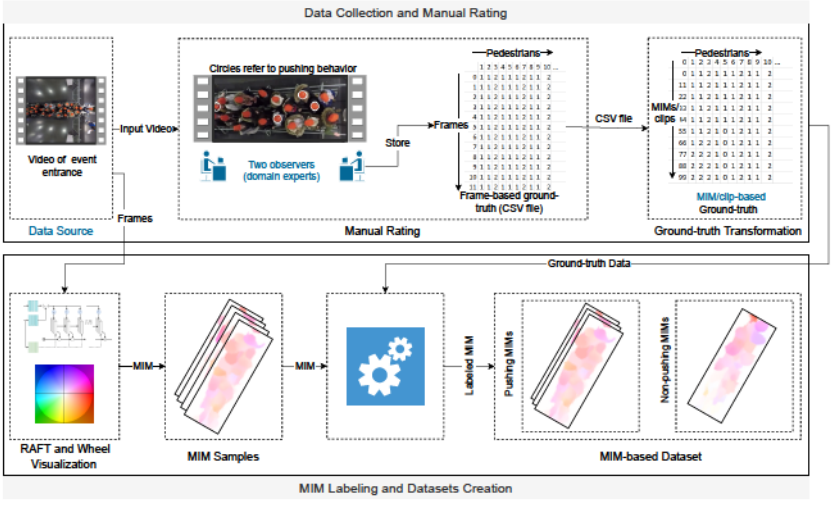

Figure 4: The pipeline of MIM-based dataset preparation.

4.1.1 Data Collection and Manual Rating

In this section, we discuss the data source and the manual rating methodology for the datasets. Five experiments were selected from the data archive hosted by the Forschungszentrum Jülich under CC Attribution 4.0 International license [42]. The experiments mimicked the crowded event entrances. The videos were recorded by a top-view static camera with a frame rate of 25 frames per second and 1920 × 1440 pixels resolution. In addition to the videos, parameters for video undistortion and trajectory data are available. In Figure 5, the left part sketches the experimental setup and Table 1 shows the different characteristics of the selected experiments.

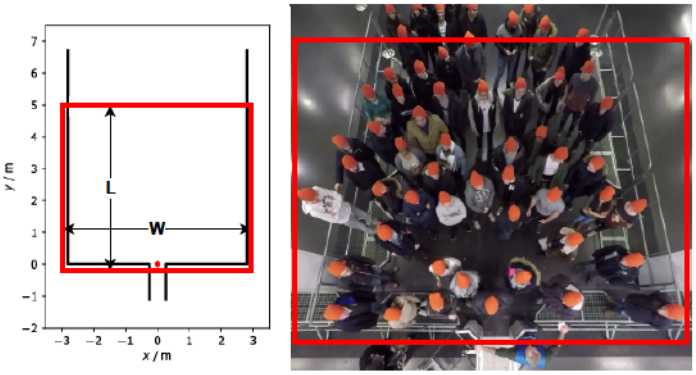

Figure 5: ROI in the entrance. (**Left**) experimental setup with the red dot indicating the coordinate origin [42], (**right**) overhead view of an exemplary experiment. The original frame in the right image is from [42]. The entrance gate width is 0.5 m. The rectangle indicates the entrance area (ROI). L: length of ROI in m. According to the experiment, the width of the ROI (w) varies from 1.2 to 5.6 m.

Table 1: Characteristics of the selected experiments.

Experiment *	Width (m)	Pedestrians	Direction	Frames **
110	1.2	63	Left to right	1285
150	5.6	57	Left to right	1408
170	1.2	25	Left to right	552
270	3.4	67	Right to left	1430
280	3.4	67	Right to left	1640

* The same names as reported in [42]; ** The number of frames that contain pedestrians in the ROI.

Experts performing the manual rating are social psychologists who developed the corresponding rating system [3]. PeTrack [7] was used to track each pedestrian one-by-one, over every frame in the video experiments. Pedestrian ratings are annotated for the first frame when the respective participant becomes visible in the video. The first rating can be extended to the whole video and every frame if that pedestrian does not change his/her behavior. If there is a behavioral change during the experiment, then the rating is also changed. Likewise, it can be extended to the rest of the frames if there is no additional change in the behavior. The rating process is finished after every frame is filled with ratings for every pedestrian. The behaviors of pedestrians are labeled with numbers $\in \{0, 1, 2\}$; 0 indicates that a corresponding pedestrian does not appear in the clip, while 1 and 2 represent non-pushing and pushing behaviors, respectively. Two ground truth

files ($MIM^{Q_{12}}$ and $MIM^{Q_{25}}$) for each experiment were produced for this paper. Further information about the manual rating can be found in [3].

4.1.2 MIM Labeling and Dataset Creation

Three steps are required to create the labeled MIM-based datasets. In the first step, we generated the samples from the videos; the samples were: RAFT-$MIM^{Q_{12}}$, RAFT-$MIM^{Q_{25}}$, Farnebäck-$MIM^{Q_{12}}$, and Farnebäck-$MIM^{Q_{25}}$ sequences. The MIM represents the crowd motion in the ROI, which is presented by the rectangle in Figure 5. It is worth mentioning that the directions of the crowd flows in the videos are not similar. This difference could influence building an efficient classifier because changing the direction is one candidate feature for pushing behavior representation. To address this problem, we unified the direction in all videos from left to right before extracting the samples. Additionally, to improve the efficiency of the datasets, we discarded roughly the first seconds from each video to guarantee that all pedestrians started to move forward.

Based on the ground truth files, the second step labels MIMs in the four MIM sequences into pushing and non-pushing. Each MIM that contains at least one pushing pedestrian is classified as pushing; otherwise, it is labeled as non-pushing.

Finally, we randomly split each dataset into three distinct sets: 70% for training, 15% for validation, and 15% for testing. The 70%-15%-15% split ratio is one of the most common ratios in the deep learning field [45]. The information about the number of pushing and non-pushing samples in the training, validation and test sets for the four MIM-based datasets is given in Table 2. As can be seen from Table 2, our MIM-based datasets suffer from two main limitations: lack of data and a class imbalance problem, since less than 20% of samples are non-pushing.

Table 2: Number of labeled samples in training, validation, and test sets for each MIM-based dataset.

Dataset		Experiment												
		110		150		170		270		280		All		
		P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	Total
RAFT-$MIM^{Q_{12}}$	Training	66	16	76	14	28	5	61	29	86	11	317	75	392
	Validation	13	3	15	3	5	1	13	6	18	2	64	15	79
	Test	13	3	15	3	5	1	13	6	18	2	64	15	79
	Total	92	22	106	20	38	7	87	41	122	15	445	105	550
RAFT-$MIM^{Q_{25}}$	Training	30	6	35	6	13	1	29	13	40	4	147	30	177
	Validation	6	2	7	1	3	1	6	2	8	1	30	7	37
	Test	6	2	7	1	3	1	6	2	8	1	30	7	37
	Total	42	10	49	8	19	3	41	17	56	6	207	44	251
Farnebäck-$MIM^{Q_{12}}$	It has the same samples as the $RAFT^{Q_{12}}$ sets while they are generated using Farnebäck.													
Farnebäck-$MIM^{Q_{25}}$	It has the same samples as the $RAFT^{Q_{25}}$ sets while they are generated using Farnebäck.													

P: pushing samples. NP: non-pushing samples. All: all experiments. 110, 150, 170, 270, and 280: names of the video experiments.

4.2 The Proposed Patch-based Approach

In this section, we propose a new patch-based approach to alleviate the limitations of the MIM-based datasets. The general idea behind our approach is to enlarge the small pushing behavior dataset by dividing each MIM into several patches. After that, we label each patch into "pushing" or "non-pushing" to create a patch-based MIM dataset. The patch should cover a region that can contain a group

of pedestrians, where the motion information of the group is essential for pushing behavior representation. Section 5.2 investigates the impact of the patch area on the classifier accuracy. To further clarify the idea of the proposed approach, we take an example of a dataset with one pushing MIM and one non-pushing MIM, as depicted in Figure 6. After applying our idea with 2×3 patches on the dataset, we obtain a patch-based MIM dataset with four pushing, six non-pushing, and two empty MIM patches. The empty patches are discarded. In conclusion, the dataset is enlarged from two images into ten images. The methodology of our approach, as shown in Figure 7 and Algorithm 2, consists of four main phases: automatic patches labeling, visualization, manual revision, and patch-based MIM dataset creation. The following paragraphs discuss the inputs and the workflow of the approach.

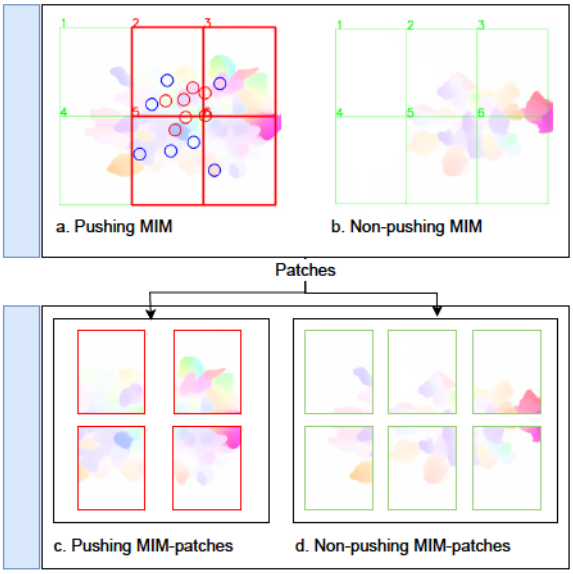

Figure 6: A simple example of the patch-based approach idea. Circles: ground truth pushing. Red boxes: pushing patches. Green boxes: non-pushing patches.

Our approach relies on four inputs (Algorithm 2 and Figure 7, inputs part): (1) MIM-based dataset, which contains a collection of MIMs with the first frame of each MIM; the frames are used in the visualization phase; (2) ROI, n and m, parameters that aim to identify the regions for patches; (3) Pedestrian trajectory data to find the pedestrians in each patch; (4) Manual rating information (ground truth file) helps to label the patches.

Algorithm 2 Patch-based approach

Inputs:
dataset ← collection of MIMs with the first frame of each MIM
ROI ← $matrix[left\_top : [x\_coordinate, y\_coordinate], right\_bottom : [x\_coordinate, y\_coordinate]]$
n, m ← the numbers of rows and columns that are used to divide ROI into $n \times m$ regions.
trajectory ← CSV file, each row represents $\langle order\ of\ \ frame(f_t), pedestrian\ no., pixel\ x - coordinate, pixel\ y - coordinate \rangle$
ground_truth ← CSV file, each row represents $\langle order of\ c_t\ or\ MIM, behavior\ of\ pedestrian\ 1, behavior\ of\ pedestrian\ 2, \ldots, behavior\ of\ last\ pedestrian \rangle$

Outputs:
pushing_folder, non-pushing_folder

1: $region \leftarrow matrix[[]]$ ▷ Automatic patches labeling
2: $patch\_width \leftarrow (ROI[1,0] - ROI[0,0])/m$
3: $patch\_height \leftarrow (ROI[1,1] - ROI[0,1])/n$
4: **for** $i \leftarrow 0, 1, \ldots, n-1$ **do**
5: **for** $j \leftarrow 0, 1, \ldots, m-1$ **do**
6: $region.append([ROI[0,0] + j \times patch\_width, ROI[0,1] + i \times patch\_height, ROI[0,0] + (j+1) \times patch\_width, ROI[0,1] + (i+1) \times patch\_height])$
7: **end for**
8: **end for**
9: $patch\_width \leftarrow (ROI[1,0] - ROI[0,0])/(m-1)$
10: $patch\_height \leftarrow (ROI[1,1] - ROI[0,1])/(n-1)$
11: **for** $i \leftarrow 0, 1, \ldots, n-2$ **do**
12: **for** $j \leftarrow 0, 1, \ldots, m-2$ **do**
13: $region.append([ROI[0,0] + j \times patch\_width, ROI[0,1] + i \times patch\_height, ROI[0,0] + (j+1) \times patch\_width, ROI[0,1] + (i+1) \times patch\_height])$
14: **end for**
15: **end for**
16: $file \leftarrow CSV\ file$
17: **for each** $MIM \in dataset$ **do**
18: $frame\_order \leftarrow MIM\ name$
19: $ped \leftarrow Filter(trajectory.frame\_order)[1]$
20: $patch\_no \leftarrow 1$
21: **for each** $patch\_region \in region$ **do**
22: $behavior \leftarrow 1$ //non-pushing
23: **for each** $ped \in patch\_region$ **do**
24: **if** $Filter(ground\_truth.frame\_order\ \&\ ped) == 2$ **then**
25: $behavior \leftarrow 2$ //pushing
26: break
27: **end if**
28: **end for**
29: $record \leftarrow [patch\_no, frame\_order, behavior]$
30: $file.write(record)$
31: $patch\_no \leftarrow patch\_no + 1$
32: **end for**
33: **end for**

▷ Visualization

34: **for each** $frame \in dataset$ **do**
35: $frame\_order \leftarrow frame\ name$
36: $ped \leftarrow Filter(trajectory.frame\_order)[1]$
37: **for each** $person \in ped$ **do**
38: $behavior \leftarrow Filter(ground\_truth.frame\_order\ \&\ person)$
39: **if** behavior ==2 **then**
40: draw a circle around the position $\langle person[2], person[3] \rangle$ of pedestrian $person[1]$ over $frame$
41: **end if**
42: **end for**
43: **for** $patch\_no \leftarrow 1, 2, \ldots, len(region)$ **do**
44: **if** $Filter(file.frame\_order\ \&\ patch\_no)[2] == 2$ **then**
45: draw a red rectangle around $region[patch\_no - 1]$ over $frame$
46: **else**
47: draw a green rectangle around $region[patch\_no - 1]$ over $frame$
48: **end if**
49: **end for**
50: **end for**

▷ Manual revision

51: **for each** $frame \in dataset$ **do**
52: **for each** $patch\_region \in region$ **do**
53: manual revision of $patch\_region$ in $frame$
54: **if** $patch\_region\ contains\ only\ a\ \ part\ of\ \ one\ pushing\ behavior\ and\ its\ label\ is\ 2$ **then**
55: manually updating the label of the patch_region in $file$ to 6, where 6 means unknown patch
56: **end if**
57: **end for**
58: **end for**

▷ Patch-based MIM dataset creation

59: **for each** $MIM \in dataset$ **do**
60: $MIM\_order \leftarrow MIM\ name$
61: **for** $patch\_no \leftarrow 1, 2, \ldots, len(region)$ **do**
62: $patch \leftarrow MIM[region[patch\_no - 1, 1] : region[patch\_no - 1, 3], [region[patch\_no - 1, 0] : region[patch\_no - 1, 2]]$
63: **if** $Filter(file.MIM\_order\ \&\ patch\_no)[2] == 2$ **then**
64: save $patch$ to pushing_folder under name "$MIM\_order - patch\_no$"
65: **else if** $Filter(file.MIM\_order\ \&\ patch\_no)[2] == 1$ **then**
66: save $patch$ to non-pushing_folder under name "$MIM\_order - patch\_no$"
67: **end if**
68: **end for**
69: **end for**

The first phase, automatic patch labeling, identifies and labels the patches in each MIM (Algorithm 2, lines 1–33 and Figure 7, first phase). The phase contains two steps: (1) Finding the regions of the patches. For this purpose, we find the

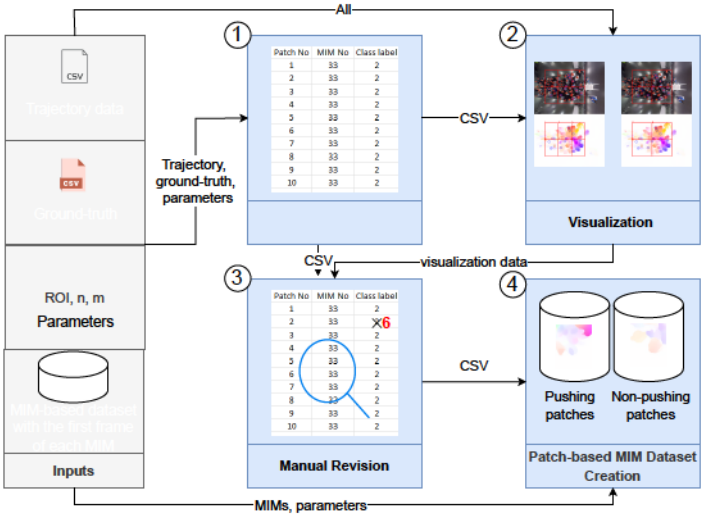

Figure 7: The flow diagram of the proposed patch-based approach. n and m: the numbers of rows and columns, respectively, that are used to divide ROI into n × m regions.

coordinates of the regions that are generated from dividing the ROI area into $n \times m$ parts. The extracted regions can be described as $\{a_k | \, k = 1, 2, \ldots, n \times m\}$, where a_k represents a patch sequence $\{p_{k,i} \in \mathbb{R}^{(w/m) \times (h/n) \times 3} \, | \, i = 1, 2, \ldots, |MIM^Q|\}$, w and h are the ROI width and height, respectively, see Algorithm 2, lines 1–15. We should point out that identifying the regions is performed on at least two levels; to avoid losing any useful information. For example, in Figure 8, we first split ROI by 3 × 3 regions (Algorithm 2, lines 2–8), while in the second level, we reduce the number of regions (2 × 2) to obtain larger patches (Algorithm 2, lines 9–15) containing the missing pushing behaviors (pushing behaviors are divided between the patches) in the first level; (2) Labeling the patches is executed according to the pedestrians' behavior in each patch $p_{k,i}$. Firstly, we find all pedestrians who appear in MIM_i (Algorithm 2, lines 18 and 19). Then, we label each $p_{k,i}$ as pushing if it contains at least one pushing behavior; otherwise, it is labeled as non-pushing (Algorithm 2, lines 20–28). Finally, we store k, i, and the label of $p_{k,i}$ in a CSV-file (Algorithm 2, lines 29 and 30).

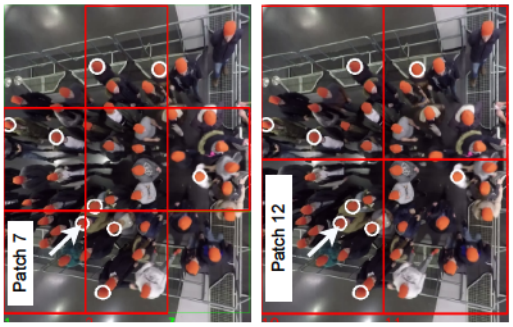

Figure 8: An example of identifying patches and the visualization process. The original frames are from [42]. Red boxes: pushing patches. Green boxes: non-pushing patches. White circles ground truth pushing.

Despite the availability of the pedestrian trajectories, the automatic patch labeling phase is not 100% accurate, affecting the quality of the dataset. The automatic way fails to label some of the patches that only contain a part of one pushing behavior. Therefore, manual revision is required to improve the dataset quality. To ease this process and make it more accurate, the visualization phase (Algorithm 2, lines 34–50 and Figure 7, second phase) visualizes the ground truth pushing (Algorithm 2, lines 36–42), and the label of each $p_{k,i}$ (Algorithm 2, lines 43–49) on the first frame of MIM_i. Figure 8 is an example of the visualization process.

The manual revision phase ensures that each $p_{k,i}$ takes the correct label by manually revising the visualization data (Algorithm 2, lines 51–58 and Figure 7, third phase). The criteria used in the revision are as follows: if $p_{k,i}$ only has a part of one pushing behavior, we change the labels to unknown labels in the CSV-file generated by the first phase; otherwise, the label of $p_{k,i}$ is not changed. The unknown patches do not offer complete information about pushing behavior or non-pushing behavior. Therefore, the final phase in our approach will discard them. A good example of an unknown patch is patch 7, Figure 8a. This patch contains a part of one pushing behavior, as highlighted by the arrow. On the other hand, patch 12 in the aforementioned example (b) contains the whole pushing behavior that we lose in discarding patch 7.

In the final phase (Algorithm 2, lines 59–69 and Figure 7, fourth phase), the patch-based MIM dataset creation is responsible for creating the labeled patch-based MIM dataset, containing two groups of MIM patches, pushing and non-pushing. Firstly, we crop $p_{k,i}$ from MIM_i (Algorithm 2, line 62). Next, and according to the labels of the patches, the pushing patches are stored in the first group (Algorithm 2, lines 63 and 64), while the second group archives the non-pushing patches (Algorithm 2, lines 65 and 66).

4.3 Patch-based MIM Dataset Creation

In this section, we aimed to create several patch-based MIM datasets using the proposed patch-based approach and the MIM-based datasets. The main purposes of the created datasets are: (1) to build and evaluate our classifier; (2) examine the influence of the patch area and clip size on classifier accuracy.

In order to study the impact of the patch area on classifier accuracy, we used two different areas. As we mentioned before, the regions covered by the patches should be enough to house a group of pedestrians. Therefore, according to the ROIs of the experiments, we selected the two patch areas as follows: 1 m × (1 to 1.2) m and 1.67 m × (1.2 to 1.86) m. The dimensions of each area refer to the length x width of patches. Due to the width difference between the experiment setups, there is a variation in the width between the experiments. Table 1 shows the width of each experiment's setup, while the length of the ROI area in all experiment setups was 5 m (Figure 5, left part). For the sake of discussion, we name the 1 m × (1 to 1.2) m patch area as the small patch, and 1.67 m × (1.2 to 1.86) m as the medium patch. Moreover, the small and medium patching with the used levels are illustrated in Figure 9.

The patch-based approach is performed on the RAFT-MIM-based training sets to generate patch-based RAFT-MIM training sets, while it creates patch-based RAFT-MIM validation sets from the RAFT-MIM-based validation sets. The created patch-based RAFT-MIM datasets with their numbers of labeled samples are

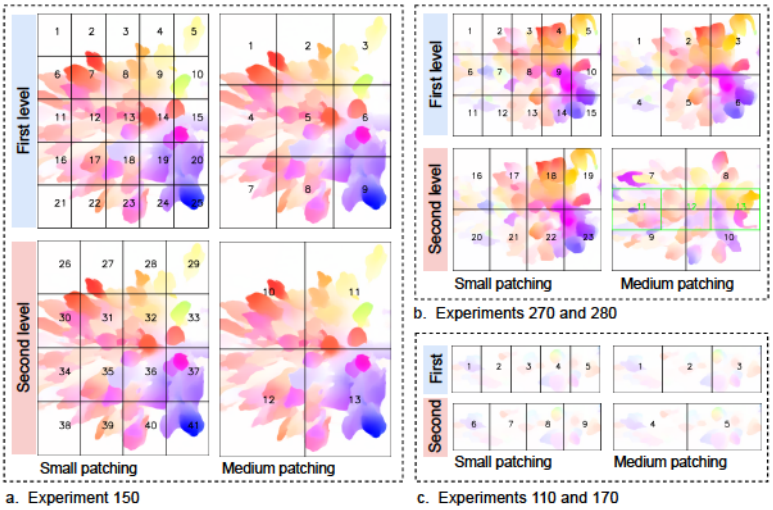

Figure 9: The visualization of patching for the experiments. Numbers represent the patch order in each experiment and level.

presented in Table 3. The table and Figure 10 demonstrate that the proposed approach enlarges the RAFT-MIM-based training and validation sets in both small and medium patching. The approach roughly duplicates the MIM-based training and validation sets 13 times in small patching. While in medium patching, each MIM-based training and validation set is duplicated 8 times. Moreover, our approach decreases the class imbalance issue significantly.

Table 3: Number of labeled MIM patches in training and validation sets for each patch-based MIM dataset.

Dataset		Experiment												
		110		150		170		270		280		All		
		P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	Total
Patch-based small RAFT-MIM$^{Q_{12}}$	Training	350	279	523	932	121	97	528	784	634	806	2156	2898	5054
	Validation	67	53	89	161	20	21	91	169	108	162	375	566	941
	Total	417	332	612	1093	141	118	619	953	742	968	2531	3464	5995
Patch-based small RAFT-MIM$^{Q_{25}}$	Training	156	124	249	419	53	42	236	379	324	354	1018	1318	2336
	Validation	33	26	35	82	9	12	56	53	67	89	200	262	462
	Total	189	150	284	501	62	54	292	432	391	443	1218	1580	2798
Patch-based medium RAFT-MIM$^{Q_{12}}$	Training	237	131	298	354	95	38	540	439	698	326	1868	1288	3156
	Validation	45	26	55	64	16	8	98	105	126	81	340	284	624
	Total	282	157	353	418	111	46	638	544	824	407	2208	1572	3780
Patch-based medium RAFT-MIM$^{Q_{25}}$	Training	107	58	142	151	42	14	242	219	338	146	871	585	1459
	Validation	22	14	20	37	8	6	56	27	68	32	174	116	290
	Total	129	72	162	188	50	20	298	246	406	178	1045	704	1749

P: pushing samples. NP: non-pushing samples. All: all experiments. 110, 150, 170, 270 and 280: names of the video experiments.

The approach reduces the difference percentage between the pushing and non-pushing classes in the patch-based MIM training and validation sets as follows: patch-based small RAFT-MIM$^{Q_{12}}$, from 62% to 16%. Patch-based medium RAFT-MIM$^{Q_{12}}$, from 62% to 17%. Patch-based small RAFT-MIM$^{Q_{25}}$, from 65% to 13%. Patch-based medium RAFT-MIM$^{Q_{25}}$, from 65% to 20%.

Despite these promising results, we can only assess the efficiency of our approach when the CNN-based classifier is trained and tested on our patch-based RAFT-MIM datasets. For this important process, we generate four patch-based

RAFT-MIM test sets. The patch-based approach applies the first level of patching on RAFT-MIM-based test sets (Table 2) to generate the patch-based RAFT-MIM test sets. We apply the first level in the small and medium patching (because we need to evaluate our classifier for detecting pushing behavior at the small and medium patches). Table 4 shows the number of labeled MIM patches in the patch-based RAFT-MIM test sets and their experiments. In Section 5.3, we discuss the impact of the patch-based approach on the accuracy of CNN-based classifiers.

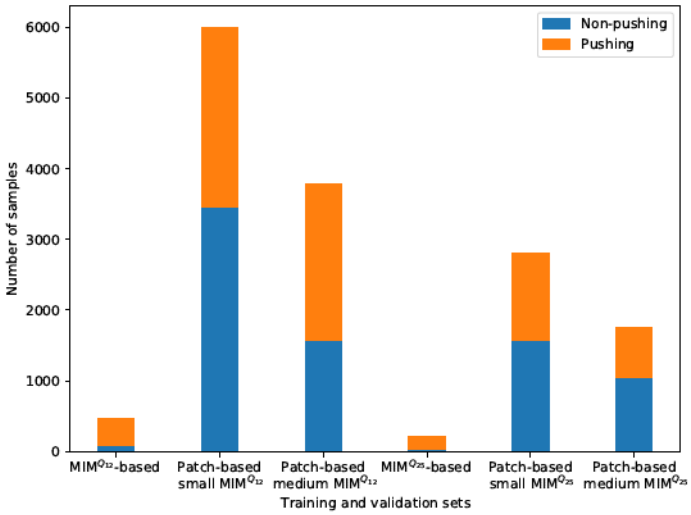

Figure 10: The visualization of the number of pushing and non-pushing samples for the training and validation sets.

Table 4: Number of labeled MIM patches in patch-based test sets.

Test Set	Experiment												
	110		150		170		270		280		All		
	P	NP	P	NP	P	NP	P	NP	P	NP	P	NP	Total
Patch-based small RAFT-MIM$^{Q_{12}}$ test	40	28	47	99	9	13	59	112	61	108	216	360	576
Patch-based small RAFT-MIM$^{Q_{25}}$ test	18	15	19	44	7	8	28	54	25	36	97	157	254
Patch-based medium RAFT-MIM$^{Q_{12}}$ test	26	16	25	47	8	6	47	41	50	40	156	150	306
Patch-based medium RAFT-MIM$^{Q_{25}}$ test	13	8	8	26	5	5	22	19	20	18	68	76	144

P: pushing samples. NP: non-pushing samples. All: all experiments. 110, 150, 170, 270, and 280: names of the video experiments.

5 Experimental Results

This section presents the parameter setup and performance metrics used in the evaluation. Then, it trains and evaluates our classifier and studies the impact of the patch area and clip size on the classifier performance. After that, we investigate the influence of the patch-based approach on the classifier performance. Next, the effect of RAFT on the classifier is discussed. Finally, we evaluate the performance of the proposed framework on the distorted videos.

5.1 Parameter Setup and Performance Metrics

For the training process, the RMSProp optimizer with a binary cross-entropy loss function was used. The batch size and epochs were set to 128 and 100, respectively.

Moreover, when the validation accuracy did not increase for 20 epochs, the training process was automatically terminated. In the RAFT and Farnebäck methods, we used the default parameters.

The implementations in this paper were performed on a personal computer running the Ubuntu operating system with an Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz (8 CPUs) 2.3 GHz and 32 GB RAM. The implementation was written in Python using PyTorch, Keras, TensorFlow, and OpenCV libraries.

In order to evaluate the performance of the proposed framework and our classifier, we used accuracy and F1 score metrics. This combination was necessary since we had imbalanced datasets. Further information on the evaluation metrics can be found in [46].

5.2 Our Classifier Training and Evaluation, the Impact of Patch Area and Clip Size

In this section, we have two objectives: (1) training and evaluating the adapted EfficientNet-B0-based classifier. (2) Investigating the impact of the clip size and patch area on the performance of the classifier.

We compare the adapted EfficientNet-B0-based classifier with three well-known CNN-based classifiers (MobileNet [47], InceptionV3 [48], and ResNet50 [49]) to achieve the above objectives. The classification part in the well-known CNN architectures is modified to be binary. The four classifiers train from scratch on the patch-based RAFT-MIM training and validation sets. Then we evaluate the trained classifiers on patch-based RAFT-MIM test sets to explore their performance.

From the results in Table 5 and Figure 11, it is seen that our trained classifier on the patch-based medium RAFT-MIM$^{Q_{12}}$ dataset achieves better accuracy and F1 scores than other classifiers. More specifically, the EfficientNet-B0-based classifier has 88% accuracy and F1 scores. Furthermore, the medium patches help all classifiers to obtain better performances than small patches. At the same time, MIM$^{Q_{12}}$ is better than MIM$^{Q_{25}}$ for training the four classifiers in terms of accuracy and F1 score.

Table 5: Comparison with well-known CNN-based classifiers on patch-based MIM datasets.

CNN-Based Classifier	Patch-Based MIM Dataset							
	Medium RAFT-MIM$^{Q_{12}}$		Small RAFT-MIM$^{Q_{12}}$		Medium RAFT-MIM$^{Q_{25}}$		Small RAFT-MIM$^{Q_{25}}$	
	Acc.%	F1 Score%	Acc.%	F1 Score%	Acc.%	F1 Score%	Acc.%	F1 Score%
MobileNet	87	87	79	78	85	85	77	74
EfficientNet-B0	**88**	**88**	**81**	**80**	**87**	**87**	**78**	**78**
InceptionV3	85	85	76	75	80	80	76	74
ResNet50	80	80	70	70	74	73	71	69

Acc.: accuracy. Bold: best results in each dataset. Gray highlight: Best results among all datasets.

The patch area influences the classifier performance significantly. For example, medium patches improve the EfficientNet-B0-based classifier accuracy and F1 scores by 7% and 8%, respectively, compared to the small patches. On the other hand, the effect of the MIM sequence (clip size) on the classifier performance is lesser than the influence of the patch area. Compared to medium MIM$^{Q_{25}}$, medium MIM$^{Q_{12}}$ enhances the accuracy and F1 score by 1% in the EfficientNet-B0-based classifier.

In summary, the trained adapted EfficientNet-B0-based classifier on the patch-based medium RAFT-MIM$^{Q_{12}}$ dataset achieves the best performance.

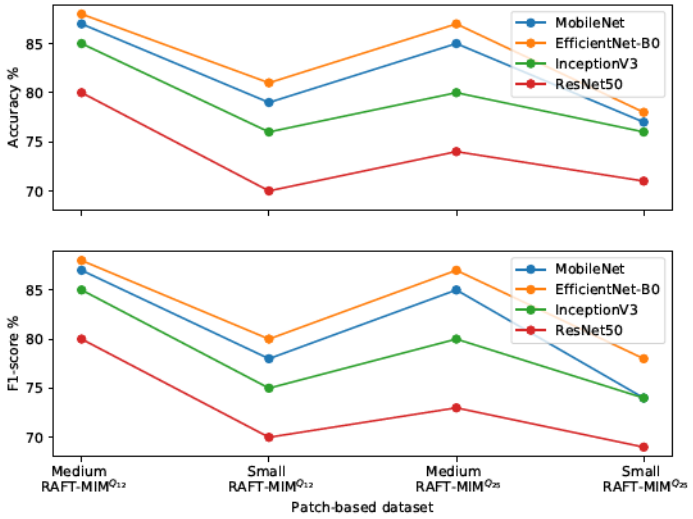

Figure 11: Comparisons of four classifiers over all patch-based RAFT-MIM sets.

5.3 The Impact of the Patch-Based Approach

We evaluated the impact of the proposed patch-based approach on the performance of the trained classifiers on patch-based medium RAFT-MIM$^{Q_{12}}$ training and validation sets. To achieve that, we trained the four classifiers on RAFT-MIM$^{Q_{12}}$-based training and validation sets (Table 2). Then the trained classifiers were evaluated on patch-based medium RAFT-MIM$^{Q_{12}}$ test sets (Table 4).

Table 6 represents the performance of MIM-based classifiers. The comparison between patch-based classifiers and MIM-based classifiers is visualized in Figure 12. We can see that the EfficientNet-B0-based classifier (MIM-based classifier) achieves the best performance, which is a 78% accuracy and F1 score. In comparison, the corresponding patch-based classifier achieves an 88% accuracy and F1 score. This means that the patch-based approach improves the accuracy and F1 score of the EfficientNet-B0-based classifier by 10%. Similarly, in other classifiers, the patch-based approach increases the accuracy and F1 score by at least 15% for each.

Table 6: MIM-based classifier evaluation.

	Patch-Based Classifier		**MIM-Based Classifier**	
CNN-Based Classifier	**Accuracy%**	**F1 Score%**	**Accuracy%**	**F1 Score%**
MobileNet	87	87	71	69
EfficientNet-B0	88	88	78	78
InceptionV3	85	85	51	34
ResNet50	80	80	51	34

5.4 The Impact of RAFT

In order to study the impact of RAFT on our classifier, we trained it using the patch-based medium Farnebäck-MIM$^{Q_{12}}$ dataset. Farnebäck is one of the most popular optical flow methods used in human action detection. Firstly, we created patch-based medium training and validation and test sets from the Farnebäck-MIM$^{Q_{12}}$-based dataset (Table 2). The training and validation sets were used to

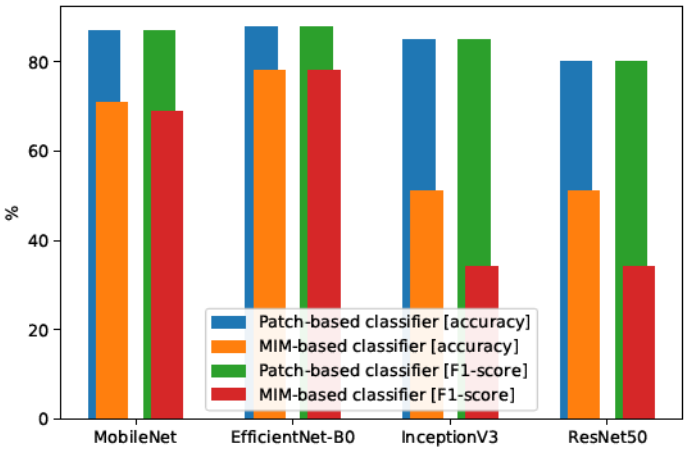

Figure 12: Comparison between MIM-based classifiers and patch-based classifiers.

train the EfficientNet-B0-based classifier (Farnebäck-based classifier), while the test set was used to evaluate the classifier. Finally, we compared the performance of the classifier based on RAFT with the classifier based on Farnebäck. As shown in Table 7 and Figure 13, we find that RAFT improves the classifier performance in all classifiers compared to Farnebäck. In particular, RAFT enhances the EfficientNet-B0-based classifier performance by 8%.

Table 7: Comparison between RAFT-based classifiers and Farnebäck-based classifiers.

	RAFT-Based Classifier		**Farnebäck-Based Classifier**	
Classifier	**Accuracy%**	**F1 Score%**	**Accuracy%**	**F1 Score%**
MobileNet	87	87	81	81
EfficientNet-B0	88	88	80	80
InceptionV3	85	85	79	79
ResNet50	80	80	74	73

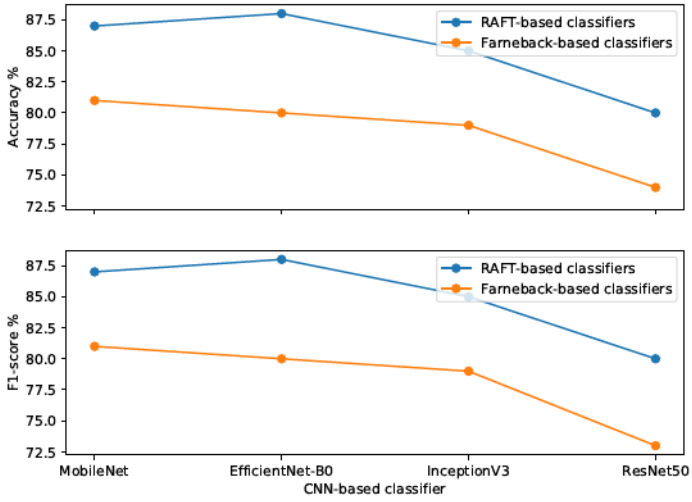

Figure 13: Comparison between the RAFT-based classifier and the Farnebäck-based classifier.

