Efficient and Robust Machine Learning Methods for Challenging Traffic Video Sensing Applications

Yifan Zhuang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:

Yinhai Wang, Chair

Edward McCormack

Xuegang Ban

Program Authorized to Offer Degree:
Civil and Environmental Engineering

University of Washington

Abstract

Efficient and Robust Machine Learning Methods for Challenging Traffic Video Sensing Applications

Yifan Zhuang

Chair of the Supervisory Committee:
Professor Yinhai Wang

The development of economics and technologies has promoted urbanization worldwide. Urbanization has brought great convenience to daily life. The fast construction of transportation facilities provides various means of transportation for everyday commuting. However, the growing traffic volume has threatened the existing transportation system by raising more traffic safety and congestion issues. Therefore, it is urgent and necessary to implement Intelligent Transportation System (ITS) with dynamic sensing and adjustment abilities. ITS shows great potential to improve traffic safety and efficiency, empowered by advanced Internet of Things (IoT) and Artificial Intelligence (AI). Within this system, the urban sensing and data analysis modules play an essential role in providing primary traffic information for follow-up works, including traffic prediction, operation optimization, and urban planning. Cameras and computer vision algorithms are the most popular toolkit in traffic sensing and analysis tasks. Deep learning-based computer vision algorithms have succeeded in multiple traffic sensing and analysis tasks, e.g., vehicle counting and crowd motion detection. The large-scale deployment of the sensor network and applications of deep learning algorithms significantly magnify previous methods' flaws, which hinder the further expansion of ITS. Firstly, the large-scale sensors and various tasks bring massive data and high workloads for data analysis on central servers. In contrast, annotated data for deep learning training in different tasks is insufficient, which leads to poor generalization when transferring to another

application scenario. Additionally, traffic sensing faces adverse conditions with insufficient data and analysis qualities.

This dissertation works on proposing efficient and robust machine learning methods for challenging traffic video sensing applications by presenting a systematic and practical workflow to optimize algorithm accuracy and efficiency. This dissertation first considers the high data volume challenge by designing a compression and knowledge distillation pipeline to reduce the model complexity and maintain accuracy. After applying the proposed pipeline, it is possible to further use the optimized algorithm on edge devices. This pipeline also works as the optimization foundation in the remaining works of this dissertation. Besides high data volume for analysis, insufficient training data is a considerable problem when deploying deep learning in practice. This dissertation has focused on two representative scenarios related to public safety – detecting and tracking small-scale persons in crowds and detecting rare objects in autonomous driving. Data augmentation and Few-Shot Learning (FSL) strategies have been applied to increase the robustness of the machine learning system with limited training data. Finally, traffic sensing targets 24/7 stable operation, even in adverse conditions that reduce visibility and increase image noise with the RGB camera. Sensor fusion by combining RGB and infrared cameras is studied to improve accuracy in all light conditions.

In conclusion, urbanization has simultaneously brought opportunities and challenges to the transportation system. ITS shows great potential to take this development chance and handle these challenges. This dissertation works on three data-oriented challenges and improves the accuracy and efficiency of vision-based traffic sensing algorithms. Several ITS applications are explored to demonstrate the effectiveness of the proposed methods, which achieve state-of-the-art accuracy and are far more efficient. In the future, additional research works can be explored based on this dissertation. With the continuing expansion of the sensor network, edge computing will be a more suitable system framework than cloud computing. Binary quantization and hardware-specific operator optimization can contribute