5.5 Comparison between the Proposed Classifier and the Customized CNN-Based Classifiers in Related Works

In this section, we evaluate our classifier by comparing it with two of the most recent customized CNN architectures (CNN-1 [25] and CNN-2 [35]) in the video-based abnormal human behavior detection field. Customized CNNs have simple architectures; CNN-1 used 75 × 75 pixels as an input image, three convolutional layers followed by batch normalization and max pooling operations. Finally, a fully connected layer with a softmax activation function was employed for classification. On the other hand, CNN-2 resized the input images into 28 × 28 pixels, then employed three convolutional layers with three max pooling layers (each max pooling layer with strides of 2 pixels). Moreover, it used two fully connected layers for predictions; the first layer was based on a ReLU activation function, while the second layer used a softmax activation function. For more details on CNN-1 and CNN-2, we refer the reader to [25, 35], respectively.

The three classifiers were trained and evaluated based on the patch-based medium RAFT-MIM$^{Q_{12}}$ dataset. As shown in Table 8 and Figure 14, CNN-1 and CNN-2 obtained low accuracy and F1 scores (less than 61%), while our classifier achieved an 88% accuracy and F1 score.

Table 8: Comparisons to the customized CNN-based classifiers in the related works.

Classifier	Accuracy%	F1 Score%
EfficientNet-B0 (our classifier)	88	88
CNN-1 [25]	60	54
CNN-2 [35]	54	35

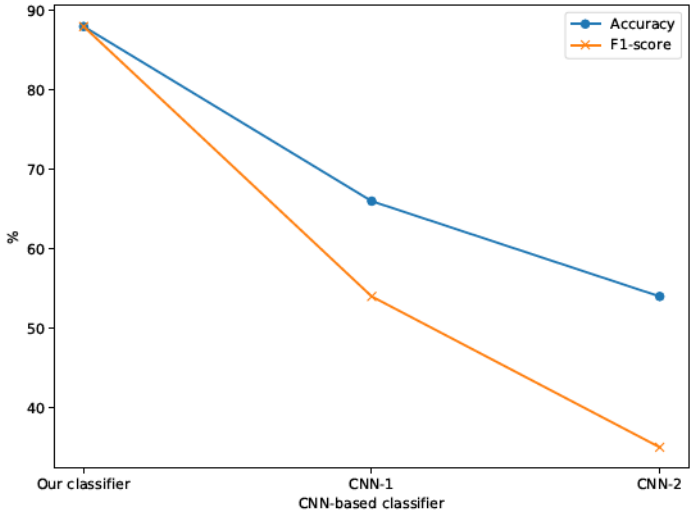

Figure 14: Comparison between our classifier, CNN-1 [25] and CNN-2 [35] based on the patch-based medium RAFT-MIM$^{Q_{12}}$ dataset.

In summary, and according to Figure 15, the reviewed customized CNN architectures are simple and not enough to detect pushing behaviors because the differences between pushing and non-pushing behaviors are not clear in many cases. To address this challenge, we need an efficient classifier (such as the proposed classifier).

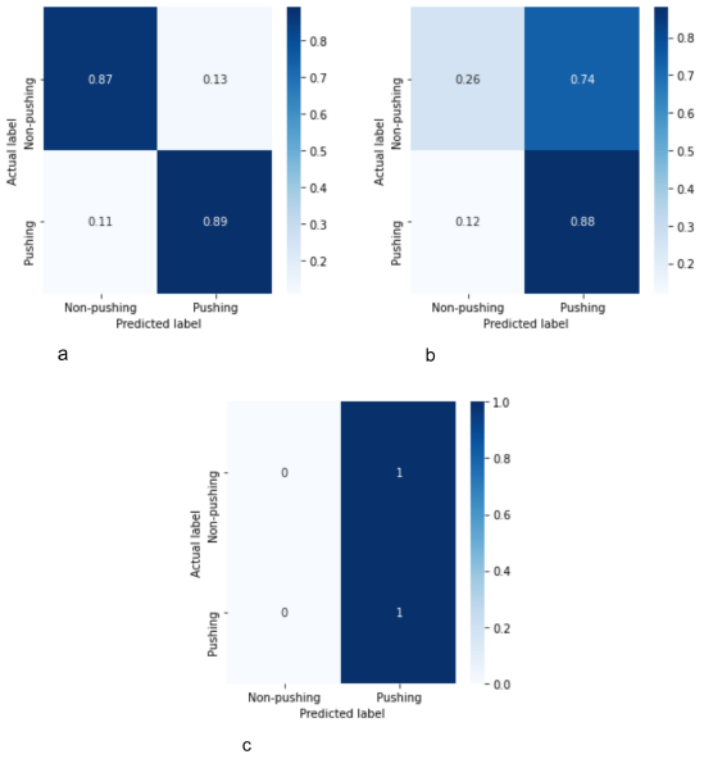

Figure 15: Confusion matrices for our classifier (**a**), CNN-1 [25] (**b**) and CNN-2 [35] (**c**) based on the patch-based medium RAFT-MIM$^{Q_{12}}$ dataset.

5.6 Framework Performance Evaluation

Optical imaging systems often suffer from distortion artifacts [50]. According to [51], distortion is "a deviation from the ideal projection considered in a pinhole camera model, it is a form of optical aberration in which straight lines in the scene do not remain straight in an image". The distortion leads to inaccurate trajectory data [52]. Therefore, PeTrack corrects the distorted videos before extracting the accurate trajectory data, whereas the required information for the correction is not often available. Unfortunately, training our classifier on undistorted videos could decrease the framework performance on distorted videos. Therefore, in this section, we evaluated the proposed framework performance on the distorted videos and studied the impact of the false reduction algorithm on the framework performance. To achieve both goals, firstly, we evaluated the framework's performance without the algorithm on the distorted videos. Then, the framework with the algorithm was evaluated. Finally, we compared both performances.

A qualitative methodology was used in both evaluations; the methodology consisted of four steps: (1) we applied the framework to annotate distorted clips corresponding to MIMs in the RAFT-MIM$^{Q_{12}}$-based test set (Figure 16); the bottom image is an example of an annotated distorted clip; (2) Unfortunately, we could not visualize the ground truth pushing on the distorted frames because the trajectory data were inaccurate. Therefore, we visualized ground truth pushing on the first frame of the corresponding undistorted clips to the distorted clips, Figure 16, top image. Then, we manually identified pushing behaviors on the dis-

torted clips based on the corresponding annotated undistorted clips; This process is highlighted by arrows in Figure 16. (3) We manually calculated the number of true pushing, false pushing, true non-pushing, and false non-pushing. Note that the empty patches were discarded. Non-empty patches containing more than half of the pushing behaviors are labeled as pushing; otherwise, they are labeled as non-pushing. Half of the pushing behavior means that more than half of the visible pedestrian body contributes to pushing; (4) Finally, we measured the accuracy and F1 score metrics.

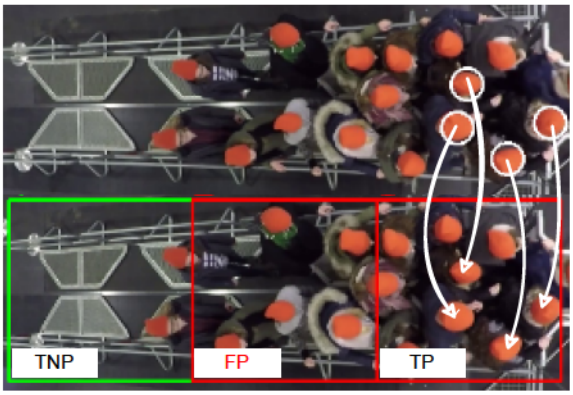

Figure 16: An example of the used qualitative methodology. (**Top**) the first frame of an undistorted clip; (**Bottom**) the first frame of a distorted clip. White arrows: connecting the pushing locations in both undistorted and distorted clips. TP: true pushing. FP: false pushing. TNP: true non-pushing. White circles: ground truth pushing. Red boxes: predicted pushing patches. Green boxes: predicted non-pushing patches.

From Table 9, we can see that our framework with the false reduction algorithm can achieve an 86% accuracy and F1 score on the distorted videos. Moreover, the false reduction improves the performance by 2%.

Table 9: The performance of the framework with and without false reduction on distorted videos.

Framework	Accuracy%	F1 Score%
Without false reduction	84	84
With false reduction	86	86

6 Conclusions, Limitations, and Future Work

This paper proposed a hybrid deep learning and visualization framework for automatic pushing behavior detection at the patch level, particularly from top-view video recordings of crowded event entrances. The framework mainly relied on the power of EfficientNet-B0-based CNN, RAFT, and wheel visualization methods to overcome the high complexity of pushing behavior detection. RAFT and wheel visualization are combined to extract crowd motion information and generate MIM patches. After that, the combination of the EfficientNet-B0-based classifier and false reduction algorithm detects the pushing MIM patches and produces the pushing annotated video. In addition to the proposed framework, we introduced an efficient patch-based approach to increase the number of samples and alleviate the class imbalance issue in pushing datasets. The approach aims to improve the accuracy of the classifier and the proposed framework. Furthermore, we created new

datasets using a real-world ground truth of pushing behavior videos and the proposed patch-based approach for evaluation. The experimental results show that: (1) the patch-based medium RAFT-MIM$^{Q_{12}}$ dataset is the best compared to the other generated datasets for training the CNN-based classifiers; (2) Our classifier outperformed the baseline well-known CNN architectures in image classification as well as customized CNN architectures in the related works; (3) Compared to Farnebäck, RAFT improved the accuracy of the proposed classifier by 8%; (4) The proposed patch-based approach helped to enhance our classifier accuracy from 78% to 88%; (5) Overall, the proposed adapted EfficientNet-B0-based classifier obtained 88% accuracy on the patch-based medium RAFT-MIM$^{Q_{12}}$ dataset; (6) The above results were based on undistorted videos, while the proposed framework obtained 86% accuracy on the distorted videos; (7) The developed false reduction algorithm improved the framework accuracy on distorted videos from 84% to 86%. The main reason behind decreasing the framework accuracy on distorted videos was training the classifier based on undistorted videos.

The main limitations of the proposed framework cannot be applied in real time. Additionally, it does not work well with recorded videos from a moving camera. Moreover, the framework was evaluated only on specific scenarios of crowded event entrances.

In future work, we plan to evaluate our framework in more scenarios of crowded event entrances. Additionally, we plan to optimize the proposed framework to allow real-time detection.

Author Contributions

Conceptualization, A.A., M.M. and M.C.; methodology, A.A., M.M. and M.C.; software, A.A.; validation, A.A.; formal analysis, A.A., M.M. and M.C.; investigation, A.A., M.M. and M.C.; data curation, A.A.; writing—original draft preparation, A.A.; writing—review and editing, A.A., M.M. and M.C.; supervision, M.M. and M.C. All authors have read and agreed to the published version of the manuscript.

Funding

This work was funded by the German Federal Ministry of Education and Research (BMBF: funding number 01DH16027) within the Palestinian-German Science Bridge project framework, and partially by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 491111487.

Institutional Review Board Statement

The experiments were conducted according to the guidelines of the Declaration of Helsinki, and approved by the ethics board at the University of Wuppertal, Germany.

Informed Consent Statement

Informed consent was obtained from all subjects involved in the experiments.

Data Availability Statement

All videos and trajectory data used in generating the datasets were obtained from the data archive hosted by the Forschungszentrum Jülich under CC Attribution 4.0 International license [42]. The undistorted videos, trained CNN-based classifiers, test sets, results, codes (framework; building, training and evaluating the classifiers) generated or used in this paper are publicly available at: `https://github.com/PedestrianDynamics/DL4PuDe` (10 April 2022). The training and validation sets are available from the author upon request.

Acknowledgments

The authors are thankful to Armin Seyfried for the many helpful and constructive discussions. They would also like to thank Anna Sieben, Helena Lügering, and Ezel Üsten for developing the rating system and annotating the pushing behavior in the video experiments. Additionally, the authors would like to thank Maik Boltes and Tobias Schrödter for valuable technical discussions.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] J. Adrian, M. Boltes, A. Sieben, and A. Seyfried, "Influence of corridor width and motivation on pedestrians in front of bottlenecks," in *Traffic and Granular Flow 2019*. Springer, 2020, pp. 3–9.

[2] J. Adrian, A. Seyfried, and A. Sieben, "Crowds in front of bottlenecks at entrances from the perspective of physics and social psychology," *Journal of the Royal Society Interface*, vol. 17, no. 165, p. 20190871, 2020.

[3] H. Lügering, E. Üsten, and A. Sieben, "Pushing and non-pushing forward motion in crowds: A systematic psychological method for rating individual behavior in pedestrian dynamics," *Manuscript submitted for publication*, 2022.

[4] M. Haghani, M. Sarvi, and Z. Shahhoseini, "When 'push'does not come to 'shove': Revisiting 'faster is slower'in collective egress of human crowds," *Transportation research part A: policy and practice*, vol. 122, pp. 51–69, 2019.

[5] A. Sieben, J. Schumann, and A. Seyfried, "Collective phenomena in crowds—where pedestrian dynamics need social psychology," *PLoS one*, vol. 12, no. 6, p. e0177328, 2017.

[6] J. Adrian, M. Boltes, S. Holl, A. Sieben, and A. Seyfried, "Crowding and queuing in entrance scenarios: influence of corridor width in front of bottlenecks," *arXiv preprint arXiv:1810.07424*, 2018.

[7] M. Boltes, A. Seyfried, B. Steffen, and A. Schadschneider, "Automatic extraction of pedestrian trajectories from video recordings," in *Pedestrian and evacuation dynamics 2008*. Springer, 2010, pp. 43–54.

[8] R. Nayak, U. C. Pati, and S. K. Das, "A comprehensive review on deep learning-based methods for video anomaly detection," *Image and Vision Computing*, vol. 106, p. 104078, 2021.

[9] M. J. Roshtkhari and M. D. Levine, "An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions," *Computer vision and image understanding*, vol. 117, no. 10, pp. 1436–1452, 2013.

[10] G. Singh, A. Khosla, and R. Kapoor, "Crowd escape event detection via pooling features of optical flow for intelligent video surveillance systems," *International Journal of Image, Graphics and Signal Processing*, vol. 10, no. 10, p. 40, 2019.

[11] M. George, C. Bijitha, and B. R. Jose, "Crowd panic detection using autoencoder with non-uniform feature extraction," in *2018 8th International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2018, pp. 11–15.

[12] G. L. Santos, P. T. Endo, K. H. d. C. Monteiro, E. d. S. Rocha, I. Silva, and T. Lynn, "Accelerometer-based human fall detection using convolutional neural networks," *Sensors*, vol. 19, no. 7, p. 1644, 2019.

[13] A. Mehmood, "Lightanomalynet: A lightweight framework for efficient abnormal behavior detection," *Sensors*, vol. 21, no. 24, p. 8501, 2021.

[14] X. Zhang, Q. Zhang, S. Hu, C. Guo, and H. Yu, "Energy level-based abnormal crowd behavior detection," *Sensors*, vol. 18, no. 2, p. 423, 2018.

[15] J. F. Kooij, M. C. Liem, J. D. Krijnders, T. C. Andringa, and D. M. Gavrila, "Multi-modal human aggression detection," *Computer Vision and Image Understanding*, vol. 144, pp. 106–120, 2016.

[16] H. Gan, C. Xu, W. Hou, J. Guo, K. Liu, and Y. Xue, "Spatiotemporal graph convolutional network for automated detection and analysis of social behaviours among pre-weaning piglets," *Biosystems Engineering*, vol. 217, pp. 102–114, 2022.

[17] H. Gan, M. Ou, E. Huang, C. Xu, S. Li, J. Li, K. Liu, and Y. Xue, "Automated detection and analysis of social behaviors among preweaning piglets using key point-based spatial and temporal features," *Computers and Electronics in Agriculture*, vol. 188, p. 106357, 2021.

[18] T.-H. Vu, J. Boonaert, S. Ambellouis, and A. Taleb-Ahmed, "Multi-channel generative framework and supervised learning for anomaly detection in surveillance videos," *Sensors*, vol. 21, no. 9, p. 3179, 2021.

[19] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[20] L. Li, S. Zhang, and B. Wang, "Apple leaf disease identification with a small and imbalanced dataset based on lightweight convolutional networks," *Sensors*, vol. 22, no. 1, p. 173, 2021.

[21] P. Wang, E. Fan, and P. Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021.

[22] E. Duman and O. A. Erdem, "Anomaly detection in videos using optical flow and convolutional autoencoder," *IEEE Access*, vol. 7, pp. 183 914–183 923, 2019.

[23] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[24] Z. Ilyas, Z. Aziz, T. Qasim, N. Bhatti, and M. F. Hayat, "A hybrid deep network based approach for crowd anomaly detection," *Multimedia Tools and Applications*, pp. 1–15, 2021.

[25] C. Direkoglu, "Abnormal crowd behavior detection using motion information images and convolutional neural networks," *IEEE Access*, vol. 8, pp. 80 408–80 416, 2020.

[26] A. A. Almazroey and S. K. Jarraya, "Abnormal events and behavior detection in crowd scenes based on deep learning and neighborhood component analysis feature selection," in *Joint European-US Workshop on Applications of Invariance in Computer Vision*. Springer, 2020, pp. 258–267.

[27] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision*. Springer, 2020, pp. 402–419.

[28] D. F. Tom Runia, "Optical flow visualization," 2020. [Online]. Available: https://github.com/tomrunia/OpticalFlow_Visualization

[29] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International journal of computer vision*, vol. 92, no. 1, pp. 1–31, 2011.

[30] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[32] S. Coşar, G. Donatiello, V. Bogorny, C. Garate, L. O. Alvares, and F. Brémond, "Toward abnormal trajectory and event detection in video surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 3, pp. 683–695, 2016.

[33] J. Jiang, X. Wang, M. Gao, J. Pan, C. Zhao, and J. Wang, "Abnormal behavior detection using streak flow acceleration," *Applied Intelligence*, pp. 1–18, 2022.

[34] M. Xu, X. Yu, D. Chen, C. Wu, and Y. Jiang, "An efficient anomaly detection system for crowded scenes using variational autoencoders," *Applied Sciences*, vol. 9, no. 16, p. 3337, 2019.

[35] N. C. Tay, T. Connie, T. S. Ong, K. O. M. Goh, and P. S. Teh, "A robust abnormal behavior detection method using convolutional neural network," in *Computational Science and Technology*. Springer, 2019, pp. 37–47.

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

[37] S. Smeureanu, R. T. Ionescu, M. Popescu, and B. Alexe, "Deep appearance features for abnormal behavior detection in video," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 779–789.

[38] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.

[39] Y. Hu, "Design and implementation of abnormal behavior detection based on deep intelligent analysis algorithms in massive video surveillance," *Journal of Grid Computing*, vol. 18, no. 2, pp. 227–237, 2020.

[40] S. Zhou, W. Shen, D. Zeng, M. Fang, Y. Wei, and Z. Zhang, "Spatial–temporal convolutional neural networks for anomaly detection and localization in crowded scenes," *Signal Processing: Image Communication*, vol. 47, pp. 358–368, 2016.

[41] C. Zhang, Y. Xu, Z. Xu, J. Huang, and J. Lu, "Hybrid handcrafted and learned feature framework for human action recognition," *Applied Intelligence*, pp. 1–17, 2022.

[42] "Crowds in front of bottlenecks from the perspective of physics and social psychology," http://doi.org/10.34735/ped.2018.1, Pedestrian Dynamics Data Archive hosted by the Forschungszentrum Jülich, 2018.

[43] G. Hollows and N. James, "Understanding focal length and field of view," *Retrieved October*, vol. 11, p. 2018, 2016.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[45] B. Genc and H. Tunc, "Optimal training and test sets design for machine learning," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 27, no. 2, pp. 1534–1545, 2019.

[46] S. A. A. Ismael, A. Mohammed, and H. Hefny, "An enhanced deep learning approach for brain cancer mri images classification using residual networks," *Artificial intelligence in medicine*, vol. 102, p. 101779, 2020.

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[50] S. Van der Jeught, J. A. Buytaert, and J. J. Dirckx, "Real-time geometric lens distortion correction using a graphics processing unit," *Optical Engineering*, vol. 51, no. 2, p. 027002, 2012.

[51] O. Stankiewicz, G. Lafruit, and M. Domański, "Multiview video: Acquisition, processing, compression, and virtual view rendering," in *Academic Press Library in Signal Processing, Volume 6.* Elsevier, 2018, pp. 3–74.

[52] L. H. Vieira, E. A. Pagnoca, F. Milioni, R. A. Barbieri, R. P. Menezes, L. Alvarez, L. G. Déniz, D. Santana-Cedrés, and P. R. Santiago, "Tracking futsal players with a wide-angle lens camera: accuracy analysis of the radial distortion correction based on an improved hough transform algorithm," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 5, no. 3, pp. 221–231, 2017.

Publication II

A Cloud-based Deep Learning Framework for Early Detection of Pushing at Crowded Event Entrances

This article has been published as Alia, Ahmed, Mohammed Maree, Mohcine Chraibi, Anas Toma, and Armin Seyfried. "A Cloud-based Deep Learning Framework for Early Detection of Pushing at Crowded Event Entrances." IEEE Access (2023).

Author's Contributions

Conceptualization: **Ahmed Alia**
Methodology: **Ahmed Alia**, Mohammed Maree, Mohcine Charaibi, Armin Seyfried
Software: **Ahmed Alia**
Validation: **Ahmed Alia**
Formal Analysis: **Ahmed Alia**
Visualization: **Ahmed Alia**, Anas Toma
Data Curation: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
writing—review and editing **Ahmed Alia**, Mohammed Maree, Mohcine Charaibi, Anas Toma, Armin Seyfried
GitHub Repository Creation: **Ahmed Alia**

GitHub: `https://github.com/PedestrianDynamics/CloudFast-DL4PuDe`

Publication DOI: `https://doi.org/10.1109/ACCESS.2023.3273770`

A Cloud-based Deep Learning Framework for Early Detection of Pushing at Crowded Event Entrances

Ahmed Alia[1,2,3*], Mohammed Maree[4*], Mohcine Chraibi[1], Anas Toma[5], Armin Seyfried[1,2*]

[1] Institute for Advanced Simulation, Forschungszentrum Jülich, 52425 Jülich, Germany.

[2] Computer Simulation for Fire Protection and Pedestrian Traffic, University of Wuppertal, 42285 Wuppertal, Germany.

[3] Department of Management Information Systems, Faculty of Engineering and Information Technology, An-Najah National University, Nablus, Palestine.

[4] Department of Information Technology, Faculty of Engineering and Information Technology, Arab American University, Jenin, Palestine.

[5] Department of Electrical and Computer Engineering, An-Najah National University, Nablus, Palestine.

Corresponding authors: a.alia@fz-juelich.de, mohammed.maree@aaup.edu, a.seyfried@fz-juelich.de

Abstract Crowding at the entrances of large events may lead to critical and life-threatening situations, particularly when people start pushing each other to reach the event faster. Automatic and timely identification of pushing behavior would help organizers and security forces to intervene early and mitigate dangerous situations. In this paper, we propose a cloud-based deep learning framework for automatic early detection of pushing in crowded event entrances. The proposed framework initially modifies and trains the EfficientNetV2B0 Convolutional Neural Network model. Subsequently, it integrates the adapted model with an accurate and fast pre-trained deep optical flow model with the color wheel method to analyze video streams and identify pushing patches in real-time. Moreover, the framework uses live capturing technology and a cloud-based environment to collect video streams of crowds in real-time and provide early-stage results. A novel dataset is generated based on five real-world experiments and their associated ground truth data to train the adapted EfficientNetV2B0 model. The experimental setups simulated a crowded event entrance, while the ground truths for each video experiment was generated manually by social psychologists. Several experiments on the videos and the generated dataset are carried out to evaluate the accuracy and annotation delay time of the proposed framework. The experimental results show that the proposed framework identified pushing behaviors with an accuracy rate of 87 % within a reasonable delay time.

Keywords: Artificial Intelligence, Computer Vision, Convolutional Neural Network, Deep Learning, Image Classification, Intelligent System, Machine Learning, Pushing Behavior Detection

1 Introduction

The entrances of large-scale events such as sport venues, concerts, and religious gatherings are organized as bottlenecks for access control, ticket validation, or

security check [1]. In these scenarios, some pedestrians might start pushing each other to gain faster access to the event. According to Lügering et al. [2], pushing for forward motion is defined as "a behavior that can involve using arms, shoulders, or elbows; or simply the upper body, in which one person actively applies force to another person (or people) to overtake, while shifting their direction to the side or back, or force them to move forward more quickly." Additionally, using gaps in the crowd is considered as a strategy of pushing because it is a form of overtaking [2]. Indeed, such behavior often increases the crowd's density [1, 3], resulting in the lack of comfort zones and, more importantly, can lead to dangerous situations [4, 5]. In such cases, early pushing detection is essential, as it can provide valuable information to the organizers and the security team for better crowd management, thereby ensuring a smoother flow at entrances with higher safety [6]. Since manual identification of pushing behavior in the early stages can be complex or impossible, developing an automatic detection framework in real-time or near real-time is crucial. However, automatic pushing detection is still a challenging task due to the highly-dense crowds, the diversity of pushing behavior strategies, and the varying features for pushing behavior representation, which still requires further investigation and identification [7].

Surveillance cameras have recently been widely integrated with computer vision techniques to automatically identify abnormal behaviors from crowds [8, 9]. Within the realm of computer vision, pushing behavior can be classified as abnormal behavior. Machine learning algorithms, particularly Convolutional Neural Network (CNN) architectures, have remarkably succeeded in several computer vision tasks; among these is abnormal behavior detection in crowds [10]. One of the critical reasons for this success is that CNN can learn the relevant features [11, 12] and classification automatically from data without human intervention [13, 14]. Although CNN architectures are powerful for modeling human behaviors, building an accurate model requires a large training dataset [15, 16], which is often unavailable. Researchers have developed hybrid-based approaches that integrate CNN with handcrafted feature descriptors to address this limitation [17, 18]. These approaches employ descriptors to obtain useful data, which is subsequently used by CNN to learn and identify abnormal behavior automatically. Due to the limited availability of labeled data for pushing behavior, hybrid-based approaches may be more appropriate for automatically identifying pushing behavior. For example, Alia et al. [7] proposed a hybrid deep learning and visualization framework for pushing behavior detection in video recordings of crowded event entrances. Unfortunately, this framework does not cope with early detection requirements because it is slow and can not work with the live camera stream. To the best of our knowledge, despite the numerous computer vision and machine learning approaches reported in the literature, none of them can detect pushing behavior in real-time or near real-time from crowds.

In order to address the above limitations, this article introduces a novel cloud-based deep learning framework for pushing patch detection in live video streams acquired from crowded event entrances. In this framework, we propose: 1) Integrating a robust deep optical flow model (GPU-based pre-trained Recurrent All-pairs Field Transforms (RAFT) [19]) with the color wheel method [20, 21] to accurately and rapidly extract the visual motion information from the crowd. 2) Adapting and training EfficientNetV2B0-based CNN [22] using visual motion information to detect pushing patches accurately. 3) Using live camera technology and a cloud environment to provide more powerful computational resources and

help to collect and annotate the video stream of the crowd in real-time.

The main contributions of this article are summarized as follows:

1. To the best of our knowledge, we propose the first real-time or near real-time automatic framework dedicated to early identifying pushing behavior in human crowds.
2. We introduce a new video analysis and pushing detection approach based on integrating an adapted version of EfficentNetV2B0, GPU-based pre-trained RAFT model, and color wheel method.
3. We create a novel dataset for pushing behavior, using five real-world experiments with their associated ground truths. This dataset is not only used as a training and evaluation resource for our adapted EfficientNetV2B0, but can also be a valuable asset for future research in this area.
4. We perform a thorough performance comparison of fifteen CNN architectures for pushing detection using the generated dataset.

The rest of the paper is organized as follows. Section 2 reviews the related studies of video-based abnormal human behavior detection. The proposed framework is presented in Section 3. Section 5 discusses the evaluation process and experimental results. Finally, the conclusion and future work are summarized in Section 5.

2 Related Work

Generally, identifying pushing behavior in videos falls under the field of computer vision, specifically in the task of abnormal behavior detection. CNNs have played a crucial role in significant advancements in this area [23]. Consequently, in this section, our objective is to examine several abnormal behavior detection approaches that have been developed using CNNs.

A customized CNN-based method to identify abnormal activities in videos was presented by Tay et al [24]. The authors trained a customized CNN for feature extraction and labeling using normal and abnormal samples. In another study, Alafif et al. [18] proposed two methods of identifying abnormal behaviors in small and large-scale crowd videos. The first method employs a combination of a CNN model and a random forest classifier to detect anomaly behaviors at the object level in a small-scale crowd. In contrast, the second method utilizes two classifiers to recognize abnormal behaviors in a large-scale crowd. The initial model, finds the frames containing abnormal behaviors, while the second classifier, You Only Look Once (version 2), processes those frames to identify abnormal behaviors exhibited by individuals. The effectiveness of these techniques relies heavily on utilizing CNNs to learn features from labeled datasets containing both normal and abnormal behaviors. A large training dataset of normal and abnormal behaviors is necessary to create an accurate and adaptable CNN model. However, obtaining such a dataset is often unattainable for various abnormal behaviors, including pushing behavior.

In order to overcome the shortage of large datasets comprising normal and abnormal behaviors, some researchers have utilized one-class classifiers with datasets consisting only of normal behaviors. It is easier to obtain or create a dataset that contains only normal behavior than a dataset that includes both normal

and abnormal behaviors [25, 26]. The fundamental concept behind the one-class classifier is to exclusively learn from normal behaviors, thereby establishing a class boundary between normal and undefined (abnormal) classes. For example, Sabokrou et al. [25] employed a pre-trained CNN for extracting motion and appearance information from crowded scenes. Subsequently, they utilized a one-class Gaussian distribution to construct the classifier using datasets comprised of normal behavior. Similarly, in [26, 27], the authors developed one-class classifiers by utilizing a dataset of normal samples. In [26], Xu et al. employed a convolutional variational autoencoder to extract features, followed by the use of multiple Gaussian models to detect abnormal behavior. Meanwhile, in [27], a pre-trained CNN model was utilized for feature extraction, with one-class support vector machines being used to identify abnormal behavior. Another study by Ilyas et al. [28] utilized a pre-trained CNN and a gradient sum of the frame difference to extract significant features. Following this, three support vector machines were trained on normal behavior to detect abnormal behaviors. Generally, the one-class classifier is commonly used when the target behavior class or abnormal behavior is infrequent or poorly defined [29]. However, pushing behavior is well-defined and not rare, particularly in high-density and competitive situations. Furthermore, this type of classifier regards new normal behavior as abnormal.

To overcome the limitations of CNN-based and one-class classifier approaches, several studies have combined multi-class CNN with one or more handcrafted feature descriptors [28, 10]. As an example, Duman et al. [17] utilized the traditional Farnebäck optical flow approach in conjunction with CNN to detect anomalous behavior. They extracted direction and speed information using Farnebäck and CNN, and then utilized a convolutional long short-term memory network to construct the classifier. Similarly, Hu et al. [30] employed a combination of the histogram of gradient and CNN for feature extraction, while a least-squares support vector was used for classification. Almazroey et al. [31] focused on utilizing the Lucas-Kanade optical flow method, pre-trained CNN, and feature selection method (neighborhood component analysis) to extract relevant features. They then used a support vector machine to generate a trained classifier. In a different study [32], Zhou et al. introduced a CNN-based method to identify and locate abnormal activities. This approach integrated optical flow with CNN for feature extraction and utilized a CNN for classification. Direkoglu [10] utilized the Lucas-Kanade optical flow method and CNN to extract relevant features and identify "escape and panic behaviors".

Most of the hybrid-based approaches for abnormal behavior detection that were reviewed have limited efficiency in detecting pushing since 1) The descriptors used in these approaches can only extract limited essential data from high-density crowds to represent pushing behavior. 2) Some CNN architectures commonly utilized in these approaches may not be effective in dealing with the increased variations within pushing behavior (intra-class variance) and the substantial resemblance between pushing and non-pushing behaviors (high inter-class similarity), which can potentially result in misclassification. To benefit from the power of hybrid-based approaches on a small dataset, Alia et al. [7] introduced a hybrid framework for pushing patch detection in video recordings of crowds. The authors utilized a robust handcrafted feature descriptor and efficient CNN architecture in this framework. In more details, the framework used a deep optical flow technique to extract the motion information from the crowds. This information is then analyzed using an EfficientNetB0-based CNN and false reduction algorithms

to identify and label pushing patches in the video. However, this framework does not cope with early detection requirements due to three reasons. First, it can only handle offline-recorded videos. Second, The deep optical flow technique employed in motion extraction is slow because it was performed on the CPU. Third, it needs to identify pushing patches for the whole video before producing the output. Moreover, as reported by the authors, the accuracy of the framework decreases with complex scenarios of pushing.

To sum up, the reviewed methods have limitations regarding early pushing detection in crowded human environments. On the one hand, approaches that rely solely on CNNs for feature extraction require a large dataset containing normal and abnormal behaviors, which is typically unavailable for pushing scenarios. On the other hand, one-class classifiers are often used for infrequent or poorly defined target behavior or abnormal behavior. However, pushing behavior is well-defined and common, particularly in high-density and competitive scenarios. Additionally, this type of classifier may misclassify new normal behavior as abnormal. Although hybrid-based approaches may be more suitable for pushing behavior, existing methods do not meet the requirements for early pushing detection in human crowds. To overcome these limitations, this article proposes a novel framework that adapts the EfficientNetV2B0 model and integrates it with GPU-based RAFT, wheel color method and live camera technology on a cloud platform. The following section provides a detailed discussion of the framework.

3 The Proposed Framework

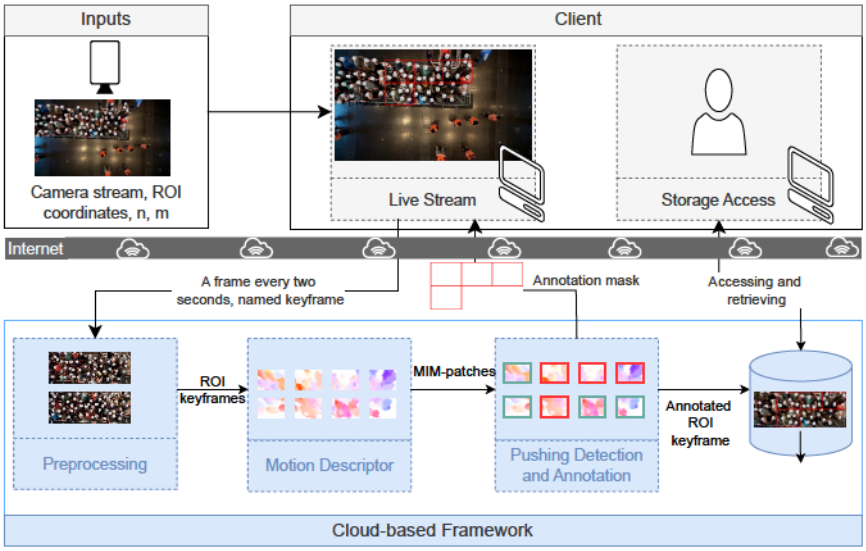

Figure 1: The proposed framework architecture. ROI refers to the entrance area (Region Of Interest). User-defined row (n) and column (m) are used to split Motion Information Map (MIM) into $n \times m$ patches.

In this section, we describe the proposed framework for early detection of pushing within the live camera stream of crowded event entrances, where the camera is fixed and top-view. Fig. 1 shows the architecture of our framework which comprises three major components: preprocessing; motion descriptor; and push-

ing detection and annotation. The first component aims to collect and process the live camera stream, as well as display the stream on the web client in real-time. Simultaneously, the second component, the motion descriptor, employs the GPU-based RAFT model and color wheel method [20, 21] to extract the visual motion information from the crowd. Finally, the pushing detection and annotation component utilizes the adapted and trained EfficientNetV2B0 model to analyze the visual motion information and detect pushing patches. Notably, it directly annotates the regions that contain pushing behavior on the live stream on the web client. The following sections provide a more detailed discussion of the three components.

3.1 Preprocessing

In order to reduce the computational time of the framework without sacrificing performance, the preprocessing component directly displays the client camera stream on the web client. At the same time, it collects only the data required for detection purposes from the live stream. Let $\{f^t\}$ represents the live camera stream, where t is the time of the frame f in the stream. Firstly, this component displays the live stream on the web client in real-time without uploading it to the cloud. Then, a frame f^t is collected from the stream every two seconds, hereafter referred to as keyframe (examples in Fig. 2a). After that, this component utilizes the user-defined coordinates in pixel units to crop the entrance area $\bar{f}^t$ (ROI keyframe) from its corresponding keyframe f^t. Finally, $\bar{f}^t$ is submitted as an input to the second component. For the brevity, we name the ROI keyframe sequence $\{\bar{f}^t, \bar{f}^{t+2}, \bar{f}^{t+4}, \dots\}$ as $\{\bar{f}_i \,|\, i = 1, 2, 3, \dots\}$, where i is the order of the ROI keyframe in the stream, and t is the time in seconds. Fig. 2b displays two examples of $\bar{f}_i$.

3.2 Motion Descriptor

Using this component, we aim to extract the crowd's motion characteristics at the patch level. More specifically, this component estimates the motion direction, magnitude, and associated spatio-temporal information from the crowds, and accordingly visualizes this information. The displayed information includes relevant features that are important for representing the pushing behavior.

As shown in Fig. 3, the component uses GPU-based pre-trained RAFT model and color wheel method to achieve its purpose. Unlike the majority of the already used optical flow methods [34, 35], a GPU-based pre-trained RAFT model performs well in terms of speed, accuracy, and generality for dense crowds [7, 19]. This model was created by training an ensemble of CNN and recurrent neural networks on the Sintel dataset to calculate the optical flow between two images. For further details about the model, we refer the reader to [19]. Firstly, the component uses the pre-trained model to calculate the displacement of each pixel $\langle x, y\rangle$ between each pair of $\bar{f}_i$ and $\bar{f}_{i+1}$, generating the dense displacement field d_i. Each pixel location $\langle x, y\rangle$ in d_i is presented by a vector, given by

$$\langle u_{\langle x,y\rangle}, v_{\langle x,y\rangle}\rangle_{\bar{f}_i,\bar{f}_{i+1}} = RAFT(\langle x, y\rangle_{\bar{f}_i,\bar{f}_{i+1}}), \tag{1}$$

where u and v are horizontal and vertical displacements of a pixel at the $\langle x, y\rangle$ location between $\bar{f}_i$ and $\bar{f}_{i+1}$, respectively. This implies that d_i is a matrix of the

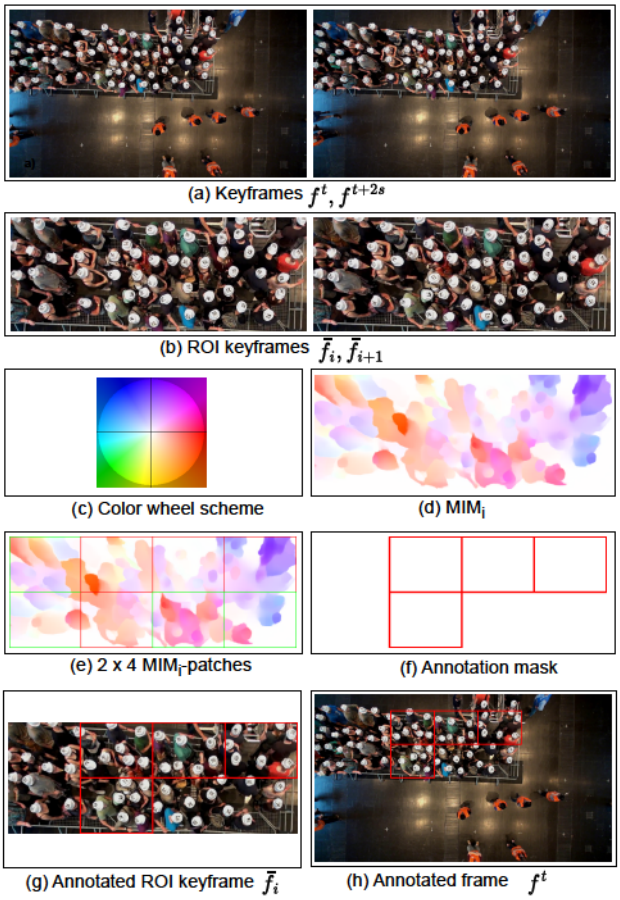

Figure 2: An illustration of two keyframes (experiment Entrance_2 [33]), two ROI keyframes, color wheel schema [21], MIM, 2×4 MIM-patches, annotation mask, annotated ROI keyframe and annotated frame. t is the time of the frame f in the stream. i is the order of the ROI keyframe in the stream. s means second. The red boxes indicate pushing patches, while the green boxes mean non-pushing patches.

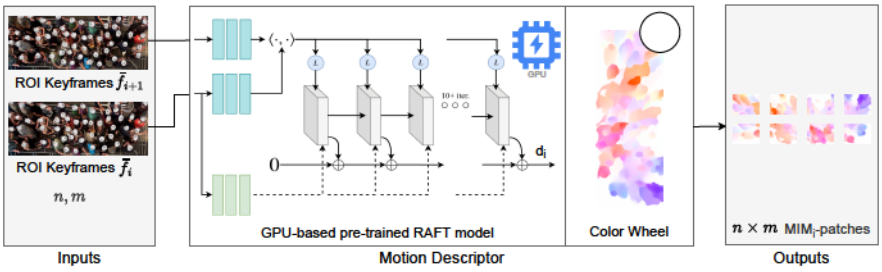

Figure 3: Motion descriptor component pipeline. i is the order of the ROI keyframe in the stream. d refers to a dense displacement field. MIM represents motion information map.

vectors, as described in

$$d_i = \left\{ \langle u_{\langle x,y \rangle}, v_{\langle x,y \rangle} \rangle_{\bar{f}_i, \bar{f}_{i+1}} \right\}_{(x,y)=(1,1)}^{(w,h)}, \tag{2}$$

where w and h are the $\bar{f}_i$ width and height, respectively.

After the estimation of d_i, the descriptor applies the color wheel method to deduce the visual motion information from d_i. It begins by calculating the direction θ and magnitude of each vector $\langle u_{\langle x,y\rangle}, v_{\langle x,y\rangle}\rangle$ in d_i using Eq. (3) and Eq. (4), respectively. The color wheel then visualizes the magnitude and direction information to generate MIM_i from the calculated information, where $\text{MIM}_i \in \mathbb{R}^{w\times h\times 3}$, and 3 is the number of channels in MIM_i. Fig. 2c is the color wheel scheme, and Fig. 2d is an example of MIM_i that is generated from the pair of $\bar{f}_i$ and $\bar{f}_{i+1}$ (Fig. 2b). According to the wheel schema, the color represents the motion direction, while the color intensity denotes the motion magnitude or speed.

$$\theta(\langle x,y\rangle)_{\bar{f}_i,\bar{f}_{i+1}} = \pi^{-1}\arctan(\frac{v_{\langle x,y\rangle}}{u_{\langle x,y\rangle}}). \tag{3}$$

$$mag(\langle x,y\rangle)_{c_i} = \sqrt{u^2_{\langle x,y\rangle} + v^2_{\langle x,y\rangle}} \tag{4}$$

The motion descriptor component divides each MIM_i into $n \times m$ MIM_i-patches to help the framework localizing pushing in ROI. The MIM_i-patches can be expressed as $\{p_{i,k} \in \mathbb{R}^{(w/m)\times(h/n)\times 3} \mid k = 1, 2, \cdots, n \times m\}$, where k is the order of the patch in MIM_i. For more clarity, MIM_i (Fig. 2d) is divided into 2×4 MIM-patches (Fig. 2e). It is worth noting that the patch should cover an area on the ground that can accommodate a group of pedestrians, as crowd characteristics are required for representing pushing behavior. To summarize, the MIM-patches represent the output of the motion descriptor component and the input of the next component.

3.3 Pushing Detection and Annotation

The primary purpose of this component (Fig. 4a) is to localize the pushing patches in the live stream, as well as blurring and storing the annotated ROI keyframes in the cloud storage. Labeling MIM-patches as pushing or non-pushing is the most important aspect of localizing pushing in the live stream. Therefore, we created an efficient binary classifier by adapting and training the EfficientnetV2B0 CNN architecture [22] from scratch, which is then utilized to label the MIM-patches.

3.3.1 Adapted EfficinetNetV2B0 Architecture

EfficientNetV2B0 is a convolutional neural network belonging to the Effivient-NetV2 family, designed by the Google Brain team [22]. Such a family outperforms state-of-the-art accuracy in different classification tasks with a far smaller model and faster converging speed. EfficientNetV2B0 is the smallest model in this family and achieves high accuracy with minimal computational cost.

Fig. 4b depicts the overall architecture of the modified EfficientNetV2B0, which firstly performs a 3×3 convolution operation on the input image, which has dimensions of $224 \times 224 \times 3$. Then it utilizes a combination of 5 Fused-MBConv (Fused Mobile Inverted Residual Bottleneck Convolution) [36] and 16 MBConv [37] modules for extracting the feature maps ($7\times7\times1280$) from the input image. The model then employs a global average pooling layer and a fully connected layer with a Sigmoid activation function for binary classification. The global average pooling2D layer transforms the dimensions of the stacked feature maps to $1\times1\times1280$ and assigns them to the fully connected layer. Finally, the fully connected layer with a Sigmoid activation function finds the probability δ of the label of the input

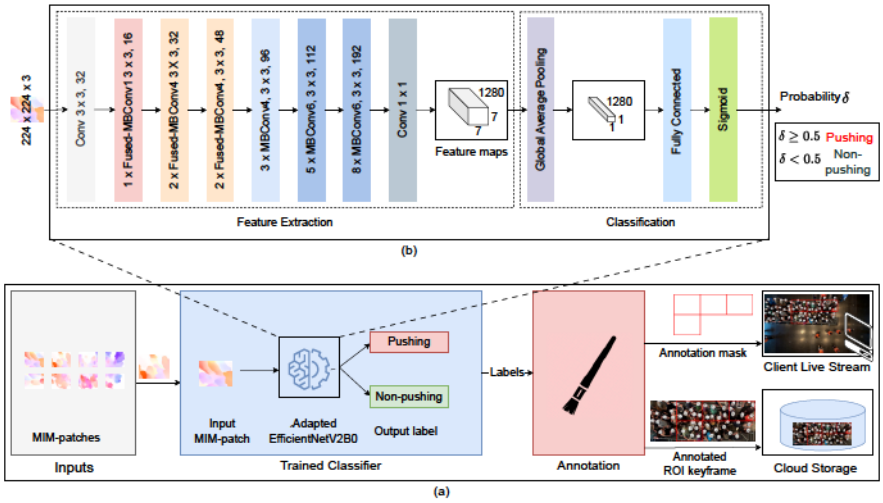

Figure 4: (a) The pipeline of pushing detection and annotation component. (b) Adapted EfficientNetV2B0 Architecture.

MIM-patch. Then, the classifier uses the threshold to determine the class of the MIM-patch as Eq. (3):

$$Class(MIM-patch) = \begin{cases} \text{pushing} & \text{if } \delta \geq 0.5 \\ \text{non-pushing} & \text{if } \delta < 0.5 \end{cases} \quad (5)$$

It's important to note that the classification part of this model differs from the original EfficientNetV2B0, which was designed to classify images into 1,000 categories. However, pushing detection requires labeling the input image into one of two possible classes.

As mentioned above, the main fundamental blocks in EfficientNetV2B0 for feature extraction are MBConv and Fused MBConv [22]. As shown in Fig. 5, MBConv firstly uses a 1×1 convolution operation to expand the input activation maps to increase the depth of the feature maps. Next, 3×3 depthwise convolutions are applied to reduce the computational complexity and the number of parameters. Then, a Squeeze-and-Excitation (SE) block enhances the representation power of the architecture. Finally, another 1×1 convolution is employed to reduce the dimensionality of the output feature maps, producing the final output of this block. Moreover, A residual connection is added to enhance the performance further. Despite depthwise convolutions having fewer parameters, they can not often fully utilize modern accelerators. In contrast, the Fused-MBConv tries to solve this problem by replacing the depthwise and expansion conv1×1 in MBConv conv3×3 with a single regular conv3×3, resulting in a faster training process (see Fig. 5). It is worth mentioning that using only Fused-MBConv in the architecture increases parameters while slowing down the training. Therefore, EfficientNetV2B0 applied a combination of MBConv and Fused-MBConv to improve training speed with a small overhead on parameters and enhance the feature extraction process [22].

The following subsection will discuss the training process for the adapted EfficientNetV2B0 model to classify MIM patches into pushing and non-pushing.

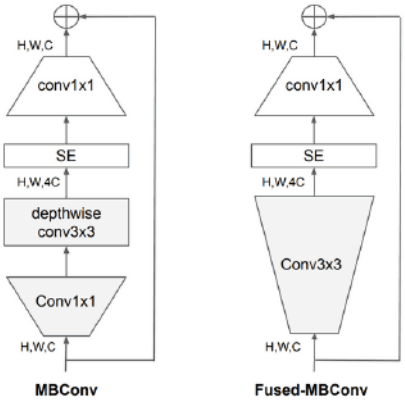

Figure 5: Structure of MBConv and Fused-MBConv [22].

3.3.2 Adapted EfficientNetV2B0 Training

To classify the MIM-patches into pushing and non-pushing categories, we trained the adapted EfficientNetV2B0 model (Fig. 4b) using new training and validation sets comprising both types of MIM-patches (details about the dataset can be found in Section 4.1). The model parameters used during the training process are listed in Table 1, and were chosen based on experimentation to obtain optimal performance with the given dataset. To prevent overfitting, we halted the training if the validation accuracy did not improve after 20 epochs.

Table 1: The hyperparameter values used in the training process.

Parameter	**Value**
Optimizer	Adam
Loss function	Binary cross-entropy
Learning rate	0.001
Batch size	32
Epoch	100

Fig. 4a shows the pipeline of the pushing detection and annotation component. Firstly, the trained classifier labels MIM-patches $p_{i,k}$ received from the previous component. Then, the current component displays an annotation mask of the pushing patches in the live stream on the web client. Simultaneously, it blurs and annotates the corresponding ROI keyframe $\bar{f}_i$ before saving it in the cloud storage. Notably, web clients can access this storage via an internet connection.

4 Evaluation and Results

This section introduces the dataset, implementation details, and performance metrics utilized in evaluating the proposed framework. The results of various experiments conducted to assess the performance of our classifier and the proposed framework are also discussed.

4.1 Dataset Preparation

Here, we explain how we prepared the labeled dataset (training, validation, and test sets) for training and evaluating the adapted EfficientNetV2B0 as well as all

models used in the evaluation. The dataset contains two classes of MIM-patches, which are pushing and non-pushing.

4.1.1 Data Collection

In this section, we discuss the data sources used to obtain our dataset. The sources are mainly based on video experiments of crowded event entrances, trajectory data, and ground truth data for pushing behavior. Five video experiments with their trajectory data are chosen from the data archive hosted by Forschungszentrum Jülich under CC Attribution 4.0 International license [38, 33]. Static top-view cameras were used to record the videos with a frame rate of 25 frames per second. It is worth mentioning that the selected experiments contain varied characteristics, which help to improve the generality of the dataset, as seen in Table 2. The ground truths for the last data source were manually created by social psychologists, who established the definition of pushing behavior in forward motion among crowds [2]. These ground truths indicate whether the behavior of each pedestrian in every frame is classified as either pushing or non-pushing.

Table 2: Characteristics of the selected video experiments.

Video	Entrance type	Gates	Width (m)	Ped.	Dur.	Resolution	ROI coordinates (pixel)	n × m patches *
110	Straight	1	1.2	63	53	1920 × 1440	(374 , 548) , (1382 , 864)	1 × 3
150	Straight	1	5.6	57	57	1920 × 1440	(364 , 200) , (1378 , 1250)	3 × 3
270	Straight	1	3.4	67	59	1920 × 1440	(374 , 330) , (1390 , 1070)	2 × 3
280	Straight	1	3.4	67	67	1920 × 1440	(374 , 330) , (1390 , 1070)	2 × 3
Entrance_2	90° Corner	2	2	123	125	1920 × 1080	(213 , 110) , (1337 , 540)	2 × 4

The video experiments' names are the same as reported in [38, 33]. "Dur." means duration. "Ped." is an abbreviation for the number of pedestrians. ROI coordinates: left–top and bottom–right coordinates of ROI in the pixel unit. $n \times m$: number of rows and columns that are used to divide ROI into $n \times m$ regions, which are required for dividing MIM_i into $n \times m$ MIM_i-patches. * These values ensure that the dimensions of each region on the ground are greater than one meter, which is enough to accommodate a group of pedestrian [7].

4.1.2 Dataset Generation

The methodology of the labeled dataset generation, as seen in Fig. 6, includes three steps: (1) MIM-patches generation, (2) MIM-patches labeling and (3) Labeled dataset generation.

In the MIM-patches generation step, the motion descriptor component was employed (Fig. 3) on the video experiments and their $n \times m$ patches (Table 2) to produce MIM-patches.

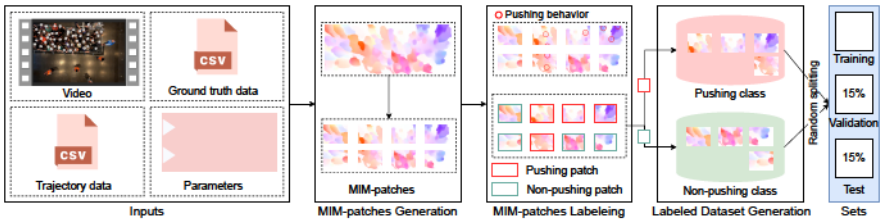

Figure 6: The methodology of the labeled dataset generation.

To increase the number of patches, the component is applied four times for each video with a different commencement; half a second is the delay duration of each time compared to the previous time. According to [7], half a second delay helps to generate diverse MIM-patches, while less than this period may

result in redundant samples. Based on the trajectory and ground truth data, the second step labels the patches as pushing and non-pushing. Patches are classified as pushing if it contains at least one pushing behavior, and non-pushing if no pedestrians engage in pushing behavior. On the other hand, the patches that only show a portion of one pedestrian pushing are discarded; because they do not offer complete information about pushing or non-pushing behavior. According to the labels of the patches, the last step stores the patches in pushing and non-pushing directories to create the labeled dataset. At the end, the generated dataset consists of 2257 pushing and 1684 non-pushing samples. To generate the holdout data, the produced dataset is randomly divided into three sets: 70 % for training, 15 % for validation, and 15 % for testing. This split ratio is one of the most commonly used splitting methods in the deep learning field [39]. Table 3 shows the number of pushing and non-pushing samples in the training, validation, and test sets.

Table 3: A number of samples in training, validation, and test sets in the generated dataset.

	Video	110	150	270	280	Entrance_2	Total
Training	P	122	182	215	258	808	1585
	NP	72	206	197	182	525	1182
	Total	194	388	412	440	1333	2767
Validation	P	26	38	45	55	172	336
	NP	15	44	42	38	112	251
	Total	41	82	87	93	284	587
Test	P	26	38	45	55	172	336
	NP	15	44	42	38	112	251
	Total	41	82	87	93	284	587
All	Total	276	552	586	626	1901	3941

"All" refers to all sets. P means pushing. NP is non-pushing.

4.2 Implementation Details and Evaluation Metrics

In this article, all the experiments and implementations were conducted on Google Colaboratory Pro (with a GPU NVIDIA of 15 GB and system RAM of 12.7 GB), utilizing JavaScript and Python 3 programming languages along with Keras, TensorFlow 2.0, and OpenCV libraries. Furthermore, all models in the experiments were trained using the same hyperparameter values utilized in the training of our adapted version of EfficientNetV2B0 (Table 1).

In order to evaluate the performance of our framework, we utilized a combination of metrics, including accuracy, macro F1-score, and area under the receiver operating characteristic curve (AUC) over the test set. This set of metrics was necessary due to the imbalanced nature of our dataset [40]. In addition to these metrics, computational time was also measured as an essential performance metric. The following provides a detailed explanation of these metrics.

Accuracy: the ratio of successfully classified MIM-patches to the total number of samples in the test set, and mathematically can be defined as

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \tag{6}$$

where TP and TN denote correctly classified pushing and non-pushing patches, respectively. FP and FN represent incorrectly predicted pushing (P) and non-pushing (NP) samples. Accuracy is not enough to evaluate the classifier's performance over an imbalanced dataset, such as our used dataset. Therefore, we used the macro F1-score and AUC metrics, which are valuable for evaluating imbalanced classification problems.

Macro F1-score: the mean of class-wise F1-scores as described in the formula below:

$$Macro\ F1-score = \frac{F1-score(P) + F1-score(NP)}{2}, \tag{7}$$

where F1-score is the harmonic average of precision and recall as described in:

$$F1-score = \frac{2 \times precision \times recall}{precision + recall}, \tag{8}$$

where recall of pushing class is the ratio of correctly classified pushing MIM-patches to all pushing samples, while precision of pushing class is the ratio of correctly classified pushing patches out of all the samples labeled as pushing by the classifier. Recall and precision are defined in Eq. (9) and Eq. (10), respectively.

$$recall = \frac{TP}{TP + FN}, \tag{9}$$

$$precision = \frac{TP}{TP + FP}. \tag{10}$$

AUC is the area under the Receiver Operating Characteristics (ROC) curve. ROC is a graph showing the performance of a classification model at all thresholds. The ROC curve plots the false positive rate on the horizontal axis and the true positive rate on the vertical axis. The AUC value ranges from 0 to 1, while a model with an AUC of 1 is considered perfect, while a value of 0.5 indicates that the model performs no better than random guessing.

Computational time: this metric was employed to calculate how long the proposed framework takes to read, analyze and annotate every input, which is two seconds of stream. In other words, computational time determines whether our framework can detect pushing patches within a reasonable time or not.

4.3 Evaluation of Our Classifier Performance

We conducted three main comparative empirical experiments to evaluate the effect of our modified EfficientNetV2B0 classifier on the performance of the proposed framework. The first experiment compared the proposed classifier against eleven of the most popular CNN architectures. In the second experiment, we compared it to two custom CNN architectures designed for detecting abnormal behavior. Lastly, it was compared to CNN architecture used for pushing detection. Our classifier and all other models were implemented, trained, and assessed utilizing the same MIM-patches dataset, environment, and settings. Moreover, we utilized accuracy, F1-score, and AUC metrics to measure each model's performance.

4.3.1 A Comparison with Eleven Popular CNNs

Table 4 depicts the popular CNN architectures used in the first experiment, as well as the comparison results. It is clear that the adapted version of the EfficientNetV2B0 classifier outperformed the rest of the exploited classifiers. In particular, the proposed classifier achieved 87 % accuracy and 86 % F1-score, whereas the second top model in this comparison, DenseNet169, produced an 83 % level of both accuracy and F1-score. This finding is primarily attributable to EfficientNetV2B0's superior efficiency for feature extraction compared to earlier CNN

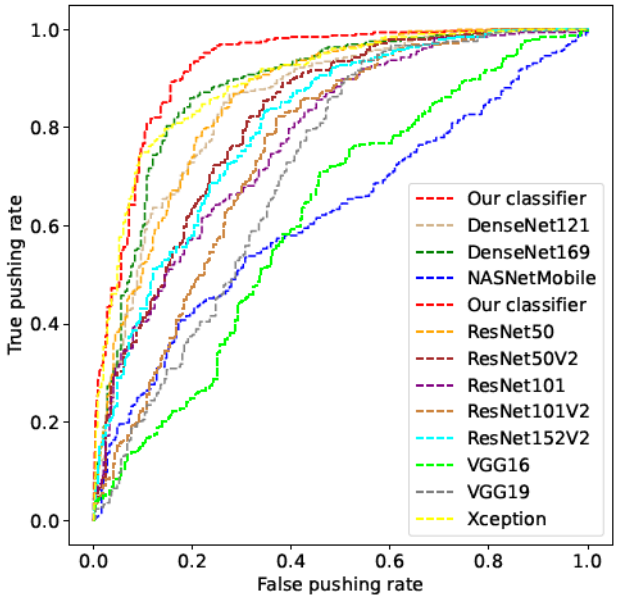

Figure 7: ROC curves of our classifier and eleven popular CNN models.

architectures. The main reason for this efficiency is the combination of MBConv and Fused-MBConv blocks used in EFficientNetV2B0.

Table 4: Comparison results to the well-known CNN-based classifiers.

CNN	Accuracy %	Precision %	Recall %	F1-score %
Xception [41]	81	81	81	81
VGG16 [42]	57	36	29	50
VGG19 [42]	61	61	62	62
ResNet50 [43]	80	79	81	79
ResNet50V2 [44]	77	76	77	75
ResNet101 [43]	72	70	72	70
ResNet101V2 [44]	72	72	72	71
ResNet152V2 [44]	74	73	73	73
DenseNet121 [45]	79	79	79	79
DenseNet169 [45]	83	83	83	83
NASNetMobile [46]	57	56	56	56
Our classifier	**87**	87	86	**86**

Furthermore, as shown in Fig. 7, the proposed classifier obtained the highest AUC score (93 %) among all the models tested, while the next best model achieving 85 %.

4.3.2 A Comparison with Customized CNNs in Abnormal Behavior Detection

Here, we have two objectives, 1) Evaluating the performance of some existing CNN models developed to detect abnormal human behavior for pushing detection purposes. 2) Further evaluation of our classifier. The customized architectures are CNN-1 [10] and CNN-2 [24]. The first architecture, CNN-1, employed 75×75 pixels as an input image. Furthermore, three convolutional layers, batch normalization, and max pooling operations were used for feature extraction. The

developers of this model utilized a fully connected layer with a softmax activation function for classification. The second architecture, CNN-2, downsized the input images to 32×32 pixels before employing three convolutional layers with three max-pooling layers. For classification, it used two fully connected layers, with the first layer based on a ReLU activation function and the second layer employing a softmax activation function.

The results in Fig. 8 and Fig. 9 show that our classifier surpassed the two classifiers in terms of accuracy, F1-score and AUC. Furthermore, as pushing detection in crowded scenarios is highly complex, CNN-1 and CNN-2's simple architectures failed to identify pushing MIM-patches. In particular, CNN-1 outperformed CNN-2, but it still produced unsatisfactory outcomes with accuracy, F1-score, and AUC values of 57 % , 56 %, and 56 %, respectively.

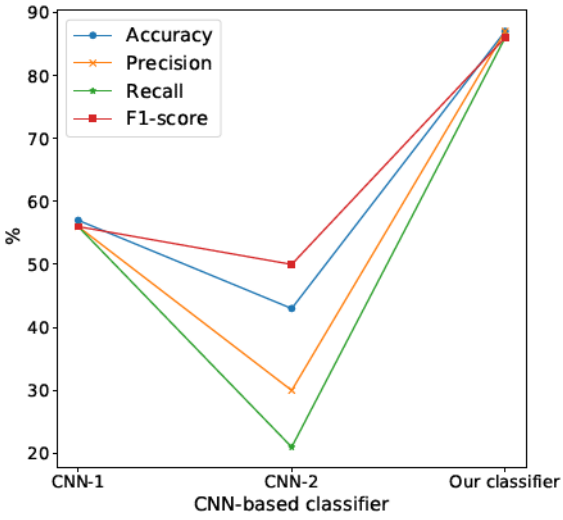

Figure 8: Comparison results of our classifier and two customized CNNs.

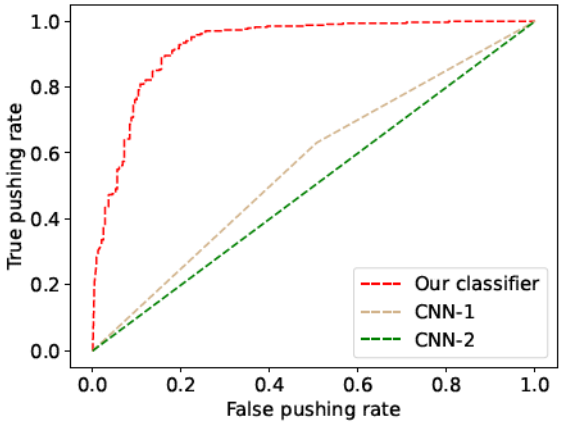

Figure 9: ROC curves of our classifier and the two customized CNNs.

4.3.3 A Comparison with Related Work in Pushing Detection

Here, we compare the proposed classifier with the CNN architecture (EfficientNetV1B0) employed in Ref. [7], which is the only published work for detecting pushing behavior for forward motion. Notably, this work does not meet the early identification requirements. As demonstrated in Table 5 and Fig. 10, our combination of adapted EfficientNetV2B0 and MIMs achieved better performance than integrating EfficientNetV1B0 with MIMs by a margin of at least 3 % in accuracy and F1-score. While EfficientNetV1B0 achieved 91 % AUC, our classifier achieved 93 %. This comparison highlights that our hybrid approach surpassed the state-of-art method in pushing detection regarding the accuracy, F1-score, and AUC metrics. In Section 4.4.2, we will analyze the computational time of both approaches.

Table 5: Comparison results to the state-of-art pushing detection approach.

CNN	Accuracy %	Precision %	Recall %	F1-score %
State-of-art approach [7]	83	83	84	83
Our hybrid approach	**87**	86	87	**86**

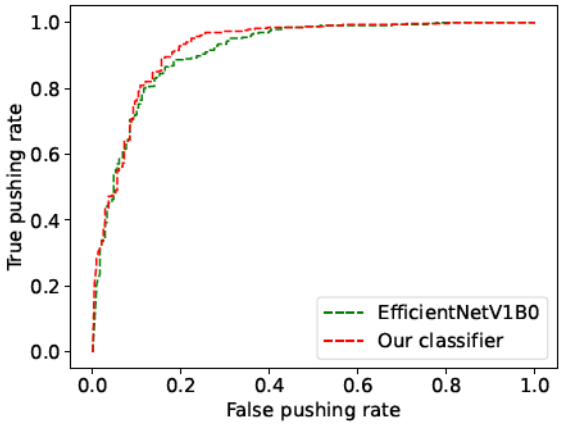

Figure 10: ROC curves of our classifier and EfficientNetV1B0.

To summarize the three comparisons, the new hybrid approach based on adapted EfficientNetV2B0 and MIMs outperformed all other tested combinations of CNN models and MIMs in the experiments. This superiority is due to the power of MBConv and Fused-MBConv blocks used in EfficientNetV2B0 for learning the features. Based on the experiments, it can be concluded that our classifier enhanced the performance of the proposed framework.

4.4 The Overall Framework Evaluation

To evaluate the quality of the proposed framework, we not only evaluated its accuracy and F1-score, but also measured the computational time required for each framework component.

4.4.1 Performance in terms of Accuracy and F1-score

The evaluation methodology used comprises several steps as follows: 1) To simulate acquiring the actual inputs, we created a live video stream of crowded event

entrances using video recordings of entrances (Table 2) and a virtual camera on a web client. In this context, we changed the camera's input to the video recordings. Moreover, we down-scaled the dimensions of each video to half their original resolution to reduce the computational time of the framework. 2) We executed the cloud-based framework to display the live camera stream on the web client, detect pushing patches and record the predicted labels for the test patches in a file. 3) We counted the number of true pushing, false pushing, true non-pushing, and false non-pushing for all videos by comparing the ground truth data with the predicted labels for the test patches, Fig. 11 exhibits the confusion matrix that presents them. 4) Finally, we computed the accuracy and F1-score metrics. After computing the accuracy and F1-score metrics from the values in the confusion matrix (as shown in Fig. 11), our proposed framework achieved an accuracy of 87 %, precision of 87 %, recall of 86 %, and F1-score of 86 %. These results are consistent with the corresponding quantitative outcomes in our adapted EfficientNetV2B0 classifier over the test set.

Table 6: The computational time of motion descriptor and detection components in our and the baseline frameworks.

	Baseline framework		Our framework	
Experiment	Motion information extraction (s)	Detection (s)	Motion descriptor (s)	Detection and annotation (s)
110	19.42	0.15	0.10	0.19
150	19.86	0.47	0.20	0.53
270	19.38	0.32	0.16	0.35
280	19.30	0.30	0.16	0.36
Entrance_2	13.51	0.41	0.11	0.45

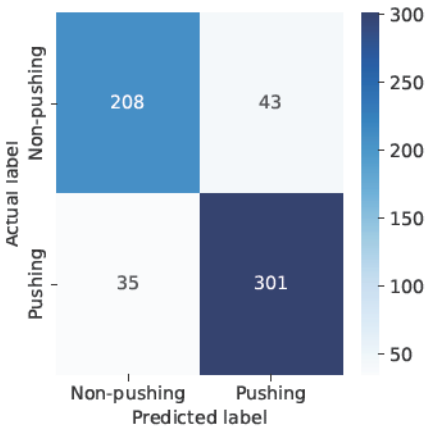

Figure 11: Confusion matrix for the proposed framework on all videos.

4.4.2 Computational Time Analysis

In order to evaluate the overall computational time of the proposed framework, we computed the required time for each component in the framework. Then, we compared the results against the corresponding parts in the baseline framework [7]. After running both the proposed and baseline frameworks in the same environment using twenty inputs, where each input is a two-second video stream,

we calculated the average run-time of all runs. As mentioned in the previous paragraph, our framework read the videos using a live camera, while the baseline framework read the same videos directly. Fig. 12 depicts the time of each component in the proposed framework over every experiment. As indicated by the produced results, the preprocessing component took more than 50 % of the overall time to collect and process keyframes from the client camera stream. The resolution of the keyframes primarily determines the time required for this component. For example, experiments 110, 150, 270, and 280 took roughly the same time because they have the same resolution, while experiment entrance_2 needed less time because its resolution is lower. In contrast, the motion descriptor component took the least time compared to others, where the ROI resolution plays the most critical role in this component speed. While in the pushing detection and annotation component, the number of patches affects the computational time of this component because each patch requires one classification process. Fig. 13 and Table 2 display the ROI resolution and the number of patches in each experiment, respectively.

In general, the computation time increases as the number of patches, frames resolution, and ROI size increase. Fig. 12 shows that our framework needed less than two seconds to collect, process, detect and annotate each input from the live stream camera. This means that our framework can annotate the live camera stream within 4 seconds; two seconds for the input duration and lower than two seconds for identifying the pushing patches.

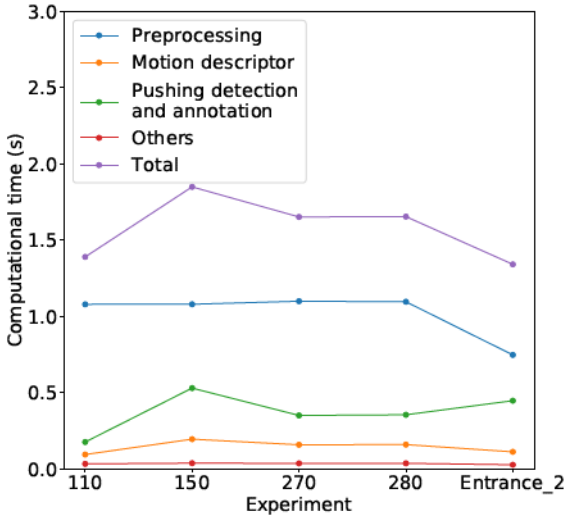

Figure 12: Computational time of each component of the proposed framework for annotating two seconds of stream.

The results in Table 6 show the comparisons between the motion descriptor, and pushing detection and annotation components in our framework with the corresponding parts in the baseline framework. The motion information extraction part in the baseline framework is similar to the motion descriptor component in our framework, whereas the motion information extraction is slow; it needs more than 13.5 seconds to generate MIM-patches from two seconds of the video stream. The main reason for this slowness is that it employed CPU-based RAFT to estimate the optical flow vectors for all pixels in the frame. To address this

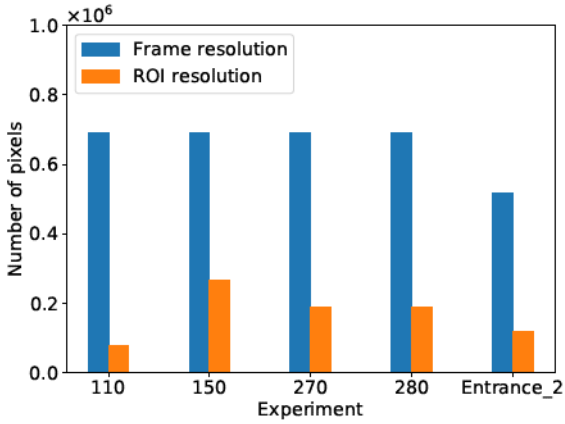

Figure 13: Comparison between frame and ROI resolutions for each experiment.

problem, the motion descriptor in our framework implemented RAFT on GPU to calculate the optical flow vectors for each pixel in ROIs instead of all pixels in the frame. As shown in Fig. 13, the number of pixels in ROIs is lesser than 40 % of the total pixels in the corresponding frames. As a result, the new component took 0.2 seconds or less to produce MIM-patches from the two seconds of the live stream. On the other hand, the baseline framework's detection part is slightly faster than the detection and annotation component in the proposed framework. For example, the previous and new components required 0.47 and 0.53 seconds to work with one input from experiment 150, respectively. It is important to highlight that the detection part in the baseline framework only finds the labels of the patches, whereas the component in our framework labels, annotates, blurs, and stores the inputs.

In summary, the proposed cloud-based framework can annotate the pushing patches in the live camera stream within four seconds and an accuracy rate of 87 %.