Contents

Listing of figures

List of Tables

Acronyms

ACF Aggregated Channel Features

ADAS Advanced Driver-Assistance Systems

AI Artificial Intelligence

AP Average Precision

CA Cellular Automata

CLIP Contrastive Language-Image Pre-training

CNN Convolutional Neural Network

D-SSD Dual-Input SSD

FCN Fully Convolutional Network

FPN Feature Pyramid Network

FPPI False Positives Per Image

FPS Frames Per Second

FSL Few-Shot Learning

FWN Fusion Weight-Computation Network

HOG Histogram of Oriented Gradients

IA Illumination-Aware

IoT Internet of Things

I-MN Network with Illumination-Aware Network

IoU Intersection of Union

ITS Intelligent Transportation System

IT-MN Illumination and Temperature-Aware Multispectral Network

KLT Kanade-Lucas-Tomasi

LiDAR Light Detection and Ranging

L-SSD Late-Fusion SSD

MAC Media Access Control

MAE Mean Absolute Error

MLP Multilayer Perceptron

MR Miss Rate

MSE Mean Square Error

MSNet-F Multi-Scale Network with one-level Feature

NIN Network in Network

N-MN Network without Awareness Network

PAN Path Aggregation Network

ROI Region of Interest

RMSE Root Mean Squared Error

SNR Signal-to-Noise Ratio

SORT Simple Online and Realtime Tracking

SPP Spatial Pyramid Pooling

TA Temperature-Aware

TDM Top-Down Modulation

TOPS Tera Operations Per Second

T-MN Network with Temperature-Aware Network

UAV Unmanned Aerial Vehicle

Acknowledgments

The Ph.D. study is a challenging but attractive journey with much uncertainty. Through this journey, I have learned a lot of new things, not limited to knowledge but a broader vision of this world. I want to show my most sincere appreciation to everyone who helps me overcome these challenges.

Firstly, I would like to express my great gratitude to my advisor, Prof. Yinhai Wang. Without his support, a forward-looking view, and detailed guidance, I would be unable to make current achievements. Besides offering academic knowledge and research experience in my Ph.D. study period, Prof. Yinhai Wang also taught me how to be a reliable and trustworthy person, which will be my life treasure and benefit my future path.

I really appreciate Prof. Edward McCormack, Prof. Xuegang Ban, and Prof. Samuel Burden for serving as my Ph.D. committee members and providing valuable suggestions for my research. Their advice and comments have broadened my horizon significantly and helped me learn from different perspectives. What is more, I want to show my sincerest appreciation to Prof. Xuesong Wang from Tongji University. Collaboration with Prof. Xuesong

Wang and his team has offered a precious opportunity to work on emerging and practical challenges. Additionally, I am deeply indebted to Prof. Yi Zhang from Tsinghua University, who has provided valuable academia and career suggestions.

My colleagues from STAR Lab at the University of Washington, Dr. Ruimin Ke, Dr. Ziyuan Pu, Dr. Zhiyong Cui, Mr. Hao Yang, Mr. Chenxi Liu, Mr. Meixin Zhu, Dr. Wei Sun, Dr. Wenbo Zhu, and Dr. John Ash, provided significant support to my research works. It will be almost impossible to complete my research work without their assistance.

At last, I am extremely thankful to my parents, Mei Li and Weilin Zhuang, and other family members, who are always my solid backing and inspire me to overcome any challenges. Special thanks are given to Yu Wang, who has been accompanying me the whole Ph.D. study period and encouraged me to reach my goals step by step.

1

Introduction

1.1 Background

1.1.1 Intelligent Transportation System

Rapid economic growth and fast technology development have prompted city urbanization. Managing such fast-growing cities has become a critical challenge in many countries [149]. City urbanization brings superior convenience to residents but also raises extra social and envi-

ronmental problems, e.g., high energy consumption and severe air pollution, at the same time[229,103]. From the transportation perspective, residents and commuters enjoy various fast transportation tools but suffer more severe traffic congestion than ever. Based on the 2017 INRIX report, traffic congestion will cost drivers additional 480 billion dollars over the coming ten years in the most congested 25 United States cities[2]. Similar situations occur in China as its urbanization speed is much faster than in other regions worldwide. In Beijing, with a population of over 21 million, the traffic congestion annually adds a cost of 1,126 dollars per person. Besides, the waste of time on traffic congestion is also tremendous. In 2017, drivers in Seattle wasted an average of 78 hours stuck in traffic. This time was incredibly 100 hours in San Francisco. Urbanization increases the daily travel distance as commuters usually live far from their workplaces to save on housing costs. They will rely more on motorized transportation tools than on walking or bicycling because of the longer travel distance. Nowadays, an average one-way commute time is over 30 minutes in large cities. It is common to see a long queue in large transit centers. Traffic congestion is gradually becoming the hurdle to sustainable development[150]. Therefore, there is an urgent need to improve the current traffic status.