5 Conclusion

This paper proposed a novel automatic framework for the early detection of pushing patches in crowded event entrances. The proposed framework is based on live camera streaming technology, cloud environment, visualization method, and deep learning algorithms. The framework first displays the live camera stream of the entrances on the web client in real-time. Then, it relies on the color wheel method and pre-trained RAFT model to extract the visual motion information from the live stream. After that, the EfficientNetV2B0-based classifier is adapted and trained to identify pushing patches from the extracted information. Finally, the framework annotates the pushing patches in the live stream on the web client. Additionally, it stores the annotated data in the cloud storage, where the stored data is blurred to protect people's privacy. In order to train and evaluate the classifier, a new dataset was generated using five real-world video experiments and their associated ground truth data. The experimental results show that the framework identified pushing patches from the live camera stream with 87 % accuracy rate within a reasonable time delay.

One of the current limitations of the proposed framework is that it is only

compatible with a fixed and top-view camera.

In future, the plan is to develop a new pushing data representation method for machine learning. This method aims to generate dynamic patches based on temporal, spatial, and size dimensions, focusing on one pedestrian for labeling each patch. This could potentially help to generate a large dataset with a more efficient sample representation.

Acknowledgment

The authors are thankful to Anna Sieben, Helena Lügering, and Ezel Üsten for the valuable discussions, manual annotation of the pushing behavior in the video experiments, and for revising the proposed framework's output.

Funding

This work was funded by the German Federal Ministry of Education and Research (BMBF: funding number 01DH16027) within the Palestinian-German Science Bridge project framework, and partially by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—491111487.

Ethical Approval

The experiments used in the dataset were conducted according to the guidelines of the Declaration of Helsinki and approved by the ethics board at the University of Wuppertal, Germany. Informed consent was obtained from all subjects involved in the experiments.

Data and code availability

All videos and trajectory data used in generating the patch-based dataset were obtained from the data archive hosted by the Forschungszentrum Jülich under CC Attribution 4.0 International license [33, 38]. The undistorted video experiments, implementation of the proposed framework, as well as codes used for building and training the models, are publicly available at: `https://github.com/PedestrianDynamics/CloudFast-DL4PuDe` (accessed on 15 Jan 2023). The generated patch-based dataset is available from the corresponding authors upon request.

References

[1] J. Adrian, A. Seyfried, and A. Sieben, "Crowds in front of bottlenecks at entrances from the perspective of physics and social psychology," *Journal of the Royal Society Interface*, vol. 17, no. 165, p. 20190871, 2020.

[2] E. Üsten, H. Lügering, and A. Sieben, "Pushing and non-pushing forward motion in crowds: A systematic psychological observation method for rating individual behavior in pedestrian dynamics," *Collective Dynamics*, vol. 7, pp. 1–16, 2022.

[3] M. Haghani, M. Sarvi, and Z. Shahhoseini, "When 'push'does not come to 'shove': Revisiting 'faster is slower'in collective egress of human crowds," *Transportation research part A: policy and practice*, vol. 122, pp. 51–69, 2019.

[4] V. Filingeri, K. Eason, P. Waterson, and R. Haslam, "Factors influencing experience in crowds–the participant perspective," *Applied ergonomics*, vol. 59, pp. 431–441, 2017.

[5] N. R. Johnson, "Panic at "the who concert stampede": an empirical assessment," *Social Problems*, vol. 34, no. 4, pp. 362–373, 1987.

[6] B. Tyagi, S. Nigam, and R. Singh, "A review of deep learning techniques for crowd behavior analysis," *Archives of Computational Methods in Engineering*, pp. 1–29, 2022.

[7] A. Alia, M. Maree, and M. Chraibi, "A hybrid deep learning and visualization framework for pushing behavior detection in pedestrian dynamics," *Sensors*, vol. 22, no. 11, p. 4040, 2022.

[8] A. Mehmood, "Efficient anomaly detection in crowd videos using pre-trained 2d convolutional neural networks," *IEEE Access*, vol. 9, pp. 138 283–138 295, 2021.

[9] A. Al-Dhamari, R. Sudirman, and N. H. Mahmood, "Transfer deep learning along with binary support vector machine for abnormal behavior detection," *IEEE Access*, vol. 8, pp. 61 085–61 095, 2020.

[10] C. Direkoglu, "Abnormal crowd behavior detection using motion information images and convolutional neural networks," *IEEE Access*, vol. 8, pp. 80 408–80 416, 2020.

[11] A. Alia and A. Taweel, "Enhanced binary cuckoo search with frequent values and rough set theory for feature selection," *IEEE access*, vol. 9, pp. 119 430–119 453, 2021.

[12] A. F. Alia and A. Taweel, "Feature selection based on hybrid binary cuckoo search and rough set theory in classification for nominal datasets," *algorithms*, vol. 14, no. 21, p. 65, 2017.

[13] H. Gan, C. Xu, W. Hou, J. Guo, K. Liu, and Y. Xue, "Spatiotemporal graph convolutional network for automated detection and analysis of social behaviours among pre-weaning piglets," *Biosystems Engineering*, vol. 217, pp. 102–114, 2022.

[14] H. Gan, M. Ou, E. Huang, C. Xu, S. Li, J. Li, K. Liu, and Y. Xue, "Automated detection and analysis of social behaviors among preweaning piglets using key point-based spatial and temporal features," *Computers and Electronics in Agriculture*, vol. 188, p. 106357, 2021.

[15] L. Li, S. Zhang, and B. Wang, "Apple leaf disease identification with a small and imbalanced dataset based on lightweight convolutional networks," *Sensors*, vol. 22, no. 1, p. 173, 2021.

[16] P. Wang, E. Fan, and P. Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021.

[17] E. Duman and O. A. Erdem, "Anomaly detection in videos using optical flow and convolutional autoencoder," *IEEE Access*, vol. 7, pp. 183 914–183 923, 2019.

[18] T. Alafif, A. Hadi, M. Allahyani, B. Alzahrani, A. Alhothali, R. Alotaibi, and A. Barnawi, "Hybrid classifiers for spatio-temporal abnormal behavior detection, tracking, and recognition in massive hajj crowds," *Electronics*, vol. 12, no. 5, p. 1165, 2023.

[19] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision*. Springer, 2020, pp. 402–419.

[20] D. F. Tom Runia, "Optical flow visualization," 2020. [Online]. Available: https://github.com/tomrunia/OpticalFlow_Visualization

[21] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International journal of computer vision*, vol. 92, no. 1, pp. 1–31, 2011.

[22] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 096–10 106.

[23] P. Kuppusamy and V. Bharathi, "Human abnormal behavior detection using cnns in crowded and uncrowded surveillance–a survey," *Measurement: Sensors*, vol. 24, p. 100510, 2022.

[24] N. C. Tay, T. Connie, T. S. Ong, K. O. M. Goh, and P. S. Teh, "A robust abnormal behavior detection method using convolutional neural network," in *Computational Science and Technology*. Springer, 2019, pp. 37–47.

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

[26] M. Xu, X. Yu, D. Chen, C. Wu, and Y. Jiang, "An efficient anomaly detection system for crowded scenes using variational autoencoders," *Applied Sciences*, vol. 9, no. 16, p. 3337, 2019.

[27] S. Smeureanu, R. T. Ionescu, M. Popescu, and B. Alexe, "Deep appearance features for abnormal behavior detection in video," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 779–789.

[28] Z. Ilyas, Z. Aziz, T. Qasim, N. Bhatti, and M. F. Hayat, "A hybrid deep network based approach for crowd anomaly detection," *Multimedia Tools and Applications*, pp. 1–15, 2021.

[29] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.

[30] Y. Hu, "Design and implementation of abnormal behavior detection based on deep intelligent analysis algorithms in massive video surveillance," *Journal of Grid Computing*, vol. 18, no. 2, pp. 227–237, 2020.

[31] A. A. Almazroey and S. K. Jarraya, "Abnormal events and behavior detection in crowd scenes based on deep learning and neighborhood component analysis feature selection," in *Joint European-US Workshop on Applications of Invariance in Computer Vision*. Springer, 2020, pp. 258–267.

[32] S. Zhou, W. Shen, D. Zeng, M. Fang, Y. Wei, and Z. Zhang, "Spatial–temporal convolutional neural networks for anomaly detection and localization in crowded scenes," *Signal Processing: Image Communication*, vol. 47, pp. 358–368, 2016.

[33] "Entrance 2, entry with guiding barriers (corridor setup)," http://doi.org/10.34735/ped.2013.1, Pedestrian Dynamics Data Archive hosted by the Forschungszentrum Jülich, 2013.

[34] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[35] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6044–6053.

[36] S. Gupta and M. Tan, "Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl," *Google AI Blog*, vol. 2, p. 1, 2019.

[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[38] "Crowds in front of bottlenecks from the perspective of physics and social psychology," http://doi.org/10.34735/ped.2018.1, Pedestrian Dynamics Data Archive hosted by the Forschungszentrum Jülich, 2018.

[39] B. Genc and H. Tunc, "Optimal training and test sets design for machine learning," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 27, no. 2, pp. 1534–1545, 2019.

[40] Z. DeVries, E. Locke, M. Hoda, D. Moravek, K. Phan, A. Stratton, S. Kingwell, E. K. Wai, and P. Phan, "Using a national surgical database to predict complications following posterior lumbar surgery and comparing the area under the curve and f1-score for the assessment of prognostic capability," *The Spine Journal*, vol. 21, no. 7, pp. 1135–1142, 2021.

[41] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] ——, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[45] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[46] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

Publication III

A Novel Voronoi-based Convolutional Neural Network Framework for Pushing Person Detection in Crowd Videos

This article has been published as Alia, Ahmed, Mohammed Maree, Mohcine Chraibi, and Armin Seyfried. "A Novel Voronoi-based Convolutional Neural Network Framework for Pushing Person Detection in Crowd Videos." Complex & Intelligent Systems (2024).

Author's Contributions

Conceptualization: **Ahmed Alia**
Methodology: **Ahmed Alia**, Armin Seyfried
Software: **Ahmed Alia**
Validation: **Ahmed Alia**
Formal Analysis: **Ahmed Alia**
Visualization: **Ahmed Alia**
Data Curation: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
writing—review and editing **Ahmed Alia**, Mohammed Maree, Mohcine Charaibi, Armin Seyfried
GitHub Repository Creation: **Ahmed Alia**

GitHub repository: `https://github.com/PedestrianDynamics/VCNN4PuDe`

Preprint DOI: `https://doi.org/10.1007/s40747-024-01422-2`.

A Novel Voronoi-based Convolutional Neural Network Framework for Pushing Person Detection in Crowd Videos

Ahmed Alia[1,2,3*], Mohammed Maree[4], Mohcine Chraibi[1], Armin Seyfried[1,2]

[1] Institute for Advanced Simulation, Forschungszentrum Jülich, 52425 Jülich, Germany.

[2] Computer Simulation for Fire Protection and Pedestrian Traffic, University of Wuppertal, 42285 Wuppertal, Germany.

[3] Department of Information Technology, Faculty of Engineering and Information Technology, An-Najah National University, Nablus, Palestine.

[4] Department of Information Technology, Faculty of Engineering and Information Technology, Arab American University, Jenin, Palestine.

Corresponding author: a.alia@fz-juelich.de

Abstract Analyzing the microscopic dynamics of pushing behavior within crowds can offer valuable insights into crowd patterns and interactions. By identifying instances of pushing in crowd videos, a deeper understanding of when, where, and why such behavior occurs can be achieved. This knowledge is crucial to creating more effective crowd management strategies, optimizing crowd flow, and enhancing overall crowd experiences. However, manually identifying pushing behavior at the microscopic level is challenging, and the existing automatic approaches cannot detect such microscopic behavior. Thus, this article introduces a novel automatic framework for identifying pushing in videos of crowds on a microscopic level. The framework comprises two main components: i) Feature extraction and ii) Video detection. In the feature extraction component, a new Voronoi-based method is developed for determining the local regions associated with each person in the input video. Subsequently, these regions are fed into EfficientNetV1B0 Convolutional Neural Network to extract the deep features of each person over time. In the second component, a combination of a fully connected layer with a Sigmoid activation function is employed to analyze these deep features and annotate the individuals involved in pushing within the video. The framework is trained and evaluated on a new dataset created using six real-world experiments, including their corresponding ground truths. The experimental findings demonstrate that the proposed framework outperforms state-of-the-art approaches, as well as seven baseline methods used for comparative analysis.

*Keywords:*Artificial Intelligence, Deep Learning, Convolutional Neural Network, Intelligent Video and Image Analytics, Intelligent Systems, Pushing Detection, Crowd Dynamics

1 Introduction

With the rapid development of urbanization, the dense crowd has become widespread in various locations, such as religious sites, train stations, concerts, stadiums, malls, and famous tourist attractions. In such highly dense crowds, pushing behavior can easily arise, which may further increase crowd density. This could pose a threat not only to people's comfort but also to their safety [1, 2, 3, 4]. People

may start pushing for different reasons. 1) Saving their lives in emergencies or tense scenarios [5, 6, 7]. 2) Grabbing a limited resource, such as gaining access to a crowded subway train [8, 9]. 3) Accessing a venue more quickly; for instance, in crowded event entrances, some pedestrians start pushing others to enter the event faster [10, 11, 12]. The focus of this article is the pushing that occurs in crowded event entrances due to the availability of public real-world experiments about such entrances.

In this context, Lügering et al. [10] defined pushing as "a behavior that can involve using arms, shoulders, or elbows; or simply the upper body, in which one person actively applies force to another person (or people) to overtake, while shifting their direction to the side or back, or force them to move forward more quickly". Additionally, using gaps in the crowd is considered as a strategy of pushing because it is a form of overtaking [10]. For more clarity, the definition of pushing behavior adopted in this article, published in 2022, describes it as a tactic pedestrians use to move forward more quickly through dense crowds [10], rather than as a strategy for fighting [13].

Understanding the microscopic dynamics of pushing plays a pivotal role in effective crowd management, helping safeguard the crowd from tragedies and promoting overall well-being [14, 1, 15, 16, 17]. The study [10] has introduced a manual rating system to understand pushing dynamics at the microscopic level. The method relies on two trained psychologists to classify pedestrians' behaviors over time in a video of crowds into pushing or non-pushing categories, helping to know when, where, and why pushing behavior occurs. However, this manual method is time-consuming, tedious and prone to errors in some scenarios. Additionally, it requires trained observers, which may not always be feasible. Consequently, an increasing demand is for an automatic approach to identify pushing at the microscopic level within crowd videos. Detecting pushing behavior automatically is a demanding task that falls within the realm of computer vision. This challenge arises from several factors, such as dense crowds gathering at event entrances, the varied manifestations of pushing behavior, and the significant resemblance and overlap between pushing and non-pushing actions.

Recently, machine learning algorithms, particularly Convolutional Neural Network (CNN) architectures, have shown remarkable success in various computer vision tasks, including face recognition [18], object detection [19, 20, 21], image classification [22] and abnormal behavior detection [23]. One of the key reasons for this success is that CNN can learn the relevant features [24, 25, 26] automatically from data without human supervision [27, 28]. As a result of CNN's success in abnormal behavior detection, which is closely related to pushing detection, some studies have started to automate pushing detection using CNN models [29, 30, 31]. For instance, Alia et al. [29] introduced a deep learning framework that leverages deep optical flow and CNN models for pushing patch detection in video recordings. Another study [30] introduced a fast hybrid deep neural network model based on GPU to enhance the speed of video analysis and pushing patch identification. Similarly, the authors of [31] developed an intelligent framework that combines deep learning algorithms, a cloud environment, and live camera stream technology to annotate the pushing patches in real-time from crowds accurately. Yet, the current automatic methods focus on identifying pushing behavior at the level of regions (macroscopic level) rather than at the level of individuals (microscopic level), where each region can contain a group of persons. For more clarity, "patch level" refers to identifying regions that contain at least one person engaged

in pushing behavior. In contrast, "individual level" refers to detecting the persons joining in pushing. Fig. 1a shows a visualized example that demonstrates the identification of pushing behavior based on levels of patch and individual. In other words, the automatic approaches reported in the literature can not detect pushing at the microscopic level, limiting their contributions to help comprehend pushing dynamics in crowds. For example, they cannot accurately determine the relationship between the number of individuals involved in pushing behavior and the onset of critical situations, thereby hindering a precise understanding of when a situation may escalate to a critical level.

To overcome the limitations of the aforementioned methods, this article introduces a novel Voronoi-based CNN framework for automatically identifying individuals engaging in pushing behavior in crowd video recordings, using a single frame every second. It requires two types of input: the crowd's video recordings and individuals' trajectory data. The proposed framework comprises two components: feature extraction and detection. The first component utilizes a novel Voronoi-based EfficientNetV1B0 CNN architecture for feature extraction. The Voronoi-based method [32] integrates the Voronoi Diagram, Convex Hull [33], and a new dummy point generation technique to identify the local region of each person every second in the input video. The boundaries of local regions are determined based on the input pedestrian trajectory data. Subsequently, the EfficientNetV1B0 (excluding its original multiclass classification part) model [34] is used to extract deep features from these regions. In this article, the local region is defined as the zone focusing only on a single person (target person), including his surrounding spaces and physical interactions with his direct neighbors. This region is crucial in guiding the proposed framework to focus on microscopic behavior. On the other hand, the second component utilizes a fully connected layer coupled with a Sigmoid activation function to create a binary classification working instead of the original multi-class classification part in the EfficentNetV1B0. This adaptation is crucial for processing deep features and effectively differentiating between pushing and non-pushing behaviors in individuals. Finally, the adapted EfficientNetV1B0 is trained from scratch on a dataset of labeled local regions generated from six real-world video experiments with their ground truths [35].

The main contributions of this work are summarized as follows:

1. To the best of our knowledge, this article introduces the first framework for automatically identifying pushing behavior at the individual level in videos of human crowds. In contrast, previous works in the literature [29, 31, 30] have focused on detecting such behavior at the patch level. Fig. 1a provides a visualization of detection methods at individual and patch levels.

2. The framework utilizes a novel Voronoi-based approach with a trained and adapted EfficientNetV1B0-based CNN model. The novel Voronoi-based approach incorporates the Voronoi Diagram, Convex Hull, and a new dummy point generation technique.

3. The article introduces a new dataset comprising both pushing and non-pushing local regions, derived from six real-world experiments, each with corresponding ground truths. It is important to note that this dataset differs from those in previous works [29, 31]. In the current dataset, the samples consist of local regions. In contrast, in the previous datasets, each sample represents a visual motion information map of the crowd within a specific

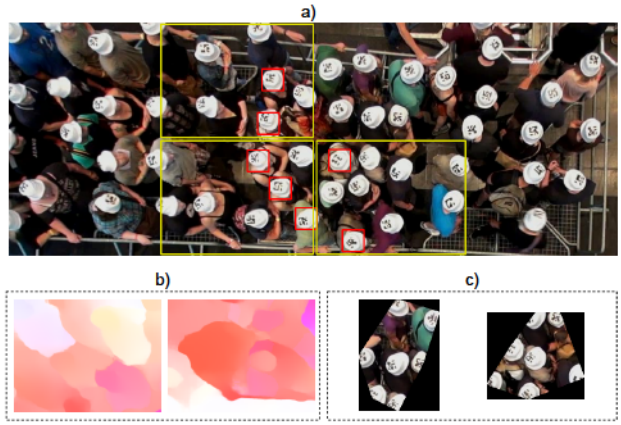

Figure 1: Illustrated examples: a) Annotated frame with yellow rectangles, each representing a patch, showing an example of pushing behavior detection at the patch level. Annotated frame with red squares, illustrating pushing behavior detection at the individual level. b) Examples of motion information map samples. c) Examples of local region samples.

patch — a region ranging from 1.2 to 2.4 m^2 on the ground — and a specific duration, Fig. 1b and c show examples of motion information maps and local region samples. Additionally, the size of the new dataset is three times larger than that of the previous datasets.

The remainder of this article is organized as follows. Section 2 reviews some automatic approaches to abnormal behavior detection in videos of crowds. The architecture of the proposed framework is introduced in Section 3. Section 4 presents the processes of training and evaluating the framework. Section 5 discusses experimental results and comparisons. Finally, the conclusion and future work are summarized in Section 6.

2 Related Work

This section begins by providing an overview of Voronoi Diagrams-based Deep Learning in Computer Vision. Subsequently, it explores CNN-based approaches for automatic video analysis and detecting abnormal behavior in crowds. The discussion then shifts to techniques specifically designed for automatically identifying pushing incidents within crowd videos.

2.1 Voronoi Diagram-based Deep Learning in Computer Vision

A Voronoi diagram partitions a plane into regions based on the distance to a set of points, known as seeds. Each region is defined by all points closer to its corresponding seed than any other (for more details about Voronoi diagram, see Section 3.1.1, direct neighbor identification step). Recently, integrating Voronoi diagrams with deep learning algorithms has led to the development of efficient applications in various computer vision tasks. These tasks include visual-query-based retrieval systems over image datasets [36], object detection [37], synthetic dataset generation [38], and image classification [39, 40].

2.2 CNN-based Abnormal Behavior Detection

Typically, behavior is considered abnormal when it is seen as unusual in specific contexts. This implies that the definition of abnormal behavior depends on the situation [41]. To illustrate, running inside a bank might be considered abnormal behavior, while running at a traffic light could be viewed as normal [42]. Several behaviors have been automatically addressed in abnormal behavior detection applications in crowds, including walking in the wrong direction [43], running away [44], sudden people grouping or dispersing [45], human falls [46], suspicious behavior, violent acts [47], abnormal crowds [48], hitting, and kicking [13].

Tay et al. [41] developed a CNN-based method for identifying abnormal activities from videos. The researchers specifically designed and trained a customized CNN to extract features and label samples, using a dataset comprising both normal and abnormal samples. Similarly, study [49] introduced a novel CNN-based system for detecting abnormal crowd behavior in indoor and outdoor settings. This system leverages the strengths of pre-trained DenseNet121 and EfficientNetV2 models, which were fine-tuned for enhanced feature extraction. Subsequently, the study introduced innovative modifications to multistage and multicolumn models, specifically tailored to identifying various crowd behaviors, including sudden motion changes, panic events, and human flock movement. A new method using CNNs has been developed in [50] for the real-time detection of abnormal situations, such as violent behaviors. This method comprises the Convolutional Block Attention Module combined with the ResNet50 architecture to enhance feature extraction. In [51], the authors proposed a Densely Connected Convolutional Neural Network (DenseNet121)-based approach to identify abnormal behaviors in surveillance video feeds, achieving near real-time performance. Almahadin et al. [52] have developed an innovative model to identify abnormal behaviors in video sequences of crowded scenes. They integrated convolutional layers, Long Short-Term Memory networks, and a sigmoid-based output layer to extract spatiotemporal features and detect abnormal behaviors effectively. Nevertheless, constructing accurate CNNs requires a substantial training dataset, often unavailable for many human behaviors.

To address the limited availability of large datasets containing both normal and abnormal behaviors, some researchers have employed one-class classifiers using datasets that exclusively consist of normal behaviors. Creating or acquiring a dataset containing only normal behavior is comparatively easier than obtaining a dataset that includes both normal and abnormal behaviors. [53, 54]. The fundamental concept behind the one-class classifier is to learn exclusively from normal behaviors, thereby establishing a class boundary between normal and undefined (abnormal) classes. For example, Sabokrou et al. [53] utilized a pre-trained CNN to extract motion and appearance information from crowded scenes. They then employed a one-class Gaussian distribution to build the classifier, utilizing datasets of normal behavior. Similarly, in [54, 55], the authors constructed one-class classifiers by leveraging a dataset composed exclusively of normal samples. In [54], Xu et al. employed a convolutional variational autoencoder to extract features, followed by the use of multiple Gaussian models to detect abnormal behavior. Meanwhile, in [55], a pre-trained CNN model was employed for feature extraction, while one-class support vector machines were utilized for detecting abnormal behavior. Another study by Ilyas et al. [56] conducted a separate study where they utilized a pre-trained CNN along with a gradient sum of the frame dif-

ference to extract meaningful features. Subsequently, they trained three support vector machines on normal behavior data to identify abnormal behaviors. In general, one-class classifiers are frequently employed when the target behavior class or abnormal behavior is rare or lacks a clear definition [57]. However, pushing behavior is well-defined and not rare, particularly in high-density and competitive situations. Furthermore, this type of classifier considers new normal behavior as abnormal.

In order to address the limitations of CNN-based and one-class classifier approaches, multiple studies have explored the combination of multi-class CNNs with one or more handcrafted feature descriptors [56, 23]. In these hybrid approaches, the descriptors are employed to extract valuable information from the data. Subsequently, CNN learns and identifies relevant features and classifications based on the extracted information. For instance, Duman et al. [42] employed the classical Farnebäck optical flow method [58] and CNN to identify abnormal behavior. They used Farnebäck and CNN to estimate direction and speed information and then applied a convolutional long short-term memory network to build the classifier. Hu et al. [59] employed a combination of the histogram of gradient and CNN for feature extraction, while a least-squares support vector was used for classification. Direkoglu [23] utilized the Lucas-Kanade optical flow method and CNN to extract relevant features and identify "escape and panic behaviors". Almazroey et al. [60] used Lucas-Kanade optical flow, a pre-trained CNN, and feature selection methods (specifically neighborhood component analysis) to extract relevant features. These extracted features were then used to train a support vector machine classifier. A framework to analyze video sequences in large-scale Hajj crowds was proposed by Aldayri et al. [61]. It integrates convolution operations, Convolutional Long Short-Term Memory, and Euclidean Distance to achieve its objectives.

Hybrid-based approaches could be more suitable for automatically detecting pushing behavior due to the limited availability of labeled pushing data. Nevertheless, most of the reviewed hybrid-based approaches for abnormal behavior detection may be inefficient for detecting pushing since 1) The descriptors used in these approaches can only extract limited essential data from high-density crowds to represent pushing behavior. 2) Some CNN architectures commonly utilized in these approaches may not be effective in dealing with the increased variations within pushing behavior (intra-class variance) and the substantial resemblance between pushing and non-pushing behaviors (high inter-class similarity), which can potentially result in misclassification.

2.3 CNN-based Pushing Behavior Detection

In more recent times, a few approaches that merge effective descriptors with robust CNN architectures have been developed for detecting pushing regions in crowds. For example, Alia et al. [29] introduced a hybrid deep learning and visualization framework to aid researchers in automatically detecting pushing behavior in videos. The framework combines deep optical flow and visualization methods to extract the visual motion information from the input video. This information is then analyzed using an EfficientNetV1B0-based CNN and false reduction algorithms to identify and label pushing patches in the video. The framework has a drawback in terms of speed, as the motion extraction process is based on a CPU-based optical flow method, which is slow. Another study [30] presented a

fast hybrid deep neural network model that labels pushing patches in short videos lasting only two seconds. The model is based on an EfficientNetB1-based CNN and GPU-based deep optical flow.

To support the early detection of pushing patches within crowds, the study [31] presented a cloud-based deep learning system. The primary goal of such a system is to offer organizers and security teams timely and valuable information that can enable early intervention and mitigate hazardous situations. The proposed system relies mainly on a fast and accurate pre-trained deep optical flow, an adapted version of the EfficientNetV2B0-based CNN, a cloud environment and live stream technology. Simultaneously, the optical flow model extracts motion characteristics of the crowd in the live video stream, and the classifier analyzes the motion to label pushing patches directly on the stream. Moreover, the system stores the annotated data in the cloud storage, which is crucial to assist planners and organizers in evaluating their events and enhancing their future plans.

To the best of our knowledge, current pushing detection approaches in the literature primarily focus on identifying pushing at the patch level rather than at the individual level. However, identifying the individuals involved in pushing would be more helpful for understanding the pushing dynamics. Hence, this article introduces a new framework for detecting pushing individuals in videos of crowds. The following section provides a detailed discussion of the framework.

3 Proposed Framework Architecture

This section describes the proposed framework for automatic pushing person detection in videos of crowds. As depicted in Fig. 2, there are two main components: feature extraction and detection. The first component extracts the deep features from each individual's behavior. In contrast, the second component analyzes the extracted deep features and annotates the pushing persons within the input video. The following sections will discuss both components in more detail.

3.1 Feature Extraction Component

This component aims to extract deep features from each individual's behavior, which can be used to classify pedestrians as pushing or non-pushing. To accomplish this, the component consists of two modules: Voronoi-based local region extraction and EfficientNetV1B0-based deep feature extraction. The first module selects a frame from the input video every second and identifies each person's local region based on the pedestrian trajectory data within those extracted frames. Subsequently, the second module extracts deep features from each local region and feeds them to the next component for pedestrian detection. Before diving into these modules, let us define the local region term at one frame.

A frame f_t is captured every second from the input video. Here, t represents the timestamp, in seconds, since the start of the video and can range from 1 to T, where T is the total duration of the video in seconds. We can analyze individual pedestrians within each of these frames, such as f_t. The positions are given by discrete trajectories that assign a position $\langle x, y \rangle_i$ to each person i at frame f_t: $(\{[i, t, x, y]\}_{t=1}^{T})$. Let $\mathcal{N}_i$ denote the set of pedestrians whose Voronoi cells are adjacent to that of pedestrian i. Specifically, pedestrian j belongs to $\mathcal{N}_i$ if and only if their Voronoi cells share a boundary. The local region for pedestrian i at f_t, $\mathcal{L}_i$, forms a two-dimensional closed polygon, defined by the positions of all

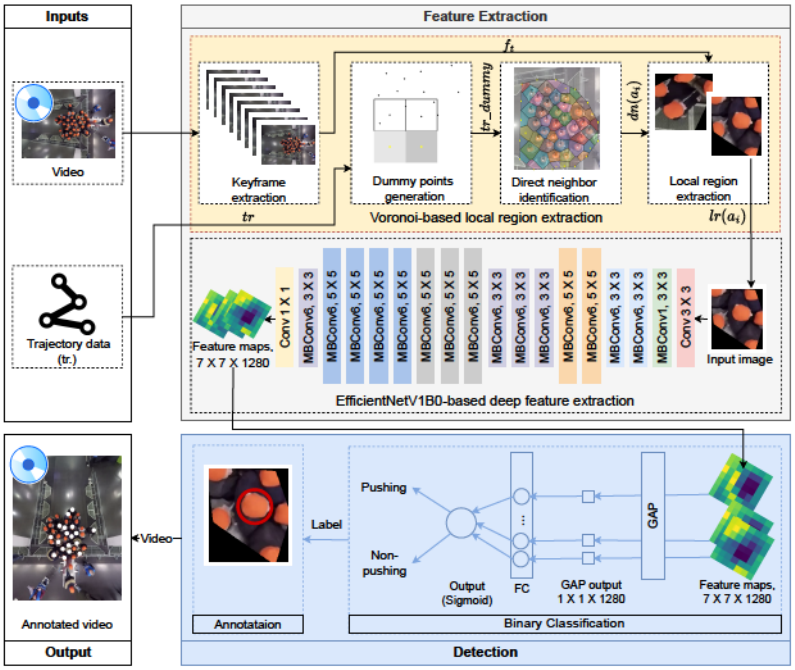

Figure 2: The architecture of the proposed framework. In f_t, f signifies an extracted frame, while t indicates its timestamp in seconds, counted from the beginning of the input video (with t taking values like $1, 2, 3, \ldots$). For a target person i at f_t, $\mathcal{L}_i(f_t)$ denotes the local region, while $\mathcal{N}_i(f_t)$ represents the direct neighbors of i at f_t. The input trajectory data denoted by tr assists the Voroni-based local region extraction method in identifying the boundaries of each $\mathcal{L}_i(f_t)$. The term tr_dummy refers to the data that includes the generated dummy points and those in tr. FC stands for fully connected layer, while GAP refers to global average pooling.

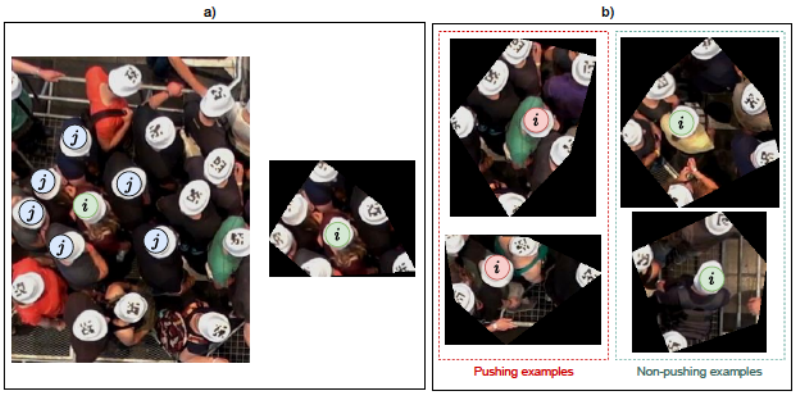

Figure 3: An illustration of direct neighbors (a) and examples of local regions (b). The red circles represent individuals engaged in pushing, while the green circles represent individuals not involved in pushing. Direct neighbors j of a person i are indicated with blue circles.

pedestrians in $\mathcal{N}_i$. As illustrations, Fig. 3a provides examples of both $\mathcal{N}_i$ (left image) and $\mathcal{L}_i$ (right image).

The region $\mathcal{L}_i$ encapsulates the crowd dynamics around individual i, reflecting potential interactions between i and its neighbors $\mathcal{N}_i$. Notably, the characteristics around a pushing individual might diverge from those around a non-pushing one, a

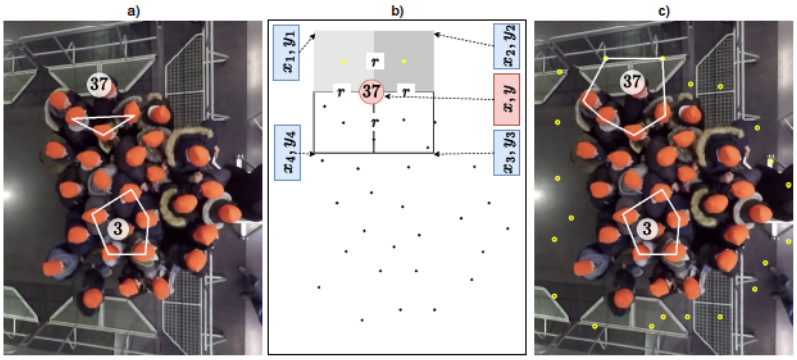

Figure 4: An illustration of the effect of dummy points on creating the local regions, as well as a sketch of the dummy points generation technique. a) $\mathcal{L}_{37}$ and $\mathcal{L}_3$ without dummy points. b) a sketch of the dummy points generation technique. c) $\mathcal{L}_{37}$ and $\mathcal{L}_3$ with dummy points. The white polygon represents the border of the local regions. Yellow small circles refer to the generated dummy points, while black points in b denote the positions of pedestrians. r is the dimension of each square.

distinction pivotal for highlighting pushing behaviors. Fig. 3b showcases examples of such $\mathcal{L}_i$ regions for pushing and non-pushing individuals. The following section introduces a novel method for extracting $\mathcal{L}_i$.

3.1.1 Voronoi-based Local Region Extraction

This section presents a novel method for extracting the local regions of pedestrians from the input video over time t. Besides the input video recordings, this method requires trajectory data $\{[i, t, x, y]\}_{t=1}^{T}$ to determine the coordinates $\langle x, y\rangle_j$ at frame f_t, which aids in identifying the corners of the polygonal local region $\mathcal{L}_i$ at f_t. The technique consists of several steps: frame extraction, dummy points generation, direct neighbor identification, and local region extraction.

Based on the definition of $\mathcal{L}_i$ presented earlier, the determination of each i's regional boundary is contingent upon $\mathcal{N}_i$ at f_t ($\mathcal{N}_i(f_t)$). Nonetheless, this definition might not always guarantee the inclusion of every i within their respective local region. This can be particularly evident when i at f_t lacks neighboring points from all directions, exemplified by person 37 in Fig. 4a. To address this issue, we introduce a step to generate dummy points. This involves adding points around each i at f_t in areas where they lack direct neighbors. This ensures every i remains encompassed within their local regions, as illustrated by person 37 in Fig. 4c. For this purpose, as depicted in Fig. 4b and Algorithm 1, firstly, this step involves reading the trajectory data of i that corresponds to f_t (Algorithm 1, lines 1-8). Concurrently, the area surrounding every i is divided into four equal square regions, each can accommodate at least one i (Algorithm 1, lines 9-17). The location $\langle x, y\rangle_i$ corresponds to the first 2D coordinate of each region (Algorithm 1, lines 12-13). In contrast, the remaining 2D coordinates ($\langle x1, y1\rangle$,$\langle x2, y2\rangle$,$\langle x3, y3\rangle$, $\langle x4, y4\rangle$) required for identifying the regions can be determined by:

$$\begin{aligned} \langle x1, y1\rangle &= \langle x - r, y + r\rangle \\ \langle x2, y2\rangle &= \langle x + r, y + r\rangle \\ \langle x3, y3\rangle &= \langle x + r, y - r\rangle \\ \langle x4, y4\rangle &= \langle x - r, y - r\rangle, \end{aligned} \tag{1}$$

Algorithm 1 Pseudo code for generating dummy points.