The introduction of AI and IoT technologies[244] has enabled cities with stronger and more intelligent sensing and adjustment abilities[75]. Cities utilize all kinds of sensors to collect data and automated analysis technologies to help make better decisions and improve multiple perspectives, e.g., safety, health, time, etc. In general, a smart city can be divided into three layers. The first is the technology base, including AI, edge sensors, and IoT communications. These advanced technologies support all upper-level functions. Beyond the technology base, the middle layer is responsible for data fusion, fusion, and mining, which further dives into

raw data and extracts valuable information. Finally, most residents can only interact with the last layer of broad applications, e.g., visualization and intelligent services. The promotion of smart cities benefits the transportation field and contributes to easing traffic congestion and shortening travel time. As a substantial portion of smart cities, ITS targets to elevate traffic safety and efficiency[348]. Daily commuters have easy access to real-time traffic information and adjust their travel plans. Traffic management agencies can automatically acquire and analyze dynamic traffic data using edge sensors and AI algorithms. Big traffic data helps build robust prediction and controlling models to optimize traffic congestion and cover more corner situations. By 2025, ITS is expected to cut down over 15% commute time on average[18] based on the city scale, transit system construction, and commuting patterns.

The general framework of ITS has four hierarchical components – urban sensing & data collection, data management, data analysis, and service providing[358]. Urban sensing, data management, and data analysis provide the data foundation for subsequent services, e.g., traffic signal optimization. Firstly, advanced sensing and IoT technologies have enhanced sensors for ITS, working as the frontend to collect traffic data. Modern sensing technologies can increase information variance and quality by providing more precise and low-noise raw data. IoT technologies can compress sensor sizes and improve communication ability. Thus, more sensors can be deployed to sites to cover more complicated situations. These sensors offer massive and detailed data that constructs a solid basis for traffic analysis, management, and optimization[251]. After collecting high-quality traffic data, AI technologies bring powerful and automated tools to analyze these complex data simultaneously in different forms and from diverse application scenarios. However, the vast data quantity and various analysis tasks raise a higher demand and bring serious challenges for accurate and efficient traffic

sensing[130].

1.1.2 Data Collection and Analysis Methods in ITS

Traffic data collection and analysis are the foundation of ITS and provide thorough extracted information for various traffic services, e.g., traffic flow prediction[199], and traffic signal timing optimization[325]. Generally, traffic data collection and analysis are two-step works conducted on different platforms. The first step is data collection through miscellaneous sensors, including automotive radar[203], Light Detection and Ranging (LiDAR)[306], magnetic sensor[66], and camera[165]. Each kind of sensor has its strengths and is suitable for different scenarios. Automotive radars are widely applied in parking assistant systems and automated speed-camera enforcement. The shared feature of these applications is that coarse object information is required. Both radars and LiDAR depend on sensing the distance between the sensor and target. The difference lies in that LiDAR uses light in laser pulses with more precise distance measurements. Therefore, LiDAR is much more accurate and offers denser data points. Although mobile devices have integrated the LiDAR system for 3D reconstruction, the automotive LiDAR system is still expensive, and each one can cost over $20,000. In contrast, radars only cost several hundred dollars. Magnetic sensors are the most power-efficient among these sensors and are mainly deployed for parking space and traffic flow detection without an external power supply. Nevertheless, magnetic sensors are easily affected by magnetic field change raised by surrounding objects, e.g., passing-py heavy vehicles. Cameras or vision-based sensing systems are the most common in ITS because of broad coverage, scalable deployment, and compelling analysis algorithms. Besides, sensor fusion is a favored strategy for utilizing the strengths of multiple sensors to cope with complicated situations, e.g., ad-

verse weather[322] and scenario reconstruction. For example, autonomous vehicles combine LiDAR and cameras in the perception system for better 3D scenario reconstruction since LiDAR offers 3D information and cameras offer detailed target information, e.g., category.