Inputs:

tr= $\{row_1, row_2, row_3, \dots\}$, // A file of pedestrian trajectory data
where each row =[pedestrian Id, frame order in the corresponding video, x-coordinate of pedestrian Id, y-coordinate of pedestrian Id]
fps: the frame rate in the corresponding video
r: the dimension of each square region in pixel unit

Outputs:

// A file of pedestrian trajectory data with dummy points
tr_dummy=$tr \cup \{row_1, row_2, row_3, \dots\}$
where row =["dummy point", frame order in the corresponding video, x-coordinate of the dummy point, y-coordinate of the dummy point]

```
1: file ← open(tr)
2: file_dummy ← open(tr_dummy)
3: while not EOF(file) do
4:     rec ← read(file)
5:     if rec[1] % fps = 0 then
6:         write(rec) to file_dummy
7:     end if
8: end while
9: regions ← [[]]
10: while not EOF(file_dummy) do
11:     rec ← read(file_dummy)
12:     x ← rec[2]
13:     y ← rec[3]
14:     append ([x − r, y + r]) to regions
15:     append ([x + r, y + r]) to regions
16:     append ([x + r, y − r]) to regions
17:     append ([x − r, y − r]) to regions
18:     while corner in regions do
19:         if empty(area([x, y], corner]) then
20:             dummy_point ← [(x+corner[0])/2 , (y+corner[1])/2]
21:             dumy_rec ← [0, rec[0], dummy_point[0], dummy_point[1]]
22:             append (dumy_rec) to file_dummy
23:         end if
24:     end while
25: end while
26: tr.close()
27: tr_dummy.close()
```

where r is the dimension of each square region. Subsequently, each region is checked to verify if it has any pedestrians. In case a region is empty, a dummy point in its center is appended to the input trajectory data. Fig. 4b illustrates an example of four regions surrounding person 37 and two dummy points (yellow dots in first and second empty regions), see Algorithm 1, lines 18-24. After generating the dummy points for all i at f_t, the trajectory data is forwarded to the next step, direct neighbor identification. Fig. 4c shows a crowd with dummy points in a single f_t.

The third step, direct neighbor identification, employs a combination of Voronoi Diagram [32] and Convex Hull [33] to find $\mathcal{N}_i(f_t)$ from the input trajectory data with dummy points. A Voronoi Diagram is a method for partitioning a plane into several polygonal regions (named Voronoi cells $\mathcal{V}$s) based on a set of objects/points (called sites) [32]. Each $\mathcal{V}$ contains edges and vertices, which form its boundary. Fig. 5a depicts an example of a Voronoi Diagram for 51 $\mathcal{V}s$ of 51 sites, where black and yellow dots denote the sites. In the same figure, the set of sites contains $\langle x, y\rangle_i$ (dummy points are included) at a specific f_t, then each $\mathcal{V}_i$ includes only one site $\langle x, y\rangle_i$, and all points within $\mathcal{V}_i$ are closer to site $\langle x, y\rangle_i$ than any other

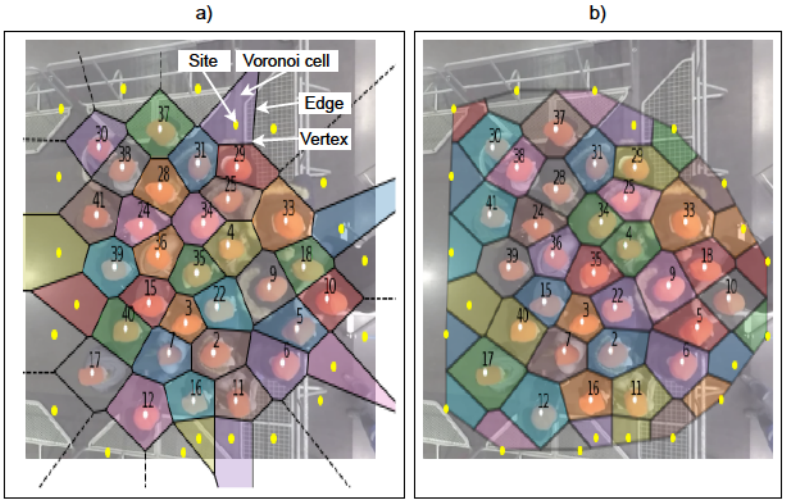

Figure 5: a) Example of a simple Voronoi decomposition. b) Example of bounded Voronoi decomposition. Both are constructed using 30 pedestrian points and 21 dummy points.

sites $\langle x, y \rangle_q$. Where $q \in$ all i at that f_t, and $q \neq i$.

Furthermore, $\mathcal{V}_i$ and $\mathcal{V}_q$ at f_t are considered adjacent if they share at least one edge or two vertices. For instance, as seen in Fig. 5, $\mathcal{V}_4$ and $\mathcal{V}_{34}$ are adjacent, while $\mathcal{V}_4$ and $\mathcal{V}_3$ are not adjacent. Since the Voronoi Diagram contains unbounded cells, determining the adjacent cells for each $\mathcal{V}_i$ at f_t may yield inaccurate results. For instance, most cells of yellow points, which are located at the scene's borders, are unbounded cells, as depicted in Fig. 5a. For further clarity, $\mathcal{V}_i(f_t)$ becomes unbounded when i is a vertex of the convex hull that includes all instances of i at f_t. As a result, the Voronoi Diagram may not provide accurate results when determining adjacent cells, which is a crucial factor in identifying $\mathcal{N}_i(f_t)$. To overcome such limitation, Convex Hull [33] is utilized to finite the Voronoi Diagram (unbounded cells) as shown in Fig. 5b. The Convex Hull is the minimum convex shape that encompasses a given set of points, forming a polygon that connects the outermost points of the set while ensuring that all internal angles are less than 180° [62]. For this purpose, the intersection of each $\mathcal{V}_i(f_t)$ with Convex Hull of all i at f_t is calculated, then the $\mathcal{V}_i(f_t)$ in the diagram are updated based on the intersections to obtain the bounded Voronoi Diagram of all i at f_t (Algorithm 2, lines 5-12). In more details, the Convex Hull of all i at f_t is measured (Algorithm 2, line 8). After that, the intersection between $\mathcal{V}_i(f_t)$ and the Convex Hull at f_t is computed. And finally, we update the Voronoi Diagram at each f_t using the calculated interactions to obtain the corresponding bounded one as shown in Fig. 5b (Algorithm 2, lines 8-11). After creating the bounded Voronoi Diagram, individuals in the direct adjacent Voronoi cells of $\mathcal{V}_i(f_t)$ are $\mathcal{N}_i(f_t)$, (Algorithm 2, lines 12-20). For example, in Fig. 5b, direct adjacent Voronoi cells of $\mathcal{V}_3$ at f_t are $\{\mathcal{V}_2, \mathcal{V}_{22}, \mathcal{V}_{35}, \mathcal{V}_{15}, \mathcal{V}_7\}$, and $\mathcal{N}_3 = \{2, 22, 35, 15, 7\}$.

The last step, local region extraction, aims to extract the local region of each i at f_t, where $i \notin$ dummy points. The step firstly finds $\mathcal{L}_i(f_t)$ based on each $\langle x, y \rangle_j$, where $j \in \mathcal{N}_i(f_t)$, Fig. 4c. Then, $\mathcal{L}_i(f_t)$ are cropped from corresponding f_t and passed to the next module, which will be discussed in the next section. Fig. 3b displays examples of cropped local regions.

Algorithm 2 Pseudo code of direct neighbor identification step

Inputs:

tr_dummy=$\{row_1, row_2, row_3, \dots\}$,

where each row =[dummy point or pedestrian Id, frame order in the corresponding video, x-coordinate point, y- coordinate]

Outputs:

$direct\_neighbor$=$\{row_1, row_2, row_3, \dots\}$,

where each row =[frame order in the corresponding video, pedestrian Id, [direct neighbors (Ids) of pedestrian Id]]

```
file ← open(tr_dummy)
file_dn ← open(direct_neighbor)
data ← load(file)
frames ← unique(data[:, 0])
while fr in frames do
    data_fr ← filter(data , fr)
    vor_diagram ← Voronoi(data_fr[:, 2 : 4])
    CH ← ConvexHull(data_fr[:, 2 : 4])
    for each region ∈ vor_diagram.regions do
        vor_diagram.region ← vor_diagram.region ∩ CH
    end for
    cells ← vor_diagram.regions
    while i in data_fr[:, 0] do
        cell ← cells[i]
        dn_cells ← find_direct_neighbor_cells (cell)
        dn_i ← dn_cells.sites
        rec ← [fr, i, dn_i]
        write (rec) to file_dn
    end while
end while
file.close()
file_dn.close()
```

3.1.2 EfficientNetV1B0-based Deep Feature Extraction

To extract the deep features from each individual's behavior, the feature extraction part of EfficientNetV1B0 is used over their local regions $\mathcal{L}_i(f_t)$. EfficientNetV1B0 is a CNN architecture that has gained popularity for various computer vision tasks due to its efficient use of resources and fewer parameters than other state-of-the-art models [34]. Furthermore, it has achieved high accuracy on multiple image classification datasets. Additionally, the experiments in this article (Section 5.2) indicate that combining EfficientNetV1B0 (without local classification part) with local regions yields the highest accuracy compared to other popular CNN architectures integrated with local regions. Therefore, EfficientNetV1B0's feature extraction part is employed to find more helpful information from each individual's behavior.

The architecture of the efficientNetV1B0-based deep feature extraction model is depicted in Fig. 2. Firstly, it applies a 3×3 convolution operation to the input image, a local region with dimensions of $224 \times 224 \times 3$. Following this, 16 mobile inverted bottleneck convolution (MBConv) blocks [63] are employed to extract deep features (feature maps) $\in \mathbb{R}^{7\times7\times320}$ from each $\mathcal{L}_i(f_t)$. In more detail, the MBConv blocks used consist of one MBConv1, 3×3, six MBConv6, 3×3, and nine MBConv6, 5×5. Fig. 6 illustrates the structure of the MBConv block, which employs a 1×1 convolution operation to expand the depth of feature maps and capture more information. A 3×3 depthwise convolution follows this to decrease the computational complexity and number of parameters. Additionally, batch normalization and swish activation [64] are applied after each convolution

operation. The MBConv then employs a Squeeze-and-Excitation block [65] to enhance the architecture's representation power. The Squeeze-and-Excitation block initially performs global average pooling to reduce the channel dimension. Then it applies an excitation operation with Swish [64] and Sigmoid [66] activations to learn channel-wise attention weights. These weights represent the significance of each feature map and are multiplied by the original feature maps to generate the output feature maps. After the Squeeze-and-Excitation block, another 1×1 convolution with batch normalization is used to reduce the output feature maps' dimensionality, resulting in the final output of the MBConv block.

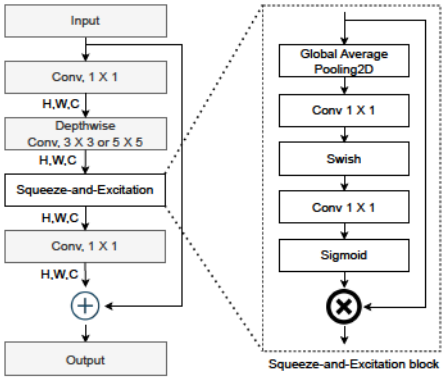

Figure 6: The architecture of MBConv block.

The main difference between MBConv6 and MBConv1 is the depth of the block and the number of operations performed in each block; MBConv6 is six times that of MBConv1. Note that MBConv6, 5×5 performs the identical operations as MBConv6, 3×3, but MBConv6, 5×5 applies a kernel size of 5×5, while MBConv6, 3×3 uses a kernel size of 3×3.

3.2 Detection Component

The primary objective of the detection component is to analyze the feature maps generated by the previous component and categorize individual behaviors as either pushing or non-pushing. This task requires a binary classification followed by an annotation process. Unfortunately, the classification part of the original EfficientNetV1B0 architecture is not designed for binary tasks and is thus unsuitable for identifying pushing behavior. Consequently, we have adapted the classification part of EfficientNetV1B0 to support binary classification. In addition to extracting deep features in the preceding component, this modification allows the EfficientNetV1B0 to classify individual behaviors as pushing or non-pushing in the detection component. To perform the binary classification task, we combine a 1×1 convolution operation, 2D global average pooling, a fully connected layer with a single neuron, and a Sigmoid activation function. Fig. 2 shows the classifier. To gain more information by increasing the number of channels in feature maps, the 1×1 convolutional operation is used. The new dimension of feature maps for each $\mathcal{L}_i(f_t)$ is $7 \times 7 \times 1280$. After that, the global average pooling2D is utilized to transform the feature maps to $1 \times 1 \times 1280$ and feed them to the fully connected layer with one neuron, which produces an output $x \in \mathbb{R}$. Subsequently, the Sigmoid function σ is applied to x, transforming it into a value between 0

and 1. The sigmoid function, commonly used in binary classification, is defined as Eq. (2):

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2}$$

where $\sigma(x)$ represents the probability of the pushing label for the corresponding i at f_t, and e denotes the mathematical constant known as Euler's number. Finally, the classifier uses a threshold value to identify the class of i at f_t as Eq. (3):

$$Class(i, f_t) = \begin{cases} \text{pushing} & \text{if } \sigma(x) \geq threshold \\ \text{non-pushing} & \text{if } \sigma(x) < threshold \end{cases} \tag{3}$$

By default, the threshold value for binary classification is configured to 0.5, which is suitable for datasets exhibiting a balanced distribution. Unfortunately, the new pushing dataset created in Section 4.1 for training and evaluating the proposed framework is imbalanced. As such, using the default threshold may lead to poor performance of the introduced trained classifier on that dataset [67]. Therefore, fine-tuning the threshold in the trained classifier becomes essential for improving accuracy across both pushing and non-pushing classes. The methodology for finding the optimal threshold for the classifier will be explained in detail in Section 4.3. Following training and adjusting the classifier's threshold, it can categorize individuals i as pushing or non-pushing. At the same time, the annotation process draws a red rectangle around the head of each pushing person in the corresponding frames f_t (see Fig. 7) and finally generates an annotated video.

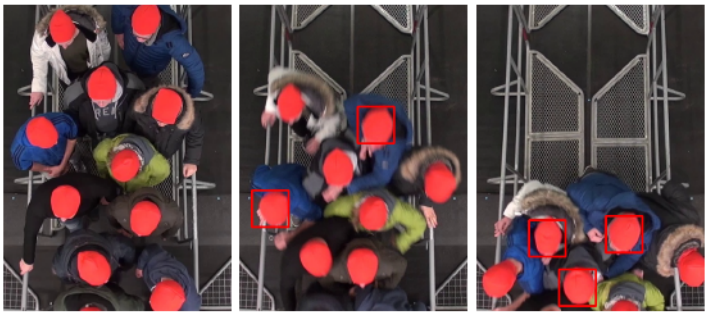

Figure 7: A visualized example of a sequence of frames annotated with red rectangles highlighting the individuals participating in pushing. The original frames were taken from [35].

The following section will discuss the training and evaluating processes of the proposed framework.

4 Training and Evaluation Metrics

This section introduces a novel labeled dataset, as well as presents the parameter setups for the training process, evaluation metrics, and the methodology for improving the framework's performance on an imbalanced dataset.

4.1 A Novel Dataset Preparation

Here, it is aimed at creating the labeled dataset for training and evaluating the proposed framework. The dataset consists of a training set, a validation set for

the learning process, and two test sets for the evaluation process. These sets comprise $\mathcal{L}_i(f_t)$ labeled as pushing or non-pushing. In this context, each pushing $\mathcal{L}_i(f_t)$ means i at f_t contributes pushing, while every non-pushing $\mathcal{L}_i(f_t)$ indicates that i at f_t follows the social norm of queuing. The following will discuss the data sources and methodology used to prepare the sets.

The dataset preparation is based on three data sources: 1) Six videos of real-world experiments of crowded event entrances. 2) Pedestrian trajectory data. 3) Ground truths for pushing behavior. Six video recordings of experiments with their corresponding pedestrian trajectory data are selected from the data archive hosted by Forschungszentrum Jülich [35, 68]. This data is licensed under CC Attribution 4.0 International license. The experimental situations mimic the crowded event entrances, and static top-view cameras were employed to record the experiments with a frame rate of 25 frames per second. For more clarity, Fig. 8 shows overhead views of exemplary experiments, and Table 1 summarizes the various characteristics of the chosen experiments. Additionally, ground truth labels constructed by the manual rating system [10] are used for the last data source. In this system, social psychologists observe and analyze video experiments frame-by-frame to manually identify individuals who are pushing over time. The experts use PeTrack software [69] to manage the manual tracking process and generate the annotations as a text file. For further details on the manual system, readers can refer to Ref. [10].

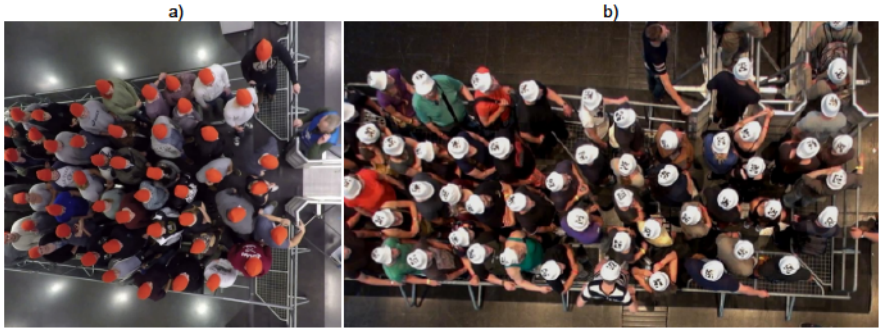

Figure 8: Overhead view of exemplary experiments. a) Experiment 270, as well as Experiments 50, 110, 150, and 280 used the same setup but with different widths of the entrance area ranging from 1.2 to 5.6 m based on the experiment [35]. b) Experiment entrance_2 [68] The entrance gate's width is 0.5 m in all setups.

Table 1: Characteristics of the chosen experiments.

Video experiment *	Width (m)	Pedestrian	Number of gates	Duration (s)	Resolution
50	4.5	42	1	37	1920 × 1440
110	1.2	63	1	53	1920 × 1440
150	5.6	57	1	57	1920 × 1440
270	3.4	67	1	59	1920 × 1440
280	3.4	67	1	67	1920 × 1440
Entrance_2	3.4	123	2	125	1920 × 1080

*The same names as reported in [35, 68]. m stands for meter, and s refers to second.

Here, the methodology used for papering the dataset is described. As shown in Fig. 9, it consists of two phases: local region extraction; and local region

labeling and set generation. The first phase aims to extract local regions (samples) from videos while avoiding duplicates. To accomplish this, the phase initially extracts frames from the input videos second by second. It employs After that the Voronoi-based local region extraction module to identify and crop the samples from the extracted frames based on the trajectory data. Table 2 demonstrates the number of extracted samples from each video, and Fig. 3b shows several examples of local regions. Preventing the presence of duplicate samples between the training, validation, and test sets is crucial to obtain a reliable evaluation for the model. Therefore, this phase removes similar and slightly different samples before proceeding to the next phase. It involves using a pre-trained MobileNet CNN model to extract deep features/embeddings from the samples and cosine similarity to find duplicate or near duplicate samples based on their features [70]. This technique is more robust than comparing pixel values, which can be sensitive to noise and lighting variations [71]. Table 2 depicts the number of removed duplicate samples.

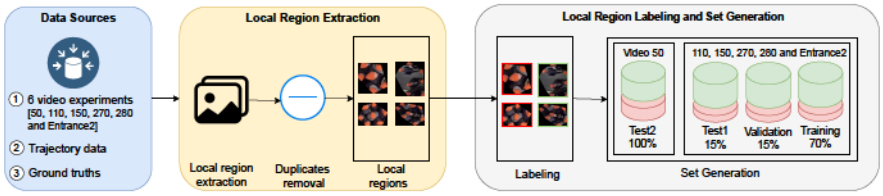

Figure 9: Pipeline of dataset preparation. In the part 'Local Region Labeling and Set Generation', red refers to the pushing class and pushing sample, while the non-pushing class and non-pushing sample are represented in green. The local region extraction component uses trajectory data to determine the coordinates of each person.

Table 2: Summary of the prepared sets.

	Number of samples			Labeled dataset		Training set		Validation set		Test set1		Test set2	
Video	Original	Deleted	Distinct	P	NP	P	NP	P	NP	P	NP	P	NP
110	1046	1	1045	548	497	365	331	99	84	84	82		
150	1469	70	1399	625	774	455	547	83	113	87	114		
270	1627	11	1616	577	1039	401	727	84	161	92	151		
280	1822	44	1778	287	1491	213	1104	44	181	30	206		
Entrance_2	6204	325	5879	1030	4849	726	3403	156	715	148	731		
Total	12168	451	11717	3067	8650	2160	6112	466	1254	441	1284		
50*				317	344							317	344

* Video 50 is used exclusively for the evaluation process, while the remaining video experiments will be employed for both training and evaluation.

On the other hand, the local region and set generation phase is responsible for labeling the extracted samples and producing the sets, including one training set, one validation set, and two test sets. This phase utilizes the ground truth label of each i at f_t to label the samples ($\mathcal{L}_i(f_t)$). If i at f_t contributing to pushing, $\mathcal{L}_i(f_t)$ is categorized as pushing; otherwise, it is classified as non-pushing. Examples of pushing samples can be found in Fig. 3b. The generated labeled dataset from all video experiments comprises 3384 pushing samples and 8994 non-pushing samples. The number of extracted pushing and non-pushing samples from each video is illustrated in Table 2. After creating the labeled dataset, the sets are generated from the dataset. Specifically, the second phase randomly divides the extracted frames from video experiments 110, 150, 270, 280, and Entrance_2 into three sets: 70 %, 15 %, and 15 % for training, validation, and test sets, respectively. Then, using these sets, it generates the training, validation, and test (test set 1) sets from

the labeled corresponding samples ($\mathcal{L}_i(f_t)$). Another test set (test set 2) is also developed from the labeled samples extracted from the complete video experiment 50. Table 2 shows the summary of the generated sets.

To summarize, four labeled sets were created: the training set, which consists of 2160 pushing samples and 6112 non-pushing samples; the validation set, which contains 466 pushing samples and 1254 non-pushing samples; the test set 1, which includes 441 pushing samples and 1284 non-pushing samples; and the test set 2, comprising 317 pushing samples and 344 non-pushing samples. It's crucial to emphasize two key aspects where the new dataset significantly deviates from the datasets outlined in References [29] and [31]. First, the samples in the introduced dataset capture the crowd's appearance surrounding each pedestrian i ($\mathcal{L}_i(f_t)$). This contrasts with samples from previous datasets, which represent the visual motion information within a region ranging from 1.2 to 2.4 square meters on the ground. While the previous datasets were focused on analyzing behavior at a patch level, the new dataset is better suited for studying pushing behavior at an individual level. Fig. 1b and c provide examples of motion information maps and local region $\mathcal{L}_i$ samples, respectively. Second, the new dataset is significantly larger, containing 12,378 samples, compared to 3,780 and 3,941 samples in the datasets reported in References [29] and [31], respectively. In other words, considering the local region around a person also results in a larger dataset than the patch approach.

4.2 Parameter Setup

Table 3 shows parameters used during the training process. The default values for the learning rate and batch size that are typically used in training CNN architectures on ImageNet in Keras were selected. Additionally, other parameters were fine-tuned through experimentation to achieve optimal performance with the new dataset. To prevent overfitting, the training was halted if the validation accuracy did not improve after 20 epochs.

Table 3: The hyperparameter values used in the training process.

Parameter	Value
Optimizer	Adam
Loss function	Binary cross-entropy
Learning rate	0.001
Batch size	32
Epoch	100

4.3 Evaluation Metrics and Performance Improvement

This section will discuss the metrics chosen for evaluating the performance of the proposed framework. Additionally, it will explore the methodology employed to enhance the performance of the trained imbalanced classifier, thereby improving the overall effectiveness of the framework.

Given the imbalanced distribution of the generated local region dataset, the framework exhibits a bias towards the majority class (non-pushing). Consequently, it becomes crucial to employ appropriate metrics for evaluating the performance of the imbalanced classifier. As a result, a combination of metrics was adopted, including macro accuracy, True Pushing Rate (TPR), True Non-Pushing Rate (TNPR), and Area Under the receiver operating characteristic Curve (AUC)

on both test set 1 and test set 2. The following provides a detailed explanation of these metrics.

TPR, also known as sensitivity, is the ratio of correctly classified pushing samples to all pushing samples, and it is defined as:

$$TPR = \frac{TP}{TP + FNP}, \tag{4}$$

where TP and FNP denote correctly classified pushing persons and incorrectly predicted non-pushing persons.

TNPR, also known as specificity, is the ratio of correctly classified non-pushing samples to all non-pushing samples, and it is described as:

$$TNPR = \frac{TNP}{TNP + FP}, \tag{5}$$

where TNP and FP stand for correctly classified non-pushing persons and incorrectly predicted pushing persons.

Macro accuracy, or balanced accuracy, is the average proportion of correct predictions for each class individually. This metric ensures that each class is given equal significance, irrespective of its size or distribution within the dataset. For more clarity, it is just the average of TPR and TNPR as:

$$Macro\,accuracy = \frac{TPR + TNPR}{2}. \tag{6}$$

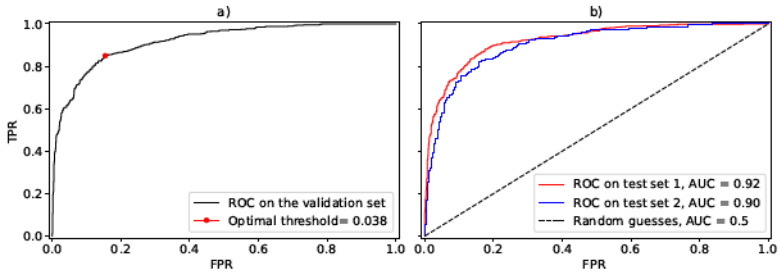

Figure 10: ROC curves for the introduced framework. a) ROC curve with an optimal threshold on the validation set. b) ROC curves with AUC values on test set 1 and test set 2. TPR stands for true pushing rate, while FPR refers to false pushing rate.

AUC is a metric that represents the area under the Receiver Operating Characteristics (ROC) curve. The ROC curve illustrates the performance of a classification model across various threshold values. It plots the false positive rate (FPR) on the horizontal axis against the true positive rate (TPR) on the vertical axis. AUC values range from 0 to 1, where a perfect model achieves an AUC of 1, while a value of 0.5 indicates that the model performs no better than random guessing [72]. Fig. 10a shows an example of a ROC curve with AUC value.

As mentioned above, the binary classifier employs a threshold to convert the calculated probability into a predicted class. The pushing class is predicted if the probability exceeds the threshold; otherwise, the non-pushing label is predicted. The default threshold is typically set at 0.5. However, this value leads to poor performance of the introduced framework because EfficientNetV1B0 and classification were trained on imbalanced dataset [67]. In other words, the default threshold yields a high TNPR and a low TPR in the framework. To address the

imbalance issue and enhance the framework's performance, it becomes necessary to determine an optimal threshold that achieves a better balance between TPR and FPR (1-TNPR). To accomplish this, the ROC curve is utilized over the validation set to identify the threshold value that maximizes TPR and minimizes FPR. Firstly, TPR and TNPR are calculated for several thresholds ranging from 0 to 1. Then, the threshold that yields the minimum value for the following objective function (Eq. (7) is considered the optimal threshold:

$$Objective\ function = |TPR - TNPR|. \tag{7}$$

As shown in Fig. 10a, the red point refers to the optimal threshold of the classifier used in the proposed framework, which is 0.038.

5 Experimental Results and Discussion

Here, several experiments were conducted to evaluate the performance of the proposed framework. Initially, the performance of the proposed framework itself is assessed. Subsequently, It is compared with five other CNN-based frameworks. After that, two customized CNN architectures in the abnormal behavior detection field were used for further evaluation of the proposed framework. The influence of the deep feature extraction module on the proposed framework's performance is also investigated. Subsequently, the manuscript explores the impact of dummy points on the framework's performance. This is followed by a comparison with two state-of-the-art approaches for pushing behavior detection. All experiments and implementations were performed on Google Colaboratory Pro, utilizing Python 3 programming language with Keras, TensorFlow 2.0, and OpenCV libraries. In Google Colaboratory Pro, the hardware setup comprises an NVIDIA GPU with a 15 GB capacity and a system RAM of 12.7 GB. Moreover, the framework and all the baselines developed for comparison in the experiments were trained using the same sets (Table 2) and hyperparameter values (Table 3).

5.1 Performance of the Proposed Framework

The performance of the proposed framework was evaluated using the generated dataset (Table 2) and various metrics, including macro accuracy, TPR, TNPR, and AUC. We first trained the proposed framework's EfficientNetB0-based deep feature extraction module and detection component on the training and validation sets. Subsequently, the framework's performance on test set 1 and test set 2 were assessed.

Table 4: Performance of the proposed framework on both test sets.

	Test set 1 (%)				Test set 2 (%)			
Threshold	Macro accuracy	TNPR	TPR	(TPR-TNPR)	Macro accuracy	TNPR	TPR	(TPR-TNPR)
Default: 0.5	83	92	74	18	82	88	76	12
Optimal: 0.038	**85**	84	86	**2**	82	81	83	2

TNPR and TPR are true non-pushing rate and true pushing rate, respectively.

Table 4 shows that the introduced framework, with the default threshold, obtained macro accuracy of 83 %, TPR of 74 %, and TNPR of 92 % on test set 1. On the other hand, it achieved 82 % macro accuracy, 88 % TNPR, and 76 % TPR on test set 2. However, it is clear that the TPR is significantly lower than the TNPR on both test sets, see Fig. 11a and c. To balance the TPR and TNPR

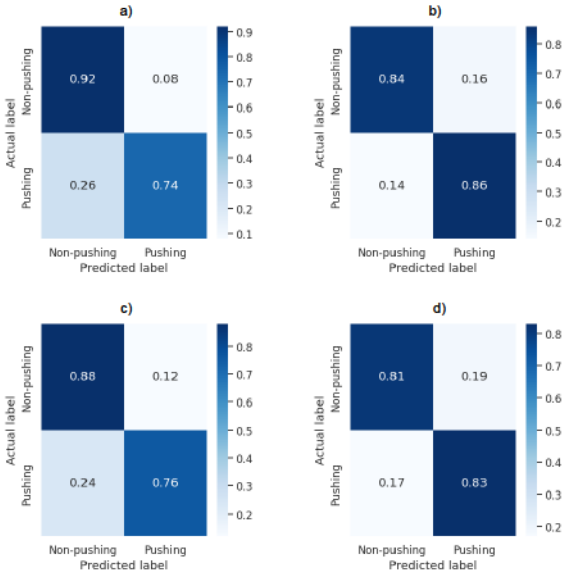

Figure 11: Confusion matrix of the proposed framework on a) Test set 1 with default threshold. b) Test set 1 with the optimal threshold. c) Test set 2 with default threshold. d) Test set 2 with the optimal threshold.

and improve the TPR, the optimal threshold is 0.038, as shown in Fig. 10a. This threshold increases TPR by 12 % and 7 % on test set 1 and test set 2, respectively, without affecting the accuracy, see Fig. 11b and d. In fact, the framework's accuracy improved by 2 % on test set 1. The ROC curves with AUC values for the framework on the two test sets are shown in Fig. 10b, with AUC values of 0.92 and 0.9 on test set 1 and test set 2, respectively.

To summarize, with the optimal threshold, the proposed framework achieved an accuracy of 85 %, TPR of 86 %, and TNPR of 84 % on test set 1, while obtaining 82 % accuracy, 81 % TPR, and 83 % TNPR on test set 2. The next section will compare the framework's performance with five baseline systems for further evaluation.

5.2 Comparison with Five Baseline Frameworks based on Popular CNN Architectures

In this section, the results of further empirical comparisons are shown to evaluate the framework's performance against five baseline systems. Specifically, it explores the impact of the EfficientNetV1B0-based deep feature extraction module on the overall performance of the framework. To achieve this, EfficientNetV1B0 in the deep feature extraction module of the proposed framework was replaced with other CNN architectures, including EfficientNetV2B0 [73] (baseline 1), Xception [74] (baseline 2), DenseNet121 [75] (baseline 3), ResNet50 [76](baseline 4), and MobileNet [77] (baseline 5). To ensure fair comparisons, the five baselines were trained and evaluated using the same sets, hyperparameters, and metrics as those used for the proposed framework.

Before delving into the comparison of the results, it is essential to know that CNN models renowned for their performance on some datasets may perform poorly on others [78]. This discrepancy becomes more apparent when datasets differ in

several aspects, such as size, clarity of relevant features among classes, or overall data quality. Powerful models can be prone to overfitting issues, while simpler models may struggle to capture relevant features in complex datasets with intricate patterns and relationships. Therefore, it's crucial to carefully select or develop an appropriate CNN architecture for a specific issue. For instance, EfficientNetV2B0 demonstrates superior performance compared to EfficientNetV1B0 across various classification tasks [73], including the ImageNet dataset. Moreover, it surpasses the previous version in identifying regions that exhibit pushing persons in motion information maps of crowds [29, 31]. These remarkable outcomes can be attributed to the efficient blocks employed for feature extraction, namely the Mobile Inverted Residual Bottleneck Convolution [63] and Fused Mobile Inverted Residual Bottleneck Convolution [79]. Nevertheless, it should be noted that the presence of these efficient blocks does not guarantee the best performance in identifying pushing individuals based on local regions within the framework. Hence, in this section, the impact of six of the most popular and efficient CNN architectures on the performance of the proposed framework was empirically studied. For clarity, EfficientNetV1B0 was used within the framework, while the remaining CNN architectures were employed in the baselines.

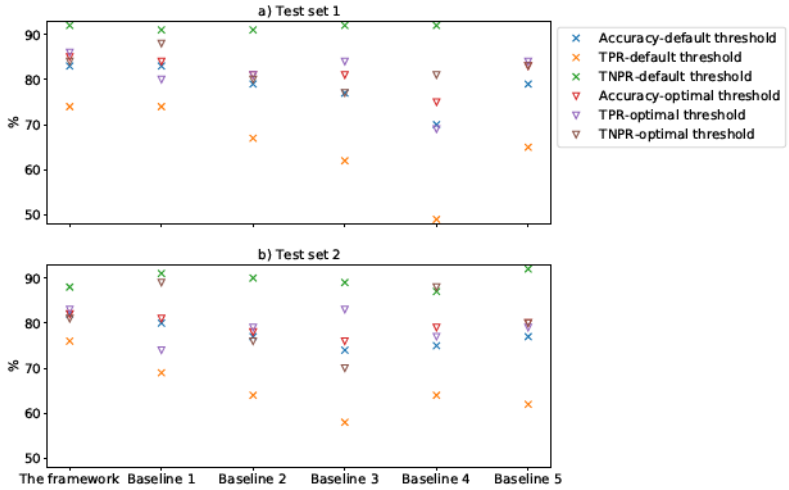

Figure 12: Comparison between the framework (based on EfficientNetV1B0) with the baseline frameworks based on other popular CNN architectures.

The performance results of the proposed framework, as well as the baselines, are presented in Table 5 and visualized in Fig. 12. The findings indicate that EfficientNetV1B0 with optimal threshold leads the framework to achieve superior macro accuracy and AUC with balanced TPR and TNPR compared to CNNs used in baselines 1-5. This can be attributed to the architecture of EfficientNetV1B0, which primarily relies on the Mobile Inverted Residual Bottleneck Convolution with relatively few parameters. As such, the architectural design proves to be particularly suited for the generated dataset focusing on local regions. The visualization in Fig. 13 shows the optimal threshold values for the baselines. These thresholds, as shown in Table 5 and Fig. 12, mostly improved the macro accuracy, TPR, and balanced TPR and TNPR in the baselines. For example, baseline 1 with optimal threshold achieved 84 % macro accuracy, roughly similar to the proposed framework. However, it fell short of achieving a balanced TPR and TNPR along

Table 5: Comparative analysis of the developed framework and the five CNN-based frameworks.

		Test set 1 (%)				Test set 2 (%)			
Framework	Threshold	M. acc.	TNPR	TPR	| TPR-TNPR |	M. acc.	TNPR	TPR	| TPR-TNPR |
The framework	Default: 0.5	83	92	74	18	82	88	76	12
	Optimal: 0.038	**85**	84	86	**2**	**82**	81	83	**2**
Baseline 1	Default: 0.5	83	91	74	17	80	91	69	22
	Optimal: 0.167	84	88	80	8	81	89	74	15
Baseline 2	Default: 0.5	79	91	67	24	77	90	64	26
	Optimal: 0.062	81	80	81	1	78	76	79	3
Baseline 3	Default: 0.5	77	92	62	30	74	89	58	31
	Optimal: 0.038	81	77	84	7	76	70	83	13
Baseline 4	Default: 0.5	70	92	49	43	75	87	64	23
	Optimal: 0.024	75	81	69	12	79	88	77	11
Baseline 5	Default: 0.5	79	94	65	29	77	92	62	30
	Optimal: 0.076	83	83	84	1	80	80	79	1

M. acc means macro accuracy. TNPR and TPR are true non-pushing rate and true pushing rate, respectively.

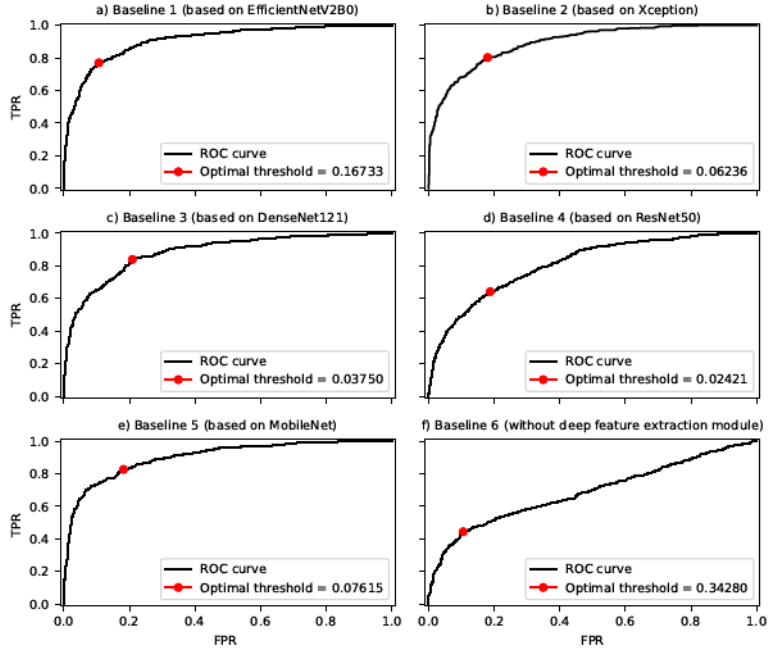

Figure 13: ROC curves with optimal thresholds for the baselines over the validation set. TPR stands for true pushing rate, while FPR refers to false pushing rate. ROC stands for Receiver Operating Characteristics.

with improving TPR on both test sets as effectively as the framework. To provide further clarity, baseline 1 achieved 80 % TPR with 8 % as the difference between TPR and TNPR, whereas the proposed framework attained an 86 % TPR with 2 % as the difference between TPR and TNPR on test set 1. Similarly, on test set 2, the framework achieved 81 % TPR, while baseline 1 achieved a TPR of 74 %.

Compared to other baselines that utilize optimal thresholds on test set 1, the proposed framework outperformed them regarding macro accuracy, TPR, and TNPR. Similarly, on test set 2, the framework surpasses all baselines except for the ResNet50-based baseline (baseline 4). However, it is essential to note that this baseline only achieved better TNPR, whereas the introduced framework excels in macro accuracy and TPR. As a result, the framework emerges as the superior

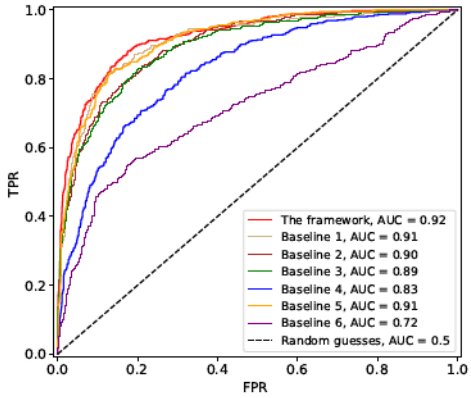

Figure 14: ROC curves with AUC values on the test set 1. Comparison between the introduced framework (based on EfficientNetV1B0) with five baselines based on different CNN architectures, as well as the one baseline without the deep feature extraction module (baseline 6). TPR stands for true pushing rate, while FPR refers to false pushing rate.

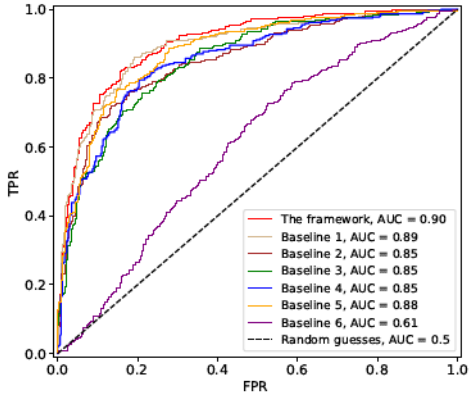

Figure 15: ROC curves with AUC values on the test set 2. Comparison between the framework (based on EfficientNetV1B0) with five baselines based on different CNN architectures, as well as the one baseline without the deep feature extraction module (baseline 6). TPR stands for true pushing rate, while FPR refers to false pushing rate.

choice on test set 2. To alleviate any confusion in the comparison, Fig. 14 shows the ROC curves with AUC values compared to its baselines on test set 1. Likewise, Fig. 15 depicts the same for test set 2. The AUC values show that the proposed framework achieved better performance than the baselines on both test sets. Moreover, they substantiate that EfficientNetV1B0 is the most suitable CNN for extracting deep features from the generated local region samples.

In conclusion, the experiments demonstrate that the proposed framework, utilizing EfficientNetV1B0, achieved the highest performance compared to the baselines relying on other CNN architectures on both test sets. Furthermore, the optimal thresholds in the developed framework and the baselines resulted in a significant improvement in the performance across both test sets.

5.3 Comparison with Customized CNN Architectures in Abnormal Behavior Detection Field

Here, there are two objectives: 1) Evaluating the performance of some existing CNN models developed to detect abnormal human behavior for pushing detection purposes. 2) Further evaluation of the trained binary classifier (EfficientNetB0 with fully connected layer and Sigmoid activation function). The customized architectures are CNN-1 [23] and CNN-2 [41]. The first architecture, CNN-1, employed 75 × 75 pixels as an input image. Furthermore, three convolutional layers, batch normalization, and max pooling operations were used for feature extraction. The developers of this model utilized a fully connected layer with a softmax activation function for classification. The second architecture, CNN-2, downsized the input images to 32×32 pixels before employing three convolutional layers with three max-pooling layers. For classification, it used two fully connected layers, with the first layer based on a ReLU activation function and the second layer employing a softmax activation function.

Table 6: Comparison to CNN-1 and CNN-2.

		Test set 1 (%)			Test set 2 (%)		
Framework	Optimal threshold	Macro accuracy	TNPR	TPR	Macro accuracy	TNPR	TPR
The framework	0.038	85	84	86	82	81	83
CNN-1	0.23	73	71	75	64	40	88
CNN-2	0.0076	71	71	71	75	66	85

TNPR and TPR are true non-pushing rate and true pushing rate, respectively.

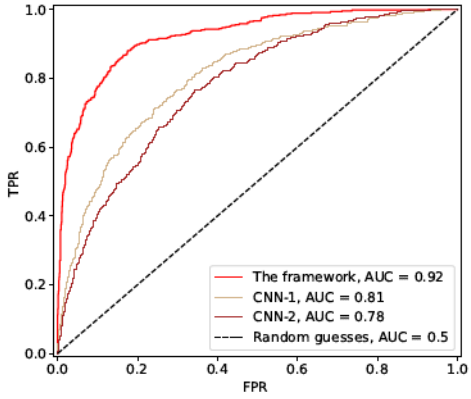

Figure 16: ROC curves of our classifier and the two customized CNNs on test set 1.

The results in Table 6 and Fig. 16 demonstrate that our classifier surpassed CNN-1 and CNN-2 by at least 11 % in all metrics, including macro accuracy, TPR, TNPR, and AUC on test set 1. In contrast, on Test Set 2, the developed classifier achieved a macro accuracy of 82 %, significantly outperforming CNN-1 and CNN-2, which attained 40 % and 66 %, respectively. Furthermore, the two customized CNN architectures exhibited high False Pushing Rates (FPR) of over 34 %, while the proposed classifier's FPR was 19 %. Additionally, as shown in Fig. 17, the AUC of our classifier stands at 90 %, in comparison to CNN-1 and CNN-2, which achieved 74 % and 83 %, respectively.

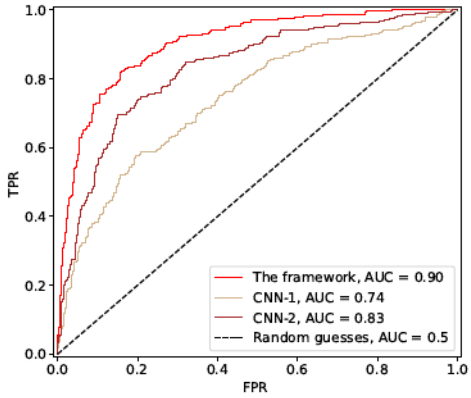

Figure 17: ROC curves of our classifier and the two customized CNNs on test set 2.

To sum up, the proposed binary classifier has demonstrated significant superiority over CNN-1 and CNN-2 across both test sets. Given the high complexity of pushing detection in crowded environments, the simple architectures of CNN-1 and CNN-2 fall short of effectively identifying pushing MIM-patches.

5.4 Impact of Deep Feature Extraction Module

This section aims to investigate how the deep feature extraction module affects the framework's performance. For this purpose, a new baseline (baseline 6) is developed, incorporating a Voronoi-based local region extraction module and detection component. In other words, the deep feature extraction module is removed from the proposed framework to construct this baseline.

Table 7: Performance results of the baseline 6.

	Test set 1 (%)			Test set 2 (%)		
Threshold	Macro accuracy	TNPR	TPR	Macro accuracy	TNPR	TPR
Default: 0.5	59	97	18	58	59	57
Optimal: 0.342	67	91	44	59	38	79

TNPR and TPR are true non-pushing rate and true pushing rate, respectively.

Table 7 demonstrates that the baseline exhibited poor performance, with macro accuracy of 67 % on test set 1 and 59 % on test set 2. Additionally, Fig. 14 and Fig. 15 illustrate AUC values of 72 % on test set 1 and 61 % on test set 2 for baseline 6. Comparing this baseline with the weakest baseline in Table 5, which utilizes ResNet50, it is evident that deep feature extraction leads to macro accuracy improvement of at least 8 % on test set 1 and at least 20 % on test set 2. Similarly, deep feature extraction enhances AUC values by at least 11 % on test set 1 and more than 24 % on test set 2.

In summary, the deep feature extraction module significantly enhances the performance of the framework.

5.5 Impact of Dummy Points

This section aims to evaluate the impact of adding dummy points on the performance of the proposed framework. For this purpose, a new dataset, identical to

the original (Table 2) but without dummy points, was prepared. The new dataset was then used to train and evaluate the framework (named baseline 7). Fig. 18, on the right in a, b, c, and d, displays several examples of local regions generated without using dummy points.

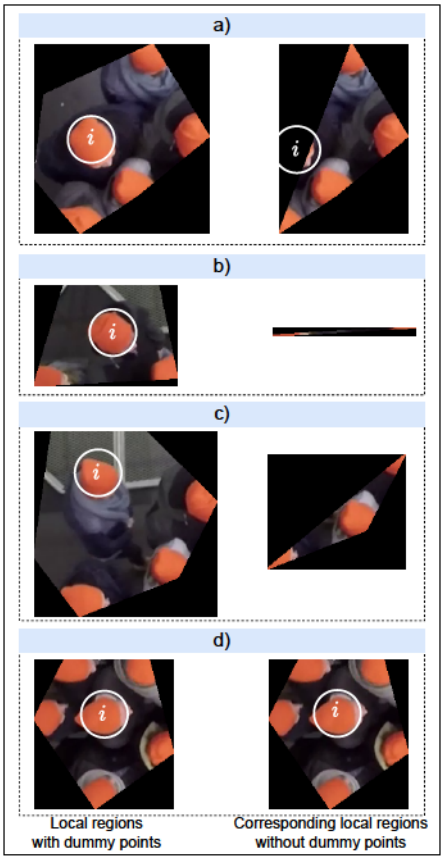

Figure 18: Examples of local regions with dummy points (left samples) and without dummy points (right samples). The white circle represents the target person i in the local region $\mathcal{L}_i$.

Table 8: Comparison to baseline 7.

		Test set 1 (%)			Test set 2 (%)		
Framework	Optimal threshold	Macro accuracy	TNPR	TPR	Macro accuracy	TNPR	TPR
The framework	0.038	85	84	86	82	81	83
Baseline 7	0.030	80	79	81	62	57	66

TNPR and TPR are true non-pushing rate and true pushing rate, respectively.

In the proposed framework, the local region is crucial in assisting it to identify the behavior of each individual i. This is because the local region encompasses the crowd dynamics around i, thereby reflecting potential interactions between i and its neighbors. It is clear that $\mathcal{L}$ of person i located at the borders of crowds when dummy points are not added, fail to encompass the crowd dynamics surrounding i (e.g., samples on the right side in Fig. 18a, b, and c). This leads to losing valuable information about the i's behavior, decreasing the framework's performance. Meanwhile, the examples on the left (Fig. 18a, b, and c) illustrate

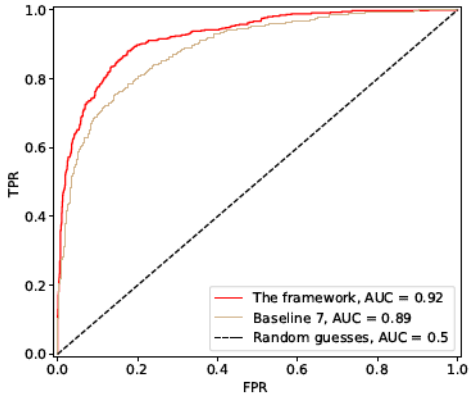

Figure 19: ROC curves of the proposed framework and baseline 7 on test set 1.

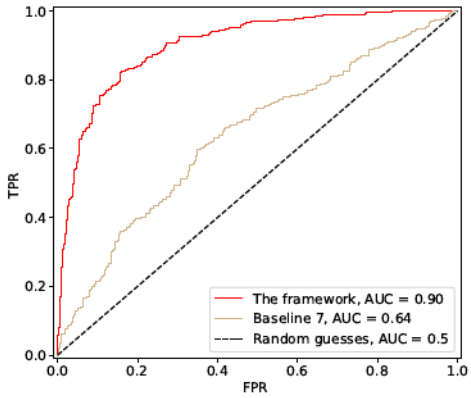

Figure 20: ROC curves of the proposed framework and baseline 7 on test set 2.

how the dummy point technique helps form $\mathcal{L}_i$ that encompasses the space around i. Moreover, Fig. 18d illustrates that the dummy point technique is not applied to i who is surrounded by neighbors in all directions.

As Table 8 illustrates, incorporating dummy points enhanced the performance of the proposed framework on both test sets, improving the macro accuracy, TPR, and TNPR by 5 %. In contrast, the framework without the dummy points technique (baseline 7) achieved lower performance, with a macro accuracy of 62 %, a TNPR of 57 %, and a TPR of 66 %, compared to the developed framework. Moreover, the dummy point technique increased AUC in the framework by 3 % and 26 % over test set 1 and test set 2, respectively.

5.6 Performance Evaluation Against Existing Pushing Detection Approaches

The primary aim of this section is to assess the performance of the proposed framework by comparing it with existing automatic approaches for detecting pushing behavior in dense crowds.

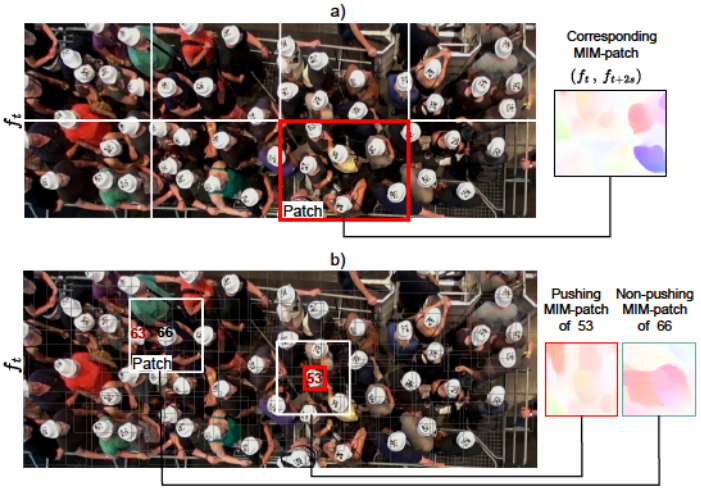

Figure 21: Visual Examples of patches and MIM-patches in related Works [29, 31]: a) A 2×4 patch array with a MIM-patch generated by related works. b) Two examples of square patches (highlighted in white) and their corresponding MIM-patches created by DL4PuDe and Cloud-DL4PuDe. The numbers 53, 63, and 55 denote the IDs of three pedestrians. Moreover, in b), the dark gray color aims to give readers an overview of the square patches created by the enhanced patch identification strategy while ensuring that the original frame remains visible. In the notation f_t, f signifies a frame at timestamp t in seconds s. MIM stands for Motion Information Map.

As mentioned in the literature, two main approaches have been published [29, 31], along with a more concise method referenced in [30], which is part of Ref. [29]. This comparison will focus on the primary approaches detailed in [29] and [31]. It is noteworthy that these approaches employ a patch-based methodology, targeting the identification of patches containing at least one individual engaged in pushing. Each patch typically encompasses an area ranging from 1.2 to 2.55 square meters on the ground. To clarify, both approaches employ a similar strategy for patch identification. They start by extracting Motion Information Maps (MIM) from consecutive frames of video or live streams that capture dense crowds. Every MIM, along with its initial corresponding frame, is then divided into a grid of MIM-patches, arranged into rows and columns as defined by the user. Fig. 21a displays an example with 2×4 patches on the left side and a single MIM-patch on the right side. This patch identification strategy enables the trained classifiers in both approaches to mark pushing patches within the crowd. The red rectangle in Fig. 21a highlights an example of such annotated pushing patches.

Table 9: Comparison to state-of-the-art automatic pushing detection Approaches.

		Test set 1 (%)			Test set 2 (%)		
Framework	Optimal threshold	Macro accuracy	TNPR	TPR	Macro accuracy	TNPR	TPR
The framework	0.038	85	84	86	82	81	83
DL4PuDe	0.023	77	76	78	62	44	80
Cloud-DL4PuDe	0.04	77	75	78	61	48	74

TNPR and TPR are true non-pushing rate and true pushing rate, respectively.

It is evident that current approaches cannot detect individuals who join in pushing, a capability offered by the proposed framework. Consequently, comparing these approaches directly with the developed framework would be unfair, as they serve different purposes. To facilitate a fair and effective, the patch identi-

fication strategy in both existing approaches has been modified to operate at a microscopic level, as opposed to the patch level. The enhanced patch identification strategy employs the input trajectory data to find a square patch for each person (i), with i's position serving as the center of the patch. The dimensions of this patch are approximately 75 cm on the ground. This dimension was chosen after observing various dimensions in video experiments of entrances employed in this work. We found that this particular dimension ensures the patch not only covers individual i but also captures the surrounding crowd dynamics, thereby providing insight into the interactions between individuals i and their direct neighbors($\mathcal{N}_i$).

For instance, as illustrated in Fig. 21b, the patches for individuals numbered 53 and 66 are marked with white squares, along with the corresponding MIM-patches for the areas surrounding individuals 53 and 66. For the sake of clarity and ease of discussion, the first [29] and second [31] approaches with enhanced patch identification strategy shall henceforth be referred to as "DL4PuDe" and "Cloud-DL4PuDe," respectively.

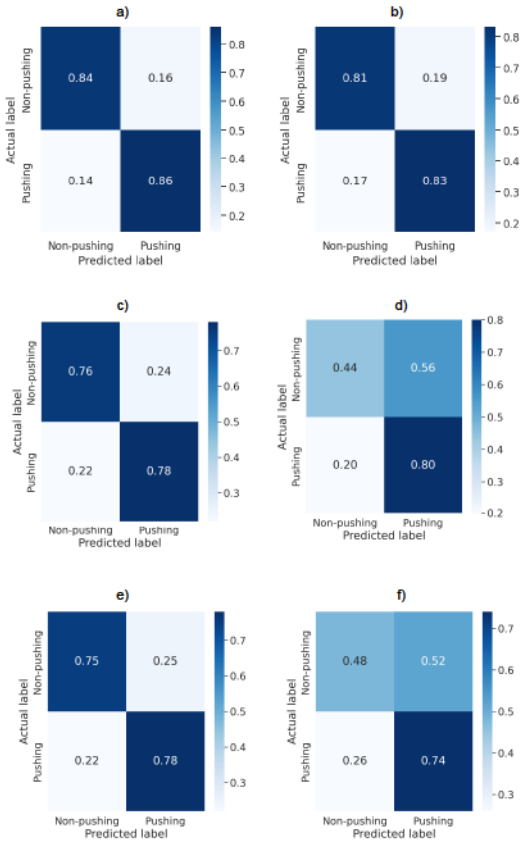

Figure 22: Confusion Matrices for the Proposed Framework, DL4PuDe, and Cloud-DL4PuDe with the optimal threshold: a) Proposed Framework on Test Set 1, b) Proposed Framework on Test Set 2, c) DL4PuDe on Test Set 1, d) DL4PuDe on Test Set 2, e) Cloud-DL4PuDe on Test Set 1, f) Cloud-DL4PuDe on Test Set 2.

To compare Dl4PuDe and Cloud-DL4PuDe approaches with the proposed framework, it is necessary to train and evaluate them using a dataset that includes both pushing and non-pushing square MIM-patches. Furthermore, the setup of param-

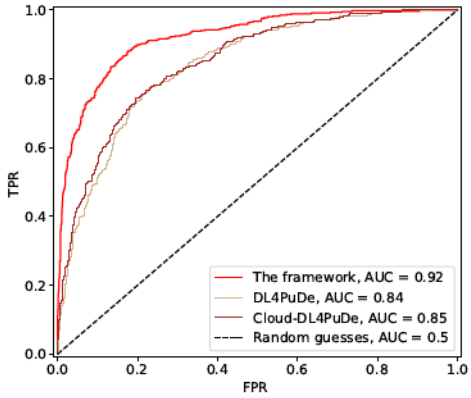

Figure 23: ROC curves and AUC values for Test Set 1: A comparison between the introduced framework, DL4PuDe, and Cloud-DL4PuDe.

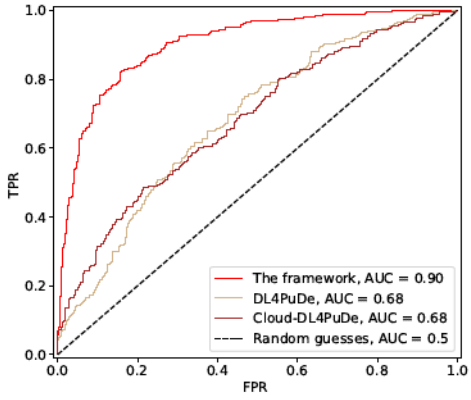

Figure 24: ROC curves and AUC values for Test Set 2: A comparison between the introduced framework, DL4PuDe, and Cloud-DL4PuDe.

eters, as outlined in Table 3, will be used in the training process. For this purpose, we created such a dataset using the same video experiments, trajectory data, and ground truth data employed in preparing the dataset for the proposed framework. Initially, MIMs are extracted from the video experiments using a combination of deep optical flow and color wheel methods, similar to the previous approaches. For more information, we refer the reader to Ref. [31], specifically Section IV.A.2. Next, the improved patch identification strategy is employed to extract patches from those MIMs. Afterward and based on the ground truth data, MIM-patches containing a person i engaged in pushing are labeled as pushing; otherwise, they are labeled non-pushing. For more clarity, in Fig. 21b, the MIM-patch for individual 53 is categorized as pushing due to the pushing behavior of person 53. Conversely, the MIM-patch for individual 66 is non-pushing, given that person 66 does not join in such behavior. Finally, the same splitting technique used to generate the dataset for training and evaluating the proposed framework was also employed to create the new dataset for DL4PuDe and Cloud-DL4PuDe ap-

proaches.

Table 9 displays the comparison results between the proposed framework and the enhanced DL4PuDe and Cloud-DL4PuDe approaches. The results indicate that the proposed framework significantly outperformed both approaches on both test sets. On test set 1, the proposed framework achieved a minimum 8 % improvement in macro accuracy, TPR, and TNPR compared to the related approaches. In contrast, on test set 2, DL4PuDe and Cloud-DL4PuDe exhibited high false positive rates (FPR) with 62 %and 61 % macro accuracy, respectively. Meanwhile, the framework achieved an 82 % macro accuracy on test set 2. Fig. 22 presents the confusion matrices for each approach on both test sets. Moreover, as shown in Fig. 23 and Fig. 24, the framework consistently outperformed the other approaches in terms of the AUC metric. It achieved an improvement of at least 7 % on test set 1 and 22 % on test set 2. These results can be attributed to the fact that square patches may contain both pushing and non-pushing behavior simultaneously. This can lead the CNN classifier to learn irrelevant features from the patches. For more clarity, the patch of person 66 in Fig. 21b is classified as non-pushing because person 66 is not engaged in pushing behavior, even though person 63 within the same patch is involved in pushing.

In summary, our developed framework achieved superior performance, demonstrating improvements of at least 8 % in macro accuracy, TPR, TNPR, and AUC on both test sets compared to the enhanced DL4PuDe and cloud-DL4PuDe approaches. Furthermore, both approaches experience overfitting when attempting to detect pushing behavior at the microscopic level. This serves as evidence that our novel approach, used in our framework to identify the local regions of each person, efficiently assists EficientNetV1B0-based CNN in learning the relevant features for pushing behavior.

6 Conclusion and Future Work

This article introduced a new framework for automatically identifying pushing at the microscopic level within video recordings of crowds. The proposed framework utilizes a novel Voronoi-based method to determine the local region of each person in the input video over time. It further applies EfficientNetV1B0 to extract deep features from these local regions, capturing valuable information about individual behavior. Finally, a fully connected layer with a Sigmoid activation function is employed to analyze the deep features and annotate the pushing persons over time in the input video. To train and evaluate the performance of the framework, a novel dataset was created using six real-world experiments with their trajectory data and corresponding ground truths. The experimental findings demonstrated that the proposed framework surpassed state-of-the-art approaches, as well as seven baseline methods in terms of macro accuracy, true pushing rate, and true non-pushing rate.

The proposed framework has some limitations. First, it was designed to work exclusively with top-view camera video recordings that include trajectory data. Second, it was trained and evaluated based on a limited number of real-world experiments, which may impact its generalizability to a broader range of scenarios. Our future goals include improving the framework in two key areas: 1) Enabling it to detect pushing persons from video recordings without the need for trajectory data as input. 2) Improving its performance in terms of macro accuracy, true

pushing rate, and true non-pushing rate by: a) Enlarging the dataset by utilizing additional videos from real-world experiments. These videos will encompass various scenarios. b) Employing transfer learning and data augmentation techniques. c) Processing a sequence of frames instead of a single frame, to extract more valuable features.

Acknowledgments The authors are thankful to Anna Sieben, Helena Lügering, and Ezel Üsten for the valuable discussions, manual annotation of the pushing behavior in the video of the experiments.

Authors' contributions Conceptualization, A.A.; methodology, A.A., A.S.; software, A.A.; validation, A.A.; formal analysis, A.A.; investigation, A.A.; data curation, A.A.; writing—original draft preparation, A.A.; writing—review and editing, A.A., M.M., M.C. and A.S.; supervision, M.M., M.C. and A.S.; All authors have read and agreed to the published version of the manuscript.

Funding This work was funded by the German Federal Ministry of Education and Research (BMBF: funding number 01DH16027) within the Palestinian-German Science Bridge project framework, and partially by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—491111487.

Availability of data and code All videos and trajectory data used in generating the patch-based dataset were obtained from the data archive hosted by the Forschungszentrum Jülich under CC Attribution 4.0 International license [68, 35]. The implementation of the proposed framework, codes used for building training and evaluating the models, as well as test sets and trained models are publicly available at: `https://github.com/PedestrianDynamics/VCNN4PuDe` or at [80] (accessed on 23 July 2023). The training and validation sets are available from the corresponding authors upon request.

Declarations

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this article.

Ethical approval The experiments used in the dataset were conducted according to the guidelines of the Declaration of Helsinki and approved by the ethics board at the University of Wuppertal, Germany. Informed consent was obtained from all subjects involved in the experiments.

References

[1] S. Feldmann and J. Adrian, "Forward propagation of a push through a row of people," *Safety science*, vol. 164, p. 106173, 2023.

[2] X. Li, X. Xu, J. Zhang, K. Jiang, W. Liu, R. Yi, and W. Song, "Experimental study on the movement characteristics of pedestrians under sudden contact forces," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 6, p. 063406, 2021.

[3] J. Adrian, A. Seyfried, and A. Sieben, "Crowds in front of bottlenecks at entrances from the perspective of physics and social psychology," *Journal of the Royal Society Interface*, vol. 17, no. 165, p. 20190871, 2020.

[4] C. Wang and W. Weng, "Study on the collision dynamics and the transmission pattern between pedestrians along the queue," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2018, no. 7, p. 073406, 2018.

[5] J. P. Keating, "The myth of panic," *Fire Journal*, vol. 76, no. 3, pp. 57–61, 1982.

[6] J. D. Sime, "Affiliative behaviour during escape to building exits," *Journal of environmental psychology*, vol. 3, no. 1, pp. 21–41, 1983.

[7] J. Li, J. Wang, S. Xu, J. Feng, J. Li, Z. Wang, and Y. Wang, "The effect of geometric layout of exit on escape mechanism of crowd," in *Building Simulation*. Springer, 2022, pp. 1–10.

[8] T. Goyal, D. Kahali, and R. Rastogi, "Analysis of pedestrian movements on stairs at metro stations," *Transportation research procedia*, vol. 48, pp. 3786–3801, 2020.

[9] C. Project, "Crowd management in transport infrastructures (project number 13n14530 to 13n14533)," https://www.croma-projekt.de/de, 2018.

[10] E. Üsten, H. Lügering, and A. Sieben, "Pushing and non-pushing forward motion in crowds: A systematic psychological observation method for rating individual behavior in pedestrian dynamics," *Collective Dynamics*, vol. 7, pp. 1–16, 2022.

[11] J. Adrian, M. Boltes, S. Holl, A. Sieben, and A. Seyfried, "Crowding and queuing in entrance scenarios: influence of corridor width in front of bottlenecks," *arXiv preprint arXiv:1810.07424*, 2018.

[12] A. Sieben and A. Seyfried, "Inside a life-threatening crowd: Analysis of the love parade disaster from the perspective of eyewitnesses," *arXiv preprint arXiv:2303.03977*, 2023.

[13] J. F. Kooij, M. C. Liem, J. D. Krijnders, T. C. Andringa, and D. M. Gavrila, "Multi-modal human aggression detection," *Computer Vision and Image Understanding*, vol. 144, pp. 106–120, 2016.

[14] C. Wang, L. Shen, and W. Weng, "Experimental study on individual risk in crowds based on exerted force and human perceptions," *Ergonomics*, vol. 63, no. 7, pp. 789–803, 2020.

[15] C. Project, "Technologies for computer-assisted crowd management, fetopen-01-2018-2019-2020 fetopen challenging current thinking (project number 899739)," https://crowddna.eu/, 2020.

[16] B. project, "Bausteine für die sicherheit von großveranstaltungen (project number 13n12045)," https://www.vfsg.org/basigo-wiki/, 2012.

[17] T. Metivet, L. Pastorello, and P. Peyla, "How to push one's way through a dense crowd," *Europhysics Letters*, vol. 121, no. 5, p. 54003, 2018.

[18] A. Budiman, R. A. Yaputera, S. Achmad, A. Kurniawan *et al.*, "Student attendance with face recognition (lbph or cnn): Systematic literature review," *Procedia Computer Science*, vol. 216, pp. 31–38, 2023.

[19] W. Lu, C. Lan, C. Niu, W. Liu, L. Lyu, Q. Shi, and S. Wang, "A cnn-transformer hybrid model based on cswin transformer for uav image object detection," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2023.

[20] Y. Dong, Z. Jiang, F. Tao, and Z. Fu, "Multiple spatial residual network for object detection," *Complex & Intelligent Systems*, vol. 9, no. 2, pp. 1347–1362, 2023.

[21] C. Ning, L. Menglu, Y. Hao, S. Xueping, and L. Yunhong, "Survey of pedestrian detection with occlusion," *Complex & Intelligent Systems*, vol. 7, pp. 577–587, 2021.

[22] Q. Liu, X. Wang, Y. Wang, and X. Song, "Evolutionary convolutional neural network for image classification based on multi-objective genetic programming with leader–follower mechanism," *Complex & Intelligent Systems*, vol. 9, no. 3, pp. 3211–3228, 2023.

[23] C. Direkoglu, "Abnormal crowd behavior detection using motion information images and convolutional neural networks," *IEEE Access*, vol. 8, pp. 80 408–80 416, 2020.

[24] A. F. Alia and A. Taweel, "Feature selection based on hybrid binary cuckoo search and rough set theory in classification for nominal datasets," *algorithms*, vol. 14, no. 21, p. 65, 2017.

[25] A. Alia and A. Taweel, "Enhanced binary cuckoo search with frequent values and rough set theory for feature selection," *IEEE access*, vol. 9, pp. 119 430–119 453, 2021.

[26] ——, "Hybrid nature inspired algorithms and rough set theory in feature selection for classification: A review," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 3, p. 7, 2016.

[27] H. Gan, C. Xu, W. Hou, J. Guo, K. Liu, and Y. Xue, "Spatiotemporal graph convolutional network for automated detection and analysis of social behaviours among pre-weaning piglets," *Biosystems Engineering*, vol. 217, pp. 102–114, 2022.

[28] H. Gan, M. Ou, E. Huang, C. Xu, S. Li, J. Li, K. Liu, and Y. Xue, "Automated detection and analysis of social behaviors among preweaning piglets using key point-based spatial and temporal features," *Computers and Electronics in Agriculture*, vol. 188, p. 106357, 2021.

[29] A. Alia, M. Maree, and M. Chraibi, "A hybrid deep learning and visualization framework for pushing behavior detection in pedestrian dynamics," *Sensors*, vol. 22, no. 11, p. 4040, 2022.

[30] ——, "A fast hybrid deep neural network model for pushing behavior detection in human crowds," in *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, 2022, pp. 1–2.

[31] A. Alia, M. Maree, M. Chraibi, A. Toma, and A. Seyfried, "A cloud-based deep learning framework for early detection of pushing at crowded event entrances," *IEEE Access*, 2023.

[32] P. J. Green and R. Sibson, "Computing dirichlet tessellations in the plane," *The computer journal*, vol. 21, no. 2, pp. 168–173, 1978.

[33] A. M. Andrew, "Another efficient algorithm for convex hulls in two dimensions," *Information Processing Letters*, vol. 9, no. 5, pp. 216–219, 1979.

[34] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[35] "Crowds in front of bottlenecks from the perspective of physics and social psychology," http://doi.org/10.34735/ped.2018.1, Pedestrian Dynamics Data Archive hosted by the Forschungszentrum Jülich, 2018.

[36] A. Chadha and Y. Andreopoulos, "Voronoi-based compact image descriptors: Efficient region-of-interest retrieval with vlad and deep-learning-based descriptors," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1596–1608, 2017.

[37] G. Bayar and T. Bilir, "Estimation of multiple crack propagation pattern in concrete using voronoi tessellation method," *Sādhanā*, vol. 48, no. 3, p. 165, 2023.

[38] P. Warren, N. Raju, A. Prasad, M. S. Hossain, R. Subramanian, J. Kapat, N. Manjooran, and R. Ghosh, "Grain and grain boundary segmentation using machine learning with real and generated datasets," *Computational Materials Science*, vol. 233, p. 112739, 2024.

[39] D. Moukheiber, S. Mahindre, L. Moukheiber, M. Moukheiber, S. Wang, C. Ma, G. Shih, Y. Peng, and M. Gao, "Few-shot learning geometric ensemble for multi-label classification of chest x-rays," in *MICCAI Workshop on Data Augmentation, Labelling, and Imperfections*. Springer, 2022, pp. 112–122.

[40] W. Shi, J. M. Lemoine, A.-E.-M. A. Shawky, M. Singha, L. Pu, S. Yang, J. Ramanujam, and M. Brylinski, "Bionoinet: ligand-binding site classification with off-the-shelf deep neural network," *Bioinformatics*, vol. 36, no. 10, pp. 3077–3083, 2020.

[41] N. C. Tay, T. Connie, T. S. Ong, K. O. M. Goh, and P. S. Teh, "A robust abnormal behavior detection method using convolutional neural network," in *Computational Science and Technology*. Springer, 2019, pp. 37–47.

[42] E. Duman and O. A. Erdem, "Anomaly detection in videos using optical flow and convolutional autoencoder," *IEEE Access*, vol. 7, pp. 183 914–183 923, 2019.

[43] M. J. Roshtkhari and M. D. Levine, "An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions," *Computer vision and image understanding*, vol. 117, no. 10, pp. 1436–1452, 2013.

[44] G. Singh, A. Khosla, and R. Kapoor, "Crowd escape event detection via pooling features of optical flow for intelligent video surveillance systems," *International Journal of Image, Graphics and Signal Processing*, vol. 10, no. 10, p. 40, 2019.

[45] M. George, C. Bijitha, and B. R. Jose, "Crowd panic detection using autoencoder with non-uniform feature extraction," in *2018 8th International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2018, pp. 11–15.