Whether single sensor or sensor fusion, cameras are an essential part of the current data collection system. The primary reason is to offer visual information close to human observations and convenient for manual analysis. In addition, vision-based analysis algorithms and tools are comparably mature and speed up the automated analysis process. Besides, IoT technologies further solve the data transmission problems and make large-scale deployment possible. Representative ITS applications using cameras include traffic light enforcement[238], parking space monitoring[148], traffic flow estimation[315], vehicle tracking[116], vehicle reidentification[192], and collision avoidance[84]. Self-driving is a hot topic in the technical field. Cameras play a principal role in the perception system. Furthermore, Tesla only relies on the vision-based sensing system in their autonomous vehicles and has impressive performance in production vehicles[280].

1.1.3 Computer Vision and Deep Learning in Sensing Traffic

Vision-based systems are mainstream in traffic sensing benefitting from the rapid development of sensor technologies and computer vision algorithms. This dissertation works on efficient and robust computer vision methods for challenging traffic sensing applications. After collecting images and videos, how to analyze them accurately and efficiently is significant, which will primarily affect next-step work, including traffic prediction and control optimization. The fast development of AI has enabled machines to learn about the world and respond to inputs based on practical experience. Machine learning has achieved remarkable success in

the transportation field, including traffic detection, forecasting, and management[344,51,25,16], by extracting high-level semantic information from low-level texture information in computer vision. Processing high-level information will be much easier as it has more standing features. For example, vehicle classification is one of the basic vision-based tasks in the transportation field. A general pipeline utilizes hand-crafted feature extractors to produce latent embeddings from input images or video frames. Then, one or cascaded classifiers will classify these embeddings and predict their categories with confidence scores[127].

Promoted by GPU acceleration and parallel computing, deep-learning-based algorithms, as a subfield of machine learning, show great success and have gradually replaced conventional machine-learning-based algorithms in many transportation applications[218]. Comparably, deep learning offers an end-to-end solution and simplifies the whole analysis process. Take the same example of vehicle classification. Deep learning models accept the input images and outputs classification scores directly. A loss function optimizes the model without adjusting different modules like conventional machine learning methods. Deep learning is founded on various neural networks, e.g., Multilayer Perceptron (MLP)[243,289], Convolutional Neural Network (CNN)[160,282] and transformers[104,68]. These networks have far more learnable parameters than statistical models in conventional machine learning, which bring unprecedented fitting ability to cope various input data. However, these complicated neural networks put a higher demand for computing resources, data sufficiency, and carefully designed training strategies. Deep learning applications always require powerful machines for data preprocessing, model training, and data analysis. Considering the weak computing power on the sensor side, data collection and other processes are usually on two separate platforms. There will be additional wireless or wired data transmission between the two platforms. Sensors are

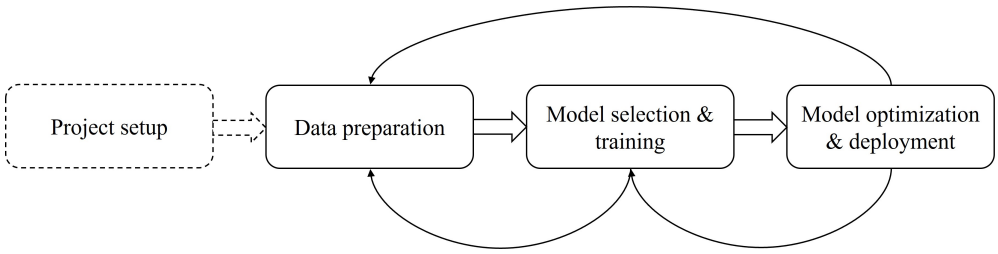


Figure 1.1: Deep Learning System Design Process

only responsible for data collection, and servers conduct data analysis, storage, and decision-making[172]. These two platforms are relatively independent. Such design, called cloud computing, simplifies the system structure and increases maintenance convenience[134,33]. However, cloud computing decreases system integrity. As sensor scale grows, there will be extra costs to upgrade data transmission bandwidth and cloud computing devices.