[46] G. L. Santos, P. T. Endo, K. H. d. C. Monteiro, E. d. S. Rocha, I. Silva, and T. Lynn, "Accelerometer-based human fall detection using convolutional neural networks," *Sensors*, vol. 19, no. 7, p. 1644, 2019.

[47] A. Mehmood, "Lightanomalynet: A lightweight framework for efficient abnormal behavior detection," *Sensors*, vol. 21, no. 24, p. 8501, 2021.

[48] X. Zhang, Q. Zhang, S. Hu, C. Guo, and H. Yu, "Energy level-based abnormal crowd behavior detection," *Sensors*, vol. 18, no. 2, p. 423, 2018.

[49] E. Ekanayake, Y. Lei, and C. Li, "Crowd density level estimation and anomaly detection using multicolumn multistage bilinear convolution attention network (mcms-bcnn-attention)," *Applied Sciences*, vol. 13, no. 1, p. 248, 2022.

[50] I.-C. Hwang and H.-S. Kang, "Anomaly detection based on a 3d convolutional neural network combining convolutional block attention module using merged frames," *Sensors*, vol. 23, no. 23, p. 9616, 2023.

[51] A. Patwal, M. Diwakar, V. Tripathi, and P. Singh, "An investigation of videos for abnormal behavior detection," *Procedia Computer Science*, vol. 218, pp. 2264–2272, 2023.

[52] G. Almahadin, M. Subburaj, M. Hiari, S. Sathasivam Singaram, B. P. Kolla, P. Dadheech, A. D. Vibhute, and S. Sengan, "Enhancing video anomaly detection using spatio-temporal autoencoders and convolutional lstm networks," *SN Computer Science*, vol. 5, no. 1, p. 190, 2024.

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

[54] M. Xu, X. Yu, D. Chen, C. Wu, and Y. Jiang, "An efficient anomaly detection system for crowded scenes using variational autoencoders," *Applied Sciences*, vol. 9, no. 16, p. 3337, 2019.

[55] S. Smeureanu, R. T. Ionescu, M. Popescu, and B. Alexe, "Deep appearance features for abnormal behavior detection in video," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 779–789.

[56] Z. Ilyas, Z. Aziz, T. Qasim, N. Bhatti, and M. F. Hayat, "A hybrid deep network based approach for crowd anomaly detection," *Multimedia Tools and Applications*, pp. 1–15, 2021.

[57] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.

[58] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[59] Y. Hu, "Design and implementation of abnormal behavior detection based on deep intelligent analysis algorithms in massive video surveillance," *Journal of Grid Computing*, vol. 18, no. 2, pp. 227–237, 2020.

[60] A. A. Almazroey and S. K. Jarraya, "Abnormal events and behavior detection in crowd scenes based on deep learning and neighborhood component analysis feature selection," in *Joint European-US Workshop on Applications of Invariance in Computer Vision*. Springer, 2020, pp. 258–267.

[61] A. Aldayri and W. Albattah, "A deep learning approach for anomaly detection in large-scale hajj crowds," *The Visual Computer*, pp. 1–15, 2023.

[62] A. Baíllo and J. E. Chacón, "Statistical outline of animal home ranges: an application of set estimation," in *Handbook of Statistics*. Elsevier, 2021, vol. 44, pp. 3–37.

[63] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[64] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[65] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[66] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International workshop on artificial neural networks*. Springer, 1995, pp. 195–201.

[67] C. Esposito, G. A. Landrum, N. Schneider, N. Stiefl, and S. Riniker, "Ghost: adjusting the decision threshold to handle imbalanced data in machine learning," *Journal of Chemical Information and Modeling*, vol. 61, no. 6, pp. 2623–2640, 2021.

[68] "Entrance 2, entry with guiding barriers (corridor setup)," http://doi.org/10.34735/ped.2013.1, Pedestrian Dynamics Data Archive hosted by the Forschungszentrum Jülich, 2013.

[69] M. Boltes, A. Seyfried, B. Steffen, and A. Schadschneider, "Automatic extraction of pedestrian trajectories from video recordings," in *Pedestrian and evacuation dynamics 2008*. Springer, 2010, pp. 43–54.

[70] B. M. Eric Hofesmann, "Find and remove duplicate images with fiftyone," 2022. [Online]. Available: https://github.com/voxel51/fiftyone-examples/blob/master/examples/image_deduplication.ipynb

[71] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 4480–4488.

[72] Z. DeVries, E. Locke, M. Hoda, D. Moravek, K. Phan, A. Stratton, S. Kingwell, E. K. Wai, and P. Phan, "Using a national surgical database to predict complications following posterior lumbar surgery and comparing the area under the curve and f1-score for the assessment of prognostic capability," *The Spine Journal*, vol. 21, no. 7, pp. 1135–1142, 2021.

[73] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 096–10 106.

[74] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[75] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[77] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[78] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[79] S. Gupta and M. Tan, "Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl," *Google AI Blog*, vol. 2, p. 1, 2019.

[80] A. Alia, M. Maree, M. Chraibi, and A. Seyfried, "VCNN4PuDe: A Novel Voronoi-based CNN Framework for Pushing Person Detection in Crowd Videos," Jul. 2023, https://doi.org/10.5281/zenodo.8175476. [Online]. Available: https://doi.org/10.5281/zenodo.8175476

Publication IV

On the Exploitation of GPS-based Data for Real-time Visualization of Pedestrian Dynamics in Open Environments

This article has been published as Alia, Ahmed, Mohammed Maree, and Mohcine Chraibi. "On the exploitation of GPS-based data for real-time visualisation of pedestrian dynamics in open environments." Behaviour & Information Technology 41, no. 8 (2022): 1709-1723.

Author's Contributions

Conceptualization: **Ahmed Alia**, Mohammed Maree
Methodology: **Ahmed Alia**, Mohammed Maree, Mohcine Charaibi
Software: **Ahmed Alia**
Validation: **Ahmed Alia**
Formal Analysis: **Ahmed Alia**
Visualization: **Ahmed Alia**
Writing – Original Draft Preparation: **Ahmed Alia**
writing—review and editing **Ahmed Alia**, Mohammed Maree, Mohcine Charaibi
GitHub Repository Creation: **Ahmed Alia**

GitHub: `https://github.com/PedestrianDynamics/GPSdataColVis`

Publication DOI: `https://doi.org/10.1080/0144929X.2021.1896781`

On the Exploitation of GPS-based Data for Real-time Visualization of Pedestrian Dynamics in Open Environments

Ahmed Alia[1,2], Mohammed Maree[3*], Mohcine Chraibi[1]

[1] Institute for Advanced Simulation, Forschungszentrum Jülich, 52425 Jülich, Germany.

[2] Department of Management Information Systems, Faculty of Engineering and Information Technology, An-Najah National University, Nablus, Palestine.

[3] Department of Information Technology, Faculty of Engineering and Information Technology, Arab American University, Jenin, Palestine.

Corresponding author: mohammed.maree@aaup.edu

Abstract Over the past few years, real-time visualization of pedestrian dynamics has become more crucial to successfully organize and monitor open-crowded events. However, the process of collecting, efficiently handling and visualizing a large volume of pedestrians' dynamic data in real time is challenging. This challenge becomes even more pronounced when pedestrians move in large-size, high-density, open and complex environments. In this article, we propose an efficient and accurate approach to acquire, process and visualize pedestrians' dynamic behavior in real time. Our goal in this context is to produce GPS-based heat maps that assist event organizers as well as visitors in dynamically finding crowded spots using their smartphone devices. To validate our proposal, we have developed a prototype system for experimentally evaluating the quality of the proposed solution using real-world and simulation-based experimental datasets. The first phase of experiments was conducted in an open area with 37,000 square meters in Palestine. In the second phase, we have carried out a simulation for 5000 pedestrians to quantify the level of efficiency of the proposed system. We have utilized PHP scripting language to generate a larger-scale sample of randomly-moving pedestrians across the same open area. A comparison with two well-known Web-based spatial data visualization systems was conducted in the third phase. Findings indicate that the proposed approach can collect pedestrian's GPS-based trajectory information within four meters horizontal accuracy in real time. The system demonstrated high efficiency in processing, storing, retrieving and visualizing pedestrians' motion data (in the form of heat maps) in real time.

Keywords: Real-time Visualization, Pedestrian Dynamics, Crowd Management System, GPS Data, Heat Map Visualization.

1 Introduction

With the rapid growth of population along with the increasing number of important events, dynamically finding less-crowded areas within large-scale events is a major challenge for visitors as well as event organizers [1, 2]. As reported in [3], inefficient crowd management affects pedestrians' safety, their movement decisions and behaviors, and may also lead to increasing the efforts and costs required to efficiently monitoring events with large numbers of pedestrians [4]. In order to

organize successful events, organizers need to understand the visitors' movement characteristics and behavior over time at an early stage [5]. Accordingly, greater attention has been given to study crowded events with huge numbers of pedestrians using a variety of tools and applications, such as pedestrians' devices that are used to collect their trajectories in an attempt to efficiently identify crowd density along events [6]. These approaches can be divided into three main categories [6]: crowd video analysis [7, 8], crowd social media analysis [9, 10, 11], and crowd spatio-temporal analysis [12, 13, 14].

Developing systems under the first category plays an important role in analysing crowded event videos as they collect data about the movement features of pedestrians and assist in identifying abnormal behaviors during events [3, 15]. However, the large-size areas of crowded events and high-density of pedestrians in such events makes it difficult for such approaches to accurately respond in real time, and leads to producing a large fraction of false positive results [16]. In addition, the presence of obstacles, such as walls, trees and human bodies can block cameras from capturing and identifying correct abnormal behaviors or even true pedestrian objects [17]. Moreover, the cost and setup of cameras using this approach are normally high as reported in [6]. On the other hand, approaches that fall under the second category use social media content, in addition to other important data, such as locations of user check-ins as their data source to study human movement behavior [18]. Using such data sources, these approaches generate spatio-temporal data points and location details that can be used to analyze crowds in large areas in cities, suburbs and urban areas [19, 20]. However, it is important to point out that approaches of this category can not make use of real-time collected data because the extraction of useful information from social media content is a very complex task [21]. In addition, social media data does not give information about human movement characteristics continuously at regular time intervals. For example, sharing check-ins allows users to mark and discuss places they visited (e.g., eating at local restaurants, shopping, visiting popular areas) as part of their social interaction online [3, 11]. As such, we can conclude that approaches that fall under the first two categories may not generate sufficient information about human movement characteristics during high-density large-scale events in real time [22, 23]. Acknowledging these drawbacks, crowd spatio-temporal analysis approaches have proved to be an effective solution for collecting pedestrian data continuously during large-area events [5].

To efficiently achieve their goal, these approaches have utilized Global position system (GPS) as their source of trajectory data points acquisition [24]. It indeed offers real time response, easy and cheap to navigate feature, and has a good accuracy in outdoor areas [25]. All of these features contribute to making GPS an efficient option for real time data collection about human movements in open areas and environments [24, 26]. Recently, smartphones' built-in GPS receivers have been one of the most promising and convenient GPS devices, especially, as they are widely used worldwide. The expected number of smartphones in 2020 is 2.5 billions [27]. Also, smartphones can receive real-time data based on the current position of pedestrian's body at regular time intervals with good accuracy in outdoor areas [25]. As such, GPS and smartphones have allowed researchers to explore, investigate, and monitor pedestrians' movement characteristics in outdoor areas more accurately and less invasively compared to other conventional approaches. In recent years, researchers have used mobile applications along with existing GPS data and web servers to develop real time visualization systems that

can monitor pedestrian movement scenarios in open events [28, 29]. However, existing visualization approaches suffer from a number of drawbacks and limitations as follows. First, with the increasing number of pedestrians in the same area, their performance degrades failing to cope with the real time visualization requirement. Second, the accuracy of the collected trajectories as well as visualizing tools is not tested under real-world conditions or using real-world scenarios. Third, using paid services or libraries increases system development costs. To address these limitations, we propose developing an efficient and low-cost system for online and real time visualization of pedestrian dynamics at open events/areas using GPS data. The system is aimed to be installed on users' smartphone devices to track pedestrian movements, collect and send GPS data to a web server in real time. Unlike existing approaches, the developed system is characterized by its efficiency in terms of the following aspects:

1. The system collects pedestrians' trajectory data every second and estimates their horizontal accuracy to accurately depict the maps.
2. New data processing with temporal storage of current pedestrian positions is used to improve the data visualization performance.
3. The system provides access to an online real time visualized maps that depict pedestrian movements along with real time heat maps for spotting crowded areas.
4. The system exploits efficient free/open source software to reduce the cost of practically deploying the developed application.
5. An archive of collected GPS data with multiple formats (SQL, JSON, and CSV) is provided for researchers interested in further developing and improving the current version of the proposed system [1].

The rest of this paper is organized as follows. In section 2 we explore and discuss a number of research works that are related to our proposed approach. Section 3 presents the proposed system and highlights its main components. In section 4, we provide details on the experimental setup phase. Experimental results are then discussed in section 5. Finally, section 6 concludes this article and discusses the future extensions of our work.

2 Related Work

Accurate real time acquisition, processing, analysis and visualization of pedestrian trajectory data during crowded events are the most crucial challenges for existing approaches that attempt to investigate and identify the most influential factors on pedestrians' dynamics and their movement behavior and characteristics [30, 31, 28, 29]. Addressing these challenges helps event organizers make immediate decisions to avoid crowd accidents, and ensures proper monitoring and administration of event operations [3]. With the recent developments of smartphone industry, the possibility of tracking human positions via their GPS-enabled devices has been achieved, providing a good opportunity for developing low cost and real time data processing and visualization techniques that can assist in better understanding pedestrian dynamics at open areas that are characterized by their complex and

[1]The system's prototype and data sets can be accessed at`https://github.com/PedestrianDynamics/GPSdataColVis`

irregular nature. Among the research works that were carried out in this domain is the work proposed by Waga et al in [30] where the authors developed a system to track pedestrians using their smartphones. The collected tracking data was stored and visualized using google maps. In particular, they sent GPS tracking details to a web server every 30 seconds where they were stored using MySQL database management system. Then, to speed up the visualization component, tracking data was updated into files every 24 hours in worst case, depending on the users and the time range of tracks. Due to the time interval for updating files (which are the data source for the visualization component), the system's visualization component was inefficient in producing maps for the current collected GPS points at real time in the same manner as performed in [31]. In addition to that, [30] and [31] did not incorporate heat maps as part of the visualization module, causing the visualizing to be less effective and user friendly as we propose in our work.

Another visualization technique was described in [28] to study the behavior of crowds in a large festival (in 2013 at Zurich city in Switzerland) over a period of three days. A mobile application was developed in this context to collect users' locations continuously, and visualize their presence during the festival. To do this, the researchers collected users' GPS data and sent it to a server every 2 seconds to be stored in a database. The stored data was processed at an interval of 2 minutes to be visualized in the form of maps. The authors experimentally tested the accuracy for several locations, and they obtained an accuracy between 150 to 500 meters. According to the authors, the proposed approach was not developed for real time pedestrian tracking and visualization.

In [29] a festival's crowd conditions (crowd density, crowd turbulence, crowd velocity and crowd pressure) were visualized in real time using pedestrians' GPS location traces. To carry out this task, a mobile application was used to collect GPS locations and send them periodically after processing to a server which stores them in a database. In its heat map visualization component, multiple mathematical models and methodologies were used to generate four different heat maps to infer four crowd conditions at specific time point. The main drawback of this approach is the computational complexity required to build the heat maps. In addition, the proposed system did not aim to detect the crowd conditions during the event, but only at specific time point.

In a similar line of research, Yun et al. [32] analyzed tourists' spatio-temporal behavior at the rural festival in South Korea for five days. The authors developed a mobile application and a simple questionnaire to track the festival visitors and collect accurate spatio-temporal information about them. The questionnaire was conducted to know the visitors' socio-economic characteristics in addition to week day and weather to determine the effects of these characteristics on the crowd. In this context, each participant installed the mobile application to record his/her track, and received a personally administered questionnaire. At the end of the experiments, participants reported back their data by submitting the questionnaires, in addition to uploading their tracking records (to a web server) in order for the researchers to depict and analyse the produced heat maps. However, the proposed approach did not collect and visualize the GPS tracks in real time, and the accuracy issue was not addressed by the researchers.

As we have highlighted in the above-mentioned discussion and to the best of our knowledge, the coupling of GPS data acquired using users' smartphones and heat maps for real time visualization purposes to understand the dynamics of pedestrians moving in large-scale open environments has been very little. Never-

theless, we can summarize the main drawbacks of existing approaches that have attempted to employ visualization approaches and techniques for investigating the pedestrians' trajectory data as follows:

- Low accuracy of visualization: The lack of accuracy measurements for collected positions has led to lowering the accuracy of heat map visualization. In general, smartphones are typically accurate within a 4.9 m radius under open sky and with normal conditions, however, their accuracy can degrade based on signal blockage, atmospheric conditions and GPS receiver quality [33, 34]. Therefore, it is necessary not to submit positions with low horizontal accuracy for heat map visualization. Most of the existing approaches collected and provided the positions to the visualization component regardless of their accuracy.

- High computational cost: Some of the existing approaches used all or a large portion of the collected data to visualize the crowd density of active pedestrians which resulted in increasing the required time to visualize the maps [35].

- Inefficient storage and retrieval of GPS data points: losing the newly collected GPS points in heat map visualization degrades the quality of the produced maps. Some approaches attempted to address this drawback by selecting the required GPS points only for visualization without searching in all data, but they did not apply this protocol automatically along with each process of data collection which led to losing current pedestrians' positions needed for heat map visualization in real time.

- Real time: some approaches did not process data in all their components in real time which led to an inefficient real time approach.

In the next section, we introduce the architecture of our proposed approach and detail the methods and techniques that we employ to address the challenges of coupling GPS data points and heat map visualization techniques.

3 Architecture of the Proposed System

In this section, we first present the overall architecture of the proposed system. As depicted in Figure 1, data about each user's position is acquired every second and gets transferred to a dedicated web server for further processing, storage and visualization. To do this, the system employs several components and modules as detailed below:

Mobile-based Data Acquisition: We have deployed the developed application on client smartphone devices for tracking their positions and sending this data to the web server for further processing. In particular, the application uses the smartphone's GPS sensor to determine users' positions while they are moving in an open space. After the user starts the application and agrees to use its associated GPS data collector for research purposes, the application starts collecting the current position of pedestrians moving at open events continuously (every second), and directly transfers the obtained GPS data over an active internet connection to the web server. The application remains active unless it is explicitly closed by the user. To ensure protecting users' privacy, the application does not collect any private information about the smartphone, as it just collects

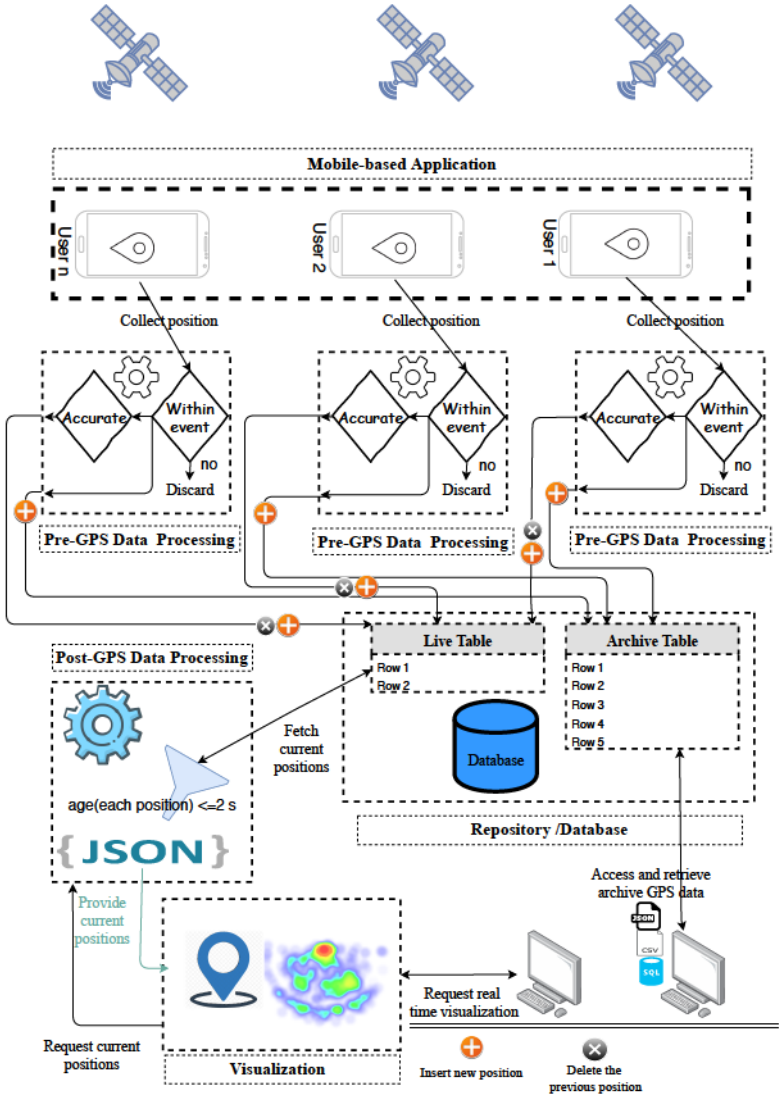

Figure 1: Overall architecture of the proposed system.

the positions (latitude and longitude) that are spotted inside the event's area to be visualized later in the form of heat maps. To further clarify this method, we present the algorithmic steps that we perform to carry out this task. As we can see in Algorithm 1, Lines 2 to 6 show how we acquire the GPS data which contains the latitude, longitude and horizontal accuracy.

Pre-processing of GPS Data: This component is employed for real time processing of GPS data obtained from the mobile application as we described in the previous step. Algorithm 2 illustrates the methods that we utilize to carry out this step.

As we can see in Algorithm 2, the input is received in the form of the current user's position that is located within the event area. This data gets accordingly stored in two database tables (Archive and Live Tables, respectively) depending

Algorithm 1 Android-based mobile GPS data collection application.

Output:
record ← *array*(
["latitude"]: latitude of the user's position,
["longitude"]: longitude of the user's position,
["accuracy"]: horizontal accuracy of the user's position,
["mobileId"]: an identifier for the user's mobile)
▷ confirmation message for storing the record into the web server
conMsg ← NULL

Require: internet connection

while Application is running **do**
  Location ← **Call** requestLocationUpdates(GPSprovider)
  record.latitude ← Location.getLatitude
  record.longitude ← Location.getLongitude
  record.accuracy ← Location.getAccuracy
  record.mobileId ← get aplication Id
  Print record
  create HttpPOSTConnection with the web server
  open HttpPOSTConnection
  submit record over HttpPOSTConnection
  conMsg ← read HTTP response message
  Prompt conMsg
  if close button is clicked **then**
    disconnect HttpPOSTConnection
    break
  end if
  wait a second
end while

Algorithm 2 Pre-processing of GPS Data.

Input:
//submitted record from mobile application
record ← *array*(
["latitude"]: latitude of the user's position,
["longitude"]: longitude of the user's position,
["accuracy"]: horizontal accuracy of the user's position,
["mobileId"]: an identifier for the user's mobile)
//Event area boundary
minLatitude ← 32.22677
maxLatitude ← 32.22955
minLongitude ← 35.21962
maxLongitude ← 35.22494
//Minimum accepted collected position accuracy for visualization
minAccuracy ← 5

Output: msg ← NULL

if record.latitude ≥ minlatitude and record.latitude ≤ maxlatitude and record.longitudey ≥ minLongitude and record.longitude ≤ maxlongitude **then**
  connect to database
  ArchiveInsertSQL ← insert record into Archive Table
  if execute(ArchiveInsertSQL) is successful **then**
    msg ← record is Archived
  else
    msg ← record is not Archived
  end if
  if record.Accuracy≤minAccuracy **then**
    if Live Table contains record from the same mobile **then**
      execute(delete the record from the Live Table)
    end if
    liveInsertSQL ← insert record into Live Table
    if execute(liveInsertSQL) **then**
      msg ← msg + and is ready for visualization
    end if
  end if
  Print msg
  disconnect from database
end if

on different criteria as follows: In the Archive Table, processed user positions are always stored for data archival purposes. For all users' positions that are within a "five meters" accuracy interval are dynamically transferred and stored in the Live Table after deleting the user's previous position from the table, more details

about these tables is provided in the Data Repository Component.

Data Repository Component: This component is employed as a reference data repository for the collected positions of pedestrians inside the event. It namely consists of two relational database tables: 1) an Archive Table that works as a repository for all collected positions of moving pedestrians, and 2) a Live Table which is a storage for the current positions of the pedestrians for real time visualization. In other words, the Live Table helps the proposed system to achieve real time visualization of the current users' positions as it contains a small number of positions which are equal to the number of current pedestrians. This means, the computational time required for retrieving and processing GPS data from the Archive Table will be increasing over time while using the system. Therefore, using only one table for archiving and visualizing is inefficient for real time visualization, because it retrieves all the previous/historical and current positions to return the current positions. The structure of the Archive and Live tables is presented in Table 1. Finally, this component can provide the GPS datasets in different formats (SQL, CSV and JSON) to enable further processing of the produced data using a variety of tools and techniques.

Table 1: Data structure of archive and live tables.

Field	Description
id	A unique identifier for the position
mobileId	An identifier for the mobile
latitude	latitude of the user's position
longitude	longitude of the user's position
timestamp	Date and time for the obtained position
accuracy	horizontal accuracy of the user's position

Post-processing of GPS Data: This component assists the visualization component to keep the Live Table up to date and retrieve and return the content of the Live Table in JSON format. As we discussed before, the Live Table aims to store the current positions of active pedestrians only during the event, pre-processing of GPS data component updates the pedestrians' positions continuously, but it can not remove the last position of each pedestrian who has left the event from Live Table. As demonstrated in Algorithm 3 in lines 8-12, the post-processing of GPS Data component removes the old positions before retrieving.

Real-time Data Visualization: The main goal of this component is visualizing pedestrians' trajectory data at real-time through normal/point map and heat map formats. It receives the current positions for all available pedestrians in the event from the post-processing, and visualizes them on a web browser. Normal map is used to represent the current position of each pedestrian as a point (Longitude and Latitude) on the map, while current pedestrian positions get automatically updated every second on the map. In this context, points on the map will be updated automatically without updating the map in the background. On the other hand, we use heat maps (one of the most commonly used methods for visualizing extensive point data sets) to continuously visualize and analyze large data sets and identify data clusters. The types of maps have proved to be helpful in obtaining an overview of the current crowd density at a glance [38]. In this context, a heat map is graphically-depicted to represent spatial data where regions are colored depending on measurement values found at the specific location [29]. In our work, the heat map represents hot and cold areas on the

Algorithm 3 Post-processing and visualization.

Input:
//normal map or heat map
mapType ← normal or heat
//map updating interval time in second
updatingDuration ← 1
//close the visualization

Output: normal map or heat map

```
1: importing leaflet libraries [36]
2:     //Create an initial map of the center of the event
3: map ← Call create map (Default Map center [32.227956522256136,35.22212731651962], Default Zoom: 18)
4: Leaflet-providers← Call OpenStreetMap layer;
5: previewing the initial map
6: connect to database
7: while request updates every second do
8:     execute(delete records from Live Table where      timeStamp=currentTime-2seconds)
9:     data ← array()()
10:    while record←fetch record from Live Table do
11:        data.push(record.latitude,record.longitude)
12:    end while
13:    data←json.encode(data)
14:    reset map
15:    if mapType = normal  then
16:        for i=0;i¡length(data);i++ do
17:            marker(data[i]).addTo(map)
18:        end for
19:    else
20:        layer ← heatLayer(data)[37]
21:        addLayerTo(map)
22:    end if
23:    if map is closed then
24:        break
25:    end if
26: end while
27: disconnect from database
```

basis of pedestrians' movement densities [39]. The hot areas (red areas) are regions where the density of pedestrians is high (more than 65%, see [40] and [37]), the cold areas (blue areas) are regions where the pedestrians density is low (less than 40%, see [40] and [37]), and the yellow colored areas are with densities in between. Figure 2 shows an example of a normal map and a heat map that are obtained from real experiments that we have conducted at the open theater at An Najah National University. In Algorithm 3, lines 1-4, we detail the steps that are required to initialize the maps of the event, both normal and heat maps. In line 7, it requests the new positions from the Post-processing of GPS Data component and convert them into JSON format in line 13. Then it updates the normal map (lines 15-18), and also updates the heat map in lines 20-21. Algorithm 3 explains Post-Processing and Visualization components in details.

4 Experimental Setup

In this section, we present the setup and details of our experiments. In order to evaluate the performance of the proposed system, we have examined the following aspects:

1. Efficiency of data collection module: Our goal in this context is to evaluate the system's ability of automatically collecting pedestrians' trajectory data at every second, in addition to the accuracy of the collected GPS-based data at open areas.

2. Real time data visualization: Our aim here is to measure the run-time of

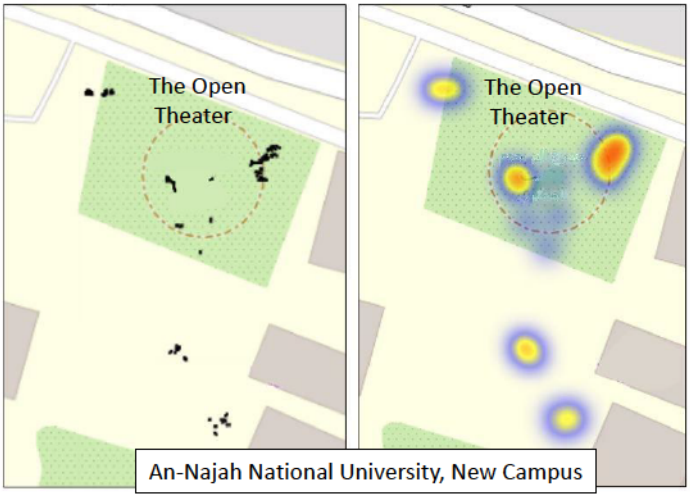

Figure 2: Screenshots of a visualization snapshot. Left: normal map. Right: heat map (High density: red, low density: blue, between low and high: yellow)

the proposed system's prototype, by calculating the computational time for every component of the system, in addition to calculating the Web page's loading time for the visualization process for both normal and heat maps.

To do this, we take an empirical approach to develop a prototype of the proposed system, conducting three types of experiments:

1. A real-world experimental scenario where we used pedestrians' smartphone devices to collect their trajectory data at open areas (see Figure 3) and submit it to a web server. The main goal of this experiment is to evaluate the efficiency of the data collection module and study the system's accuracy in collecting GPS-based trajectory data positions at open events/areas, as well as measuring the computational time of each component of the system's prototype. The real-world experiment was conducted at several open areas at the new campus at An Najah National University in Palestine. The focus was on the open theater area. The area of the new campus is about 137.000 square meters, and its bounding box is identified by the following latitude and longitude: (32.22682, 35.22493), (32.2294, 35.2196). Both an open area with no high buildings and another open area that is surrounded by high buildings were selected. Figure 3 shows the open areas and the main routes of the tracked pedestrians. Nine users with different types of android-based smartphones have installed the application and participated in this experiment. They were divided into three groups. Each group started walking normally from different points as shown in Figure 3.

2. A simulation experiment where we replaced pedestrians' smartphones with a PHP script to randomly generate a variable number of positions and submit them to the web server in an attempt to evaluate the server's computational time taken by the visualization process for both normal and heat maps with more auto-generated trajectory data.

3. A page loading time experiment where we compared the page loading time of the developed system's prototype with two well known web-based spatial

data visualization systems in an attempt to evaluate the performance of the real time visualization component of our system.

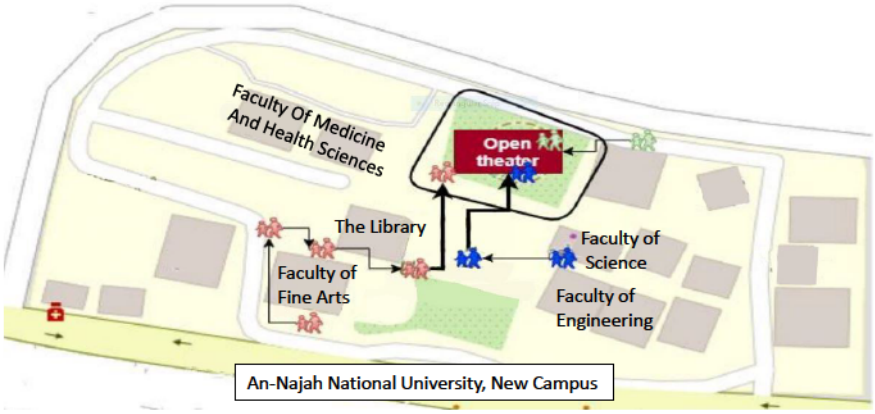

Figure 3: Real experiment area and the plan of pedestrians' movements.

We would like to point out that one of the main objectives of the proposed system is to be of low cost and efficient. Therefore, the developed prototype only exploited efficient free/open source software in the same manner as reported in [41]. In this context, we developed the android-based mobile application using Java and XML. To implement the data processing components, we used PHP, JavaScript and JSON. In addition, we used MySQL as our database engine and PHP to implement the data repository component. For the data visualization component, we employed Leaflet library [36], which is one of the most well known and efficient open source JavaScript libraries for visualizing normal and heat maps. In particular, the heat map used Leaflet.heat plugin [37] which uses a simple-heat visualization algorithm [40] that is combined with point clustering technique to form a performance grid [37]. The Simple-heat is a super-tiny JavaScript library for drawing heat maps on canvas focusing on simplicity and performance [40]. Here, the grid is colored with hot color(red) when multiple points are close to each other and with cold color (blue) when dispersed. The current system's prototype is hosted on a server with one core, Intel(R) Xeon(R) Silver 4214 CPU 2.20GHz and 256 MB RAM. It can be accessed easily by any Android-based device with an internet connection. The android-based mobile application is installed on different types of android-based mobile phones. Figure 4 shows a screenshot of the android-based mobile application and a screenshot of the home page of the system's prototype. All client implementations were run on a personal computer running Windows 10 with (i5) 2.4 GHZ processor, 8GB memory and Chrome browser.

5 Evaluation and Results

5.1 Real-World Scenario Experimental Results

This section aims to evaluate the efficiency of the data collection part by measuring the smartphone's GPS horizontal accuracy for collecting trajectory data positions at open event, and the required server computational time of each component in the proposed system's prototype.

Figure 4: Screenshots: Left. Android-based mobile application. Right. Home page of the system's prototype

5.1.1 Data Collection

The efficiency of the data collection part is discussed in this section, As shown in Table 2, 14184 positions were collected in this experiment on Mar 4, 2020, the experiment started at 11:27:40 and ended up at 11:56:54 based on local time of Palestine as illustrated in Figure 6. In Table 2 we also show a summary of the collected positions. The delay in connection to the internet and giving the permission to the android application by the user to start collecting the current position as well as the end permission caused a difference in the start and end times between the users, especially at the start time.

Table 2: Summary of the collected positions

User No.	Start time	End time	Duration (second)	No. of positions	Success %
1	9:27:40 AM	9:56:43 AM	1743	1644	94.32
2	9:28:03 AM	9:56:53 AM	1730	1724	99.65
3	9:28:12 AM	9:56:54 AM	1722	1722	100.00
4	9:29:47 AM	9:56:48 AM	1621	1581	97.53
5	9:31:44 AM	9:56:54 AM	1510	1510	100.00
6	9:30:19 AM	9:55:55 AM	1536	1534	99.87
7	9:28:13 AM	9:56:53 AM	1720	1695	98.55
8	9:29:32 AM	9:56:52 AM	1640	1632	99.51
9	9:37:47 AM	9:56:52 AM	1145	1142	99.74

The duration column in Table 2 shows the experiment's duration in seconds for each user. It is an indicator for the expected number of collected positions, where each second means one collected position for each user. Users 2, 3, 5 - 9 achieved more than 99.5% of expected number of collected positions, while users 1 and 4 collected 94.32% and 97.53%, respectively from the expected collected positions. One reason for the failure in collecting one position at one second is the weak internet connection. Figure 5 presents comparisons between the collected positions and the expected number of collected positions for each user. Also, the average of collected positions is 98.8% at different circumstances such as: open area, open area surrounded by high building, different types of android-based smartphones and poor internet connection sometimes. As a result, the android-based mobile application and the pre-processing of GPS data components were 98.8% efficient in collecting the current positions of pedestrians regularly at every one second.

Figure 5: Comparison between the number of collected positions and the number of expected positions.

5.1.2 Accuracy Evaluation

In this section, our goal is to study and measure the horizontal accuracy of the collected GPS data (latitude and longitude) by the android-based smartphones (smartphones' GPS horizontal accuracy); to see if they can collect trajectory positions with good horizontal accuracy.

In our prototype, we used the android location service to find the current position with its estimated horizontal accuracy. In this context, the term horizontal accuracy is defined as the radius (in meter) of 68% confidence [42]. In other words, if we draw a circle centered at this position's latitude and longitude, and with a radius equals to the accuracy value, then there is a 68% probability that the true location is inside the circle. Signal blockage, atmospheric conditions, and receiver design features/quality are the main local factors that affect GPS positioning accuracy [43, 34]. Therefore, in the real experiment, we took these factors into consideration to simulate real-world scenarios. The area of the experiment is divided into two parts. The first area is the open theater area with no high building, see Figure 6.b. Figure 6.c presents the second area which contains high buildings. The goal of this division is to measure the accuracy at the open area with no buildings as well as the area with high buildings using various android-based smartphone types, see Figure 6.