As this dissertation focuses on challenging vision-based traffic sensing applications from the algorithm perspective, it is necessary to introduce the general machine learning system design shown in Figure 1.1. The system design involves four steps – 1) Setting up the project; 2) conducting the data preparation; 3) selecting and training the model; 4) optimizing and deploying the model. All machine-learning-related works in this dissertation follow this procedure. Project setup will define the project goal, tasks, targets, and deliverables. The second step is building the data foundation for the project, including data collection, cleaning, and transformation following pre-defined rules in the first step. Data preparation may be the most critical step in practical applications as practitioners always adopt a mature model and train it on the target dataset. Data preparation should reduce data noise and ensure training data balance. After data preparation, the next step is model selection and training. The model selection indicates either picking up a state-of-the-art model or designing a project-specific model. In ITS applications, model training usually represents fine-tuning a well-built model

on the target dataset rather than training a vanilla model from scratch. There are two cogent reasons. Firstly, the pre-trained model can speed up the convergence and shorten the training time. Secondly, training samples are insufficient to support training the model from scratch, leading to poor generalization and the overfitting problem. The last step is optimizing the fine-tuned model to meet to demand proposed in the first step. Typical optimization methods contain model compression and training strategy adjustment.

Vision-based methods participate in various vision-based sensing tasks, e.g., object classification, detection, and semantic segmentation [58,195]. Furthermore, there are some researches on combining natural language and computer vision for more user-friendly services [253]. Object detection [367] and tracking [311] are two primary tasks in ITS and this dissertation. They are also basis of other vision-based tasks, e.g., surrounding agents' motion prediction [31,274], motion planning [219], and reidentification [360]. For example, intelligent parking facilities will reduce the search time, especially in urban areas, and indirectly ease traffic congestion. The automatic parking availability monitoring system can provide real-time parking occupancy information [82]. Drivers can quickly get directions to available parking slots if they have been in the parking garage. Others can adjust their routes based on the destination's parking availability. This system can also benefit truck parking by avoiding drivers' fatigue and improving road safety [330]. Another example is crowd monitoring, and handling emergencies like crowd abnormalities [269]. Urbanization makes people more likely to gather due to transition, entertainment, religious and commercial activities. Deep learning algorithms can help localize the abnormal crowd gathering or motion in complicated public facilities [79] to avoid crowd disasters [337]. What's more, the vision-based sensing system is essential to autonomous vehicles and Advanced Driver-Assistance Systems (ADAS) by offering collision avoidance function [213],

traffic signal recognition [198], lane line detection [310], and surrounding agent tracking [29].

As vision-based traffic sensing with deep learning becomes more popular and successful in various tasks, it is still necessary to rethink ITS application scenarios and existing challenges. Sensing in ITS has a high requirement for accuracy and efficiency as it is closely related to traffic safety. Additionally, traffic sensing usually works in severe outdoor environments, e.g., low visibility weather. Thus, much computer vision-related research in ITS concentrates on improving system accuracy and efficiency by proposing and modifying various neural networks. Deep learning is heavily dependent on data compared to statistical and conventional machine learning models. Inspired by this feature, this dissertation proposes robust and efficient computer vision solutions from the data perspective. According to Figure 1.1, data preparation is actually the first step in building a system after setting up the project. The data quality after this step can largely determine the latter performance. For example, well-annotated data in the training stage can help models learn from correct features and construct the accurate mapping from visual texture to target outputs, e.g., object categories and locations. In the validation and testing stages, cleaned data will reduce the negative impacts of noise. Meanwhile, data quantity will influence both training and inference time. This dissertation summarizes three highly correlated challenges found in traffic sensing applications.

1.2 Challenges

Although deep learning has significantly benefited and elevated vision-based traffic sensing in ITS, previously ignored and emerging challenges [22,277] hinder its next-step development and large-scale deployment with the urbanization procedure. According to Section 1.1, urbanization encourages the construction of transportation facilities. Thus, transportation agencies

have applied various sensors to monitor these newly appeared regions. With the development of technologies, high-resolution cameras are adopted to collect clearer images, and a single image has a larger size, which requires broader transmission bandwidth