Table 3: Summary of the collected positions' accuracy.

	The open theater (Area b) Accuracy (meter)				**Area c Accuracy (meter)**			
User No.	Positions	Average	Best	Worst	Positions	Average	Worst	Best
1	532	7.43	7.2	7.6	1112	7.48	9.7	7
2	931	2.87	1.5	7.5	793	4.03	12.5	1.5
3	922	3.78	3	9.5	800	4.49	24	2
4	900	4.92	2.5	9	681	6.01	14	2
5	558	2.47	1	3.4	952	2.52	4.1	1
6	1134	3.22	3.22	3.22	400	5.46	10.5	3.22
7	909	3.77	2	9	786	7.19	34.5	2.5
8	557	3.56	2	6	1075	4.24	17	2
9	557	6.86	5	9.5	585	8.15	20	5

The results in Table 3 show the number of collected positions, expected average, the best and the worst horizontal accuracy measures for each user in both areas b and c. The total number of collected positions in the two areas is roughly the same (area b: 7000, area c: 7184). The average of the horizontal accuracy

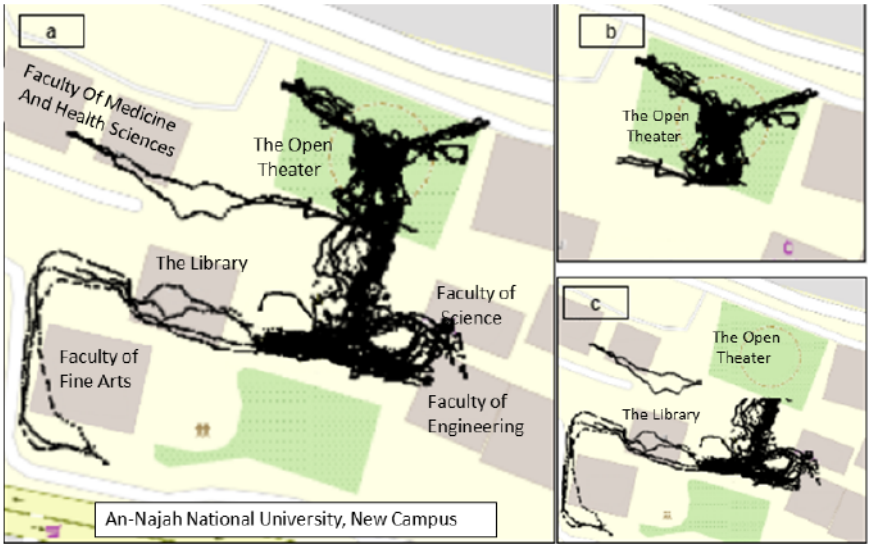

Figure 6: The real experiment area. a. All area of experiment. b. The open theater area which does not contain high buildings. c. This area is open with high buildings.

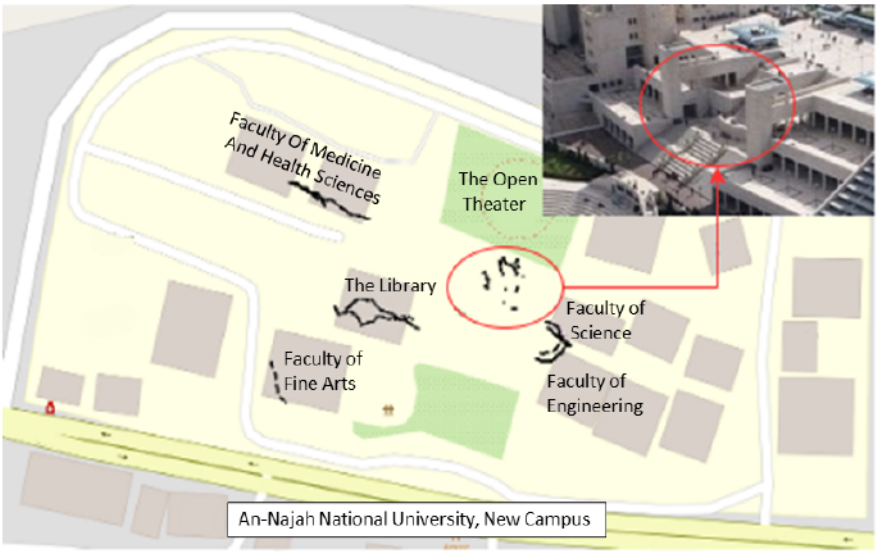

Figure 7: Locations of positions with accuracy grater than 10 meters.

for all users in area b was within four meters, while in area c, it was less than 5.18 meters. These results show that the accuracy is affected by the presence of high buildings. To guarantee the accuracy of our system, the post-processing component only processes the accurately (within 5 meters) collected positions for visualization component. Figure 7 explains the locations of the collected positions that have more than 10 meters as an accuracy, where all of them were located near to high buildings.

Many of the open and large important crowded events are organized at open environment with very little signal blockage as the open theater area in our study. Therefore, we are highlighting the results that were collected from the open theater

area. The significant difference in the average of accuracy between the users is depicted in Figure 8. As show in this figure, The lowest accuracy value was recorded for User 1 with 7.43 meters offset, while the highest accuracy value was marked for user 5 with 2.47 meters. The accuracy measures for six other users (2, 3, 5, 6, 7, 8) were less than 4 meters. We have considered the the same area (area b) and the same weather conditions for all users, however, the quality of installed GPS sensors on the used smartphone devices was different, which explains the variance in accuracy measures.

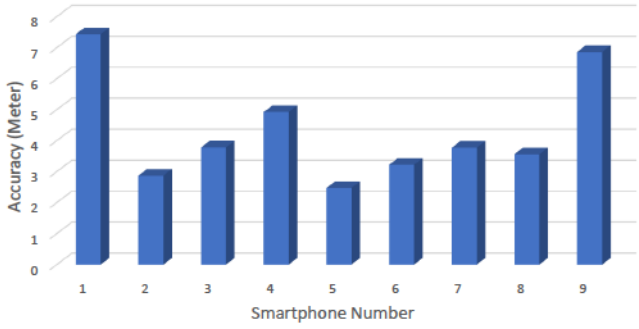

Figure 8: The average of accuracy for each user in the open theater area (area b).

Table 3 shows that the collected positions by user 1 and 9 are with more than 5 meters accuracy. Meanwhile, other users collected positions with better accuracy. The main reason for that is the quality of the GPS sensor used by users 1 and 9 which is less efficient compared to others [33]. As a result, modern android-based smartphones have a good ability to collect positions in open area with no high buildings within 4 meters accuracy or less. In comparison with iPhone 6-based Avenza software for capturing horizontal positions at open area that is reported in [44], our system achieves better accuracy which is within 4 meters, while the other system's accuracy is within 7-13 meters.

5.1.3 Server Computational Time

The section aims to evaluate the server computational time of the proposed system's prototype based on the real experiment. In addition to accuracy aspects of the proposed system, in these experiments we have also considered another important aspect that plays a crucial role on the overall's quality of the system. Our aim in this context is to evaluate the server's computational time of each of the various components of our proposed system's prototype. As we can see in Table 4, the GPS data pre-processing component took 18.8 milliseconds on average to pre-process each submitted position from the 14184 positions and to store this into the two tables of the repository component. On the other hand, the computational time for post-processing the received GPS data and further visualizing it in the forms of normal and heat maps for the nine users who were moving concurrently was recorded by the system. The total number of runs recorded in each component was 1130 times. The average computational time for post-processing GPS data (to process and retrieve the current positions) and send them to the visualization component for all runs was 0.43 milliseconds. For the visualization component, the average computational times needed to visualize normal maps and

heat maps were 12.34 milliseconds and 8.8 milliseconds, respectively. The computational time difference between both techniques increases significantly when they are applied on more data points. See Figure 9 and table 4. In the next section 5.2, we demonstrate the impact of increasing the number of data points on the required time for visualizing both types of maps.

Table 4: The computational time of each component is based on the real-world experiment.

Component	Number of runs	Computational time (millisecond)
Pre-Processing	14184	18.793446
Post-Processing	1130	0.42608
Visualization-Normal Map	1130	12.34
Visualization-Heat Map	1130	8.8

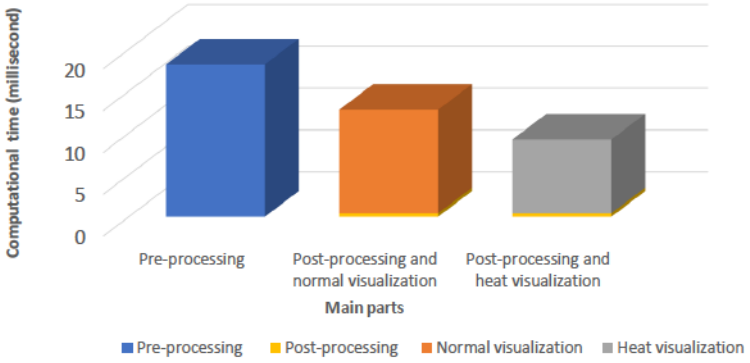

Figure 9: The computational time of the main components.

The post-processing component took less time compared to the visualization component for all runs. The main reason is because the pre-processing component (which is responsible for storing only the accurate positions within the specified area in a separate table) deletes the previous position of the same user from the data archiving table.

Based on the real-world experiment, the results show that the system's prototype can process, store and visualize the nine pedestrian positions in less than 35 milliseconds.On the other hand, post-processing with normal map visualization and post-processing with heat map visualization required about 41% and 34%, respectively of the computational time required by the pre-processing and repository components which needed more than 50% of the computational time. However, these percentages will change when applied to larger datasets of trajectory points because every request in the pre-processing component is applied on one position (it processes multiple requests at the same time in parallel), while every request in the post-processing and visualization components is applied on all current positions. This means that the total computational time of post-processing and visualization components increases when the number of the current positions increases. Therefore, the real-world experiment that was conducted to evaluate the system's prototype was not sufficient to evaluate the computational time of the post-processing and visualization components. Therefore, in next section, we test the post-processing and visualization components over more positions.

5.2 Simulation-based Experimental Results

In this section, we evaluate the server's computational time required by the post-processing and visualization components over a greater number of positions. As we have pointed out in the previous section, the number of positions that can be handled by the post-processing and visualization components affects the computational time of these components. In an attempt to demonstrate this effect, we have replaced the real-world android-based mobile application with a PHP script that generated a greater number of positions randomly within the area of the open theater and submitted them to a web server. To do so, nineteen experiments that started from 100 positions and finished with 5000 positions were conducted to evaluate the computational time for the employed components over different data sizes. Table 5 shows the results of this step. To reduce the effects of fluctuations in the computational time, each experiment was run twenty times, and we took the average of all times for each run.

Table 5: Computational time of post-processing normal and heat maps

		Computational time (milliseconds)		
Case	**Positions num.**	**Post-Processing**	**Normal visualization**	**Heat visualization**
1	100	0.99	136.55	120.2
2	200	1.4	158.2	119.2
3	400	1.956	211.56	122.7
4	600	2.698	256	125
5	800	3.262	311.86	146.2
6	1000	3.79106	356.8	155.2
7	1200	4.952	391	159.7
8	1400	4.99786	409	161.2
9	1600	5.8075	451	161.8
10	1800	6.74	504	161.9
11	2000	6.7908	553	163.6
12	2200	7.4439	571	163.4
13	2600	8.8449	655	175.4
14	3000	10.3908	822.63	184.8
15	3400	11.9399	883	205
16	3800	13.009082	1050	226.6
17	4200	13.8375	1140	234.2
18	4600	14.8353	1294	245.4
19	5000	16.479	1470	260.4

The results in Table 5 and Figure 10 show that the computational time for post-processing and normal and heat maps visualization components increases when the number of positions increases. Figure 10 presents the computational time needed to visualize the current positions in heat map and normal map formats for 19 cases. Each visualization process consists of two components, retrieving current positions (Live Table) by post-processing component and visualizing them by visualization component based on the map's format. The obtained results show that the post-processing component took very short computational time compared to the map visualization component. And the computational time needed for visualizing all positions in the largest case (5000 positions) in normal and heat maps were 1487 milliseconds and 277 milliseconds, respectively. The main reason of the difference in time between them are the leaflet plugins that are used by the system. Normal visualization used marker plugin [36] while heat map visualization used heat map plugin [37]. The representation of each point with icon in marker plugin [36] increases the computational time for loading and rendering, while the heat map plugin does not use any icon for representation [45]. In addition to that, Leaflet.heat plugin [37] used a simple-heat algorithm [40] that is very fast.

In Figure 10, normal map is affected significantly by increasing the amount of data. The time in normal map is duplicated 11 times while it is duplicated twice in heat map from the first case until the last case. Leaflet.heat plugin achieves much lower rendering times than the normal map (points map). According to [45], the leaflet.heat plugin rendered a maximum of three million points in 16,313.7 milliseconds, while normal maps [36] rendered a maximum of 100,000 points. This means, the heat map visualization is more suitable for visualizing a large amount of data in real time than normal maps. In addition to that, normal maps cover an entire map area, making each feature impossible to be identified [45].

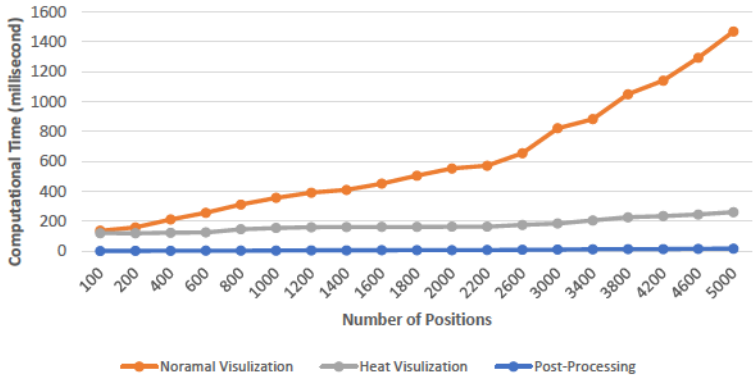

Figure 10: Computational time for post-processing and visualizing normal and heat maps over different numbers of Positions.

5.3 Page Load Time Experimental Results

In this section, we evaluate the performance of the real time visualization module of the proposed system's prototype. In particular, we compare the execution (a.k.a. Page Loading) time required by the proposed system's prototype, Maptive [46] and eSpatial [47] systems to load and display both normal and heat maps over 13 GPS data sets which are generated randomly. A Page loading time in this context is defined as the average amount of time it takes for a page to show up on a client device screen, and it is calculated from the moment at which the user clicks on a page link or types in a Web address until the page is fully loaded on the client's browser [48]. This comparison criterion is highly variable due to different factors, such as client devices, network connection, in addition to other technical specification as reported in [48]. Therefore, to make a fair comparison, we have used the same internet connection, client device, client browser and Chrome DevTools to measure the web page loading time. Both Maptive and eSpatial are efficient, available online and well-known real time web-based spatial visualization systems that support the visualization of both normal and heat maps. To carry out the experiment, we used Chrome DevTools [49] to measure the web page loading time at each run. Each experiment was run twenty times, and we calculated the average results produced per each run.

Table 6, Figure 11 and Figure 12 show the experimental results of web page loading time for the proposed system's prototype, Maptive and eSpatial web-based systems. As the results demonstrate, using the proposed system, we were able to achieve the best web page loading time required to visualize both normal and heat

Table 6: Web page loading time required by our proposed system against Maptive and eSpatial systems.

Data set	Positions num.	Maptive		eSpatial		The proposed system	
		Normal map	Heat map	Normal map	Heat map	Normal map	Heat map
1	5000	9,204	9,708	4,968	5,856	4,458	0,847
2	4600	8,678	9,506	4,808	5,686	4,254	0,6962
3	4200	8,626	9,312	4,306	5,344	3,8	0,589
4	3800	8,59	9,224	3,712	5,262	3,2	0,5388
5	3400	8,412	9,132	3,538	5,164	3,018	0,4702
6	3000	8,114	9,07	3,468	4,824	2,516	0,3944
7	2600	7,994	8,878	3,186	4,156	2,344	0,3902
8	2200	7,84	8,592	3,154	3,938	1,856	0,387
9	1800	7,722	8,432	3,07	3,824	1,51	0,3666
10	1400	7,6	8,294	2,994	3,704	1,342	0,3544
11	1000	7,58	8,192	2,922	3,694	1,092	0,3438
12	600	7,566	8,166	2,828	3,618	0,79	0,3386
13	200	7,468	8,09	2,66	3,584	0,502	0,3382
Average (second)		**8,11**	**8,82**	**3,51**	**4,51**	**2,36**	**0,47**

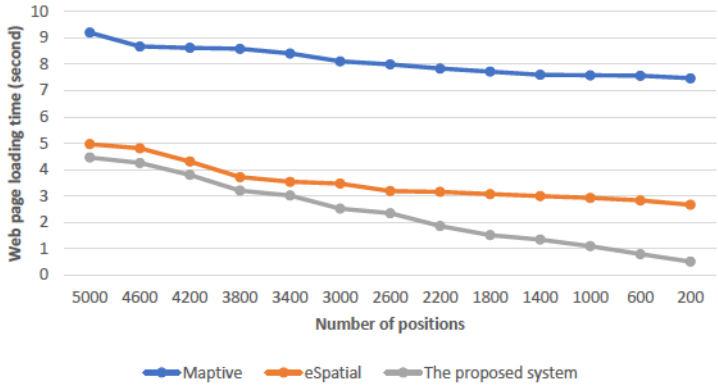

Figure 11: Web page loading time of normal map visualization over a different number of Positions.

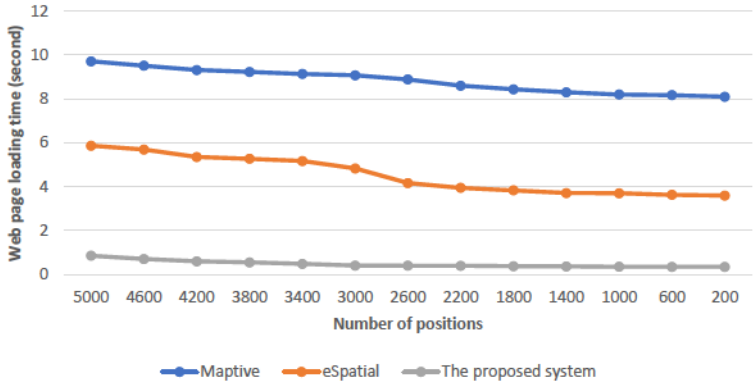

Figure 12: Web page loading time of heat map visualization over a different number of Positions.

maps over all data sets. On average, the proposed system took 0.71% and 0.33% less than Maptive and eSpatial web-based system respectively, to visualize the normal map. Moreover, it needed 0.05% and 0.1% from the web page loading time that are needed in Maptive and eSpatial to visualize heat maps. In other words,

our system's prototype visualized 5000 data points in heat map representation in less than a second while Maptive and eSpatial costed 5.8 seconds and 9.2 seconds respectively. The main reason for these results is the efficiency of the Leaflet libraries for depicting the maps that are used in our system [45]. In particular, we use a high performance heat map visualization algorithm combined with point clustering techniques to draw heat maps that are characterized by their simplicity [40]. In other words, The efficiency of the simple-heat algorithm makes the heat map visualization fast [40].

6 Conclusions and Future Work

The wide usage of smartphones in Palestine and in the world motivates us to utilize smartphones to track and explore pedestrian dynamics at open events; to avoid potential crowd movement related incidents. In this article, we have proposed a real time visualization system for pedestrian dynamics at open events. The proposed system integrated smartphone's GPS sensor, web server and open source/free software to provide an efficient, online, accurate and low cost real time system for collecting, storing and visualizing the pedestrian movements. In addition, this paper presented the first smartphone-based GPS accuracy study in open events/areas in Palestine. To evaluate our approach, a prototype was developed and tested using a real-world experiment at different open areas, nineteen simulation experiments, and 13 web page loading time experiments. The results demonstrated that our system collected the positions of pedestrians in open environment in real time and within an accuracy of 4 meters. Moreover, it processed and stored the collected position in 18.8 milliseconds, while retrieval and visualization of heat maps for 5,000 users took around 0.277 seconds. Also, the web page loading time in the developed system's prototype for heat map visualizing for 5,000 users was less than a second compared to Maptive and eSpatial which were 5.8 seconds and 9.2 seconds respectively.

Based on the system's prototype implementation and initial results, our proposed system showed promising results. It is expected to track the pedestrian and vehicles in real time at open and large areas efficiently. Therefore, in the future work, we plan to develop an iOS-based mobile application to cover a larger-scale portion of pedestrians who use various mobile platforms, and conduct extensive real experiments using both android and iOS applications. We plan in this content to investigate the efficiency of our system on real large GPS data sets. In addition, we plan to develop a new neural network approach to predict pedestrians' movement behavior, and detect abnormal events at real time that may occur in large open events.

Acknowledgement

This work was supported by the German Federal Ministry of Education and Research (BMBF: Funding number 01DH16027) within the framework of the Palestinian-German Science Bridge project.

Disclosure Statement

No potential conflict of interest was reported by the authors.

ORCID

Ahmed Alia http://orcid.org/0000-0002-3049-4924
Mohammed maree http://orcid.org/0000-0002-6114-4687
Mohcine Chraibi http://orcid.org/0000-0002-0999-6807

References

[1] U. Singh, J.-F. Determe, F. Horlin, and P. De Doncker, "Crowd forecasting based on wifi sensors and lstm neural networks," *IEEE transactions on instrumentation and measurement*, vol. 69, no. 9, pp. 6121–6131, 2020.

[2] K. Singh, S. Rajora, D. K. Vishwakarma, G. Tripathi, S. Kumar, and G. S. Walia, "Crowd anomaly detection using aggregation of ensembles of fine-tuned convnets," *Neurocomputing*, vol. 371, pp. 188–198, 2020.

[3] D. Sharma, A. P. Bhondekar, A. Shukla, and C. Ghanshyam, "A review on technological advancements in crowd management," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 3, pp. 485–495, 2018.

[4] B. Leopkey and M. M. Parent, "Risk management issues in large-scale sporting events: A stakeholder perspective," *European Sport Management Quarterly*, vol. 9, no. 2, pp. 187–208, 2009.

[5] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, pp. 1–55, 2014.

[6] Z. Ebrahimpour, W. Wan, O. Cervantes, T. Luo, and H. Ullah, "Comparison of main approaches for extracting behavior features from crowd flow analysis," *ISPRS International Journal of Geo-Information*, vol. 8, no. 10, p. 440, 2019.

[7] W. N. A. W. Samsudin and K. H. Ghazali, "Crowd behavior monitoring using self-adaptive social force model," *Mekatronika*, vol. 1, no. 1, pp. 64–72, 2019.

[8] T. Kulshrestha, D. Saxena, R. Niyogi, and J. Cao, "Real-time crowd monitoring using seamless indoor-outdoor localization," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 664–679, 2019.

[9] Q. Hu, G. Bai, S. Wang, and M. Ai, "Extraction and monitoring approach of dynamic urban commercial area using check-in data from weibo," *Sustainable cities and society*, vol. 45, pp. 508–521, 2019.

[10] R. Chaker, Z. Al Aghbari, and I. N. Junejo, "Social network model for crowd anomaly detection and localization," *Pattern Recognition*, vol. 61, pp. 266–281, 2017.

[11] M. Rizwan, W. Wan, O. Cervantes, and L. Gwiazdzinski, "Using location-based social media data to observe check-in behavior and gender difference: Bringing weibo data into play," *ISPRS International Journal of Geo-Information*, vol. 7, no. 5, p. 196, 2018.

[12] Y. Gong, Z. Li, J. Zhang, W. Liu, Y. Zheng, and C. Kirsch, "Network-wide crowd flow prediction of sydney trains via customized online non-negative matrix factorization," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1243–1252.

[13] X. Kong, H. Gao, O. Alfarraj, Q. Ni, C. Zheng, and G. Shen, "Huad: Hierarchical urban anomaly detection based on spatio-temporal data," *IEEE Access*, vol. 8, pp. 26 573–26 582, 2020.

[14] H. Xu, L. Li, and F. Fu, "Abnormal behavior detection based on spatio-temporal information fusion for high density crowd," in *International conference on Big Data Analytics for Cyber-Physical-Systems*. Springer, 2019, pp. 1355–1363.

[15] K. H. Cheong, S. Poeschmann, J. W. Lai, J. M. Koh, U. R. Acharya, S. C. M. Yu, and K. J. W. Tang, "Practical automated video analytics for crowd monitoring and counting," *IEEE Access*, vol. 7, pp. 183 252–183 261, 2019.

[16] A. Jabbari, K. J. Almalki, B.-Y. Choi, and S. Song, "Ice-mocha: Intelligent crowd engineering using mobility characterization and analytics," *Sensors*, vol. 19, no. 5, p. 1025, 2019.

[17] D. Gowsikhaa, S. Abirami, and R. Baskaran, "Automated human behavior analysis from surveillance videos: a survey," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 747–765, 2014.

[18] M. Q. Ngo, P. D. Haghighi, and F. Burstein, "A crowd monitoring framework using emotion analysis of social media for emergency management in mass gatherings," *arXiv preprint arXiv:1606.00751*, 2016.

[19] C. Wu, X. Ye, F. Ren, and Q. Du, "Check-in behaviour and spatio-temporal vibrancy: An exploratory analysis in shenzhen, china," *Cities*, vol. 77, pp. 104–116, 2018.

[20] A. C. Tricco, W. Zarin, E. Lillie, S. Jeblee, R. Warren, P. A. Khan, R. Robson, G. Hirst, S. E. Straus *et al.*, "Utility of social media and crowd-intelligence data for pharmacovigilance: a scoping review," *BMC medical informatics and decision making*, vol. 18, no. 1, p. 38, 2018.

[21] S. Stieglitz, M. Mirbabaie, B. Ross, and C. Neuberger, "Social media analytics–challenges in topic discovery, data collection, and data preparation," *International journal of information management*, vol. 39, pp. 156–168, 2018.

[22] S. Lamba and N. Nain, "Crowd monitoring and classification: a survey," in *Advances in Computer and Computational Sciences*. Springer, 2017, pp. 21–31.

[23] D. Duives, W. Daamen, and S. Hoogendoorn, "Monitoring the number of pedestrians in an area: The applicability of counting systems for density state estimation," *Journal of Advanced Transportation*, vol. 2018, 2018.

[24] X. Zhao, "On processing gps tracking data of spatio-temporal car movements: A case study," *Journal of Location Based Services*, vol. 9, no. 4, pp. 235–253, 2015.

[25] K. Konsolakis, H. Hermens, C. Villalonga, M. Vollenbroek-Hutten, and O. Banos, "Human behaviour analysis through smartphones," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, no. 19, 2018, p. 1243.

[26] S. Qureshi, "Creating a better world with information and communication technologies: health equity," pp. 1–146, 2016.

[27] J. L. Helbostad, B. Vereijken, C. Becker, C. Todd, K. Taraldsen, M. Pijnappels, K. Aminian, and S. Mellone, "Mobile health applications to promote active and healthy ageing," *Sensors*, vol. 17, no. 3, p. 622, 2017.

[28] U. Blanke, G. Tröster, T. Franke, and P. Lukowicz, "Capturing crowd dynamics at large scale events using participatory gps-localization," in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2014, pp. 1–7.

[29] M. Wirz, T. Franke, D. Roggen, E. Mitleton-Kelly, P. Lukowicz, and G. Tröster, "Inferring crowd conditions from pedestrians' location traces for real-time crowd monitoring during city-scale mass gatherings," in *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE, 2012, pp. 367–372.

[30] K. Waga, A. Tabarcea, R. Mariescu-Istodor, and P. Fränti, "System for real time storage, retrieval and visualization of gps tracks," in *2012 16th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2012, pp. 1–5.

[31] ——, "Real time access to multiple gps tracks." in *WEBIST*, 2013, pp. 293–299.

[32] H. J. Yun and M. H. Park, "Time–space movement of festival visitors in rural areas using a smart phone application," *Asia Pacific Journal of Tourism Research*, vol. 20, no. 11, pp. 1246–1265, 2015.

[33] C. Specht, P. Dabrowski, J. Pawelski, M. Specht, and T. Szot, "Comparative analysis of positioning accuracy of gnss receivers of samsung galaxy smartphones in marine dynamic measurements," *Advances in Space Research*, vol. 63, no. 9, pp. 3018–3028, 2019.

[34] "Official u.s. government information about the global positioning system (gps) and related topics," *Available online:https://www.gps.gov/systems/gps/performance/accuracy/*, (accessed on 22 April 2020).

[35] M. M. Masiane, A. Driscoll, W. Feng, J. Wenskovitch, and C. North, "Towards insight-driven sampling for big data visualisation," *Behaviour & Information Technology*, vol. 39, no. 7, pp. 788–807, 2020.

[36] V. Agafonkin, "Leaflet:a javascript library for interactive maps," *Available online: http://leafletjs.com/*, (accessed on 5 February 2019)).

[37] V. Agafonkin and et al., "Leaflet.heat." *Available online: https://github.com/Leaflet/Leaflet.heat*, (accessed on 30 June 2019).

[38] W. Kuhfled, "Heat maps: Graphically displaying big data and small tables," *SAS Institute Inc., Cary, North Carolina, USA*, 2017.

[39] N. Ihaddadene and C. Djeraba, "Real-time crowd motion analysis," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[40] Mourner and et al., "Leaflet.heat." *Available online: https://github.com/mourner/simpleheat*, (accessed on 24 Jul 2017).

[41] A. Bonaccorsi and C. Rossi, "Why open source software can succeed," *Research policy*, vol. 32, no. 7, pp. 1243–1258, 2003.

[42] P. Kouřil and M. Šimeček, "Usability of wi-fi fingerprint approach for place departure recognition in travel surveys," *Travel Behaviour and Society*, vol. 18, pp. 83–93, 2020.

[43] G. Padrón, T. Cristóbal, F. Alayón, A. Quesada-Arencibia, and C. R. García, "System proposal for mass transit service quality control based on gps data," *Sensors*, vol. 17, no. 6, p. 1412, 2017.

[44] K. Merry and P. Bettinger, "Smartphone gps accuracy study in an urban environment," *PloS one*, vol. 14, no. 7, p. e0219890, 2019.

[45] R. Netek, J. Brus, and O. Tomecka, "Performance testing on marker clustering and heatmap visualization techniques: A comparative study on javascript mapping libraries," *ISPRS International Journal of Geo-Information*, vol. 8, no. 8, p. 348, 2019.

[46] M. team, "web-based mapping software platform," 2010-2020.

[47] espatial team, "All-in-one mapping software," 2020.

[48] P. H. Shroff and S. R. Chaudhary, "Critical rendering path optimizations to reduce the web page loading time," in *2017 2nd International Conference for Convergence in Technology (I2CT)*. IEEE, 2017, pp. 937–940.

[49] C. D. team, "Chrome devtools," 2013-2020.

Conference Contributions

1. **Alia, Ahmed**, Mohammed Maree, and Mohcine Chraibi. Artificial Intelligence-based Early Pushing Detection in Live Video Streams of Crowds. 2023 the 3rd International Conference on Computers and Automation (CompAuto 2023) Paris, France – Dec 07-09, 2023. http://doi.org/10.34734/FZJ-2023-05241. [Talk: presented by Ahmed Alia].

2. **Alia, Ahmed**, Mohammed Maree, and Mohcine Chraibi. A Novel Voronoi-based Convolutional Neural Network Approach for Crowd Video Analysis and Pushing Person Detection. Helmholtz AI Conference 2023, Hamburg, Germany, 12 Jun 2023 - 14 Jun 2023. http:doi.org/10.34734/FZJ-2023-02439. [Talk: presented by Ahmed Alia].

3. **Alia, Ahmed**, Mohammed Maree, and Mohcine Chraibi. DL4PuDe: Deep-Learning Framework for Pushing Detection in Pedestrian Dynamics. Conference for Research Software Engineering in Germany, deRSE23, Paderborn, Germany, 20 Feb 2023 - 22 Feb 2023. http://hdl.handle.net/2128/34078.[Talk: presented by Ahmed Alia].

4. **Alia, Ahmed**, Mohammed Maree, and Mohcine Chraibi. A Fast Hybrid Deep Neural Network Model for pushing behavior detection in human crowds. IEEE/ACS 19th International Conference on Computer Systems and Applications, AICCSA, Zayed University, Abu Dhabi, U Arab Emirates, 5 Dec 2022 - 7 Dec 2022. http://doi.org/10.1109/AICCSA56895.2022.10017883. [Published short paper and Talk: presented by Ahmed Alia].

5. **Alia, Ahmed**, Mohammed Maree, and Mohcine Chraibi. A Real-Time Neural Network-based System for Pushing Detection in Crowded Event Entrances. Traffic and Granular Flow Conference, Indian Institute of Technology Delhi. 15 Oct 2022 - 17 Oct 2022. http://hdl.handle.net/2128/32081. [Talk: presented by Ahmed Alia].

6. **Alia, Ahmed**, Mohammed Maree, David Haensel, Mohcine Chraibi, Helena Lügering, Anna Sieben, and Ezel Üsten. Two Methods for Detecting Pushing Behavior from Videos: A Psychological Rating System and a Deep Learning-based Approach. 10th Pedestrian and Evacuation Dynamics Conference, PED2021, Melbourne. 29 Nov 2021 - 30 Nov 2021.http://hdl.handle.net/2128/29436. [Talk: presented by Ahmed Alia and Ezel Üsten].

Software Contributions

1. **DL4PuDe:** A Hybrid Framework of Deep Learning and Visualization for Pushing Behavior Detection in Pedestrian Dynamics, 2022. `https://github.com/PedestrianDynamics/DL4PuDe`, `http:doi.org/10.5281/zenodo.6433908`.
2. **CloudFast-DL4PuDe:** A Cloud-based Deep Learning Framework for Early Detection of Pushing at Crowded Event Entrances,2023. `https://github.com/PedestrianDynamics/CloudFast-DL4PuDe`, `http:doi.org/10.5281/zenodo.7570208`.
3. **VCNN4PuDe:** A Novel Voronoi-based CNN Framework for Pushing Person Detection in Crowd Videos, 2023. `https://github.com/PedestrianDynamics/VCNN4PuDe`, `http:doi.org/10.5281/zenodo.8175476`.
4. **GPSdataColVis:** GPS Data Collection and Visualization System, 2021. `https://github.com/PedestrianDynamics/GPSdataColVis`.

Schriften des Forschungszentrums Jülich
IAS Series

Band / Volume 50
Utilizing Inertial Sensors as an Extension of a Camera Tracking System for Gathering Movement Data in Dense Crowds
J. Schumann (2022), xii, 155 pp
ISBN: 978-3-95806-624-3

Band / Volume 51
Final report of the DeepRain project
Abschlußbericht des DeepRain Projektes
(2022), ca. 70 pp
ISBN: 978-3-95806-675-5

Band / Volume 52
JSC Guest Student Programme Proceedings 2021
I. Kabadshow (Ed.) (2023), ii, 82 pp
ISBN: 978-3-95806-684-7

Band / Volume 53
Applications of variational methods for quantum computers
M. S. Jattana (2023), vii, 160 pp
ISBN: 978-3-95806-700-4

Band / Volume 54
Crowd Management at Train Stations in Case of Large-Scale Emergency Events
A. L. Braun (2023), vii, 120 pp
ISBN: 978-3-95806-706-6

Band / Volume 55
Gradient-Free Optimization of Artificial and Biological Networks using Learning to Learn
A. Yeğenoğlu (2023), II, 136 pp
ISBN: 978-3-95806-719-6

Band / Volume 56
Real-time simulations of transmon systems with time-dependent Hamiltonian models
H. A. Lagemann (2023), iii, 166, XXX pp
ISBN: 978-3-95806-720-2

Band / Volume 57
Plasma Breakdown and Runaway Modelling in ITER-scale Tokamaks
J. Chew (2023), xv, 172 pp
ISBN: 978-3-95806-730-1

Schriften des Forschungszentrums Jülich
IAS Series

Band / Volume 58
Space Usage and Waiting Pedestrians at Train Station Platforms
M. Küpper (2023), ix, 95 pp
ISBN: 978-3-95806-733-2

Band / Volume 59
Quantum annealing and its variants: Application to quadratic unconstrained binary optimization
V. Mehta (2024), iii, 152 pp
ISBN: 978-3-95806-755-4

Band / Volume 60
Elements for modeling pedestrian movement
from theory to application and back
M. Chraibi (2024), vi, 279 pp
ISBN: 978-3-95806-757-8

Band / Volume 61
Artificial Intelligence Framework for Video Analytics:
Detecting Pushing in Crowds
A. Alia (2024), xviii, 151 pp
ISBN: 978-3-95806-763-9

Weitere ***Schriften des Verlags im Forschungszentrum Jülich*** unter
http://wwwzb1.fz-juelich.de/verlagextern1/index.asp

IAS Series
Band / Volume 61
ISBN 978-3-95806-763-9

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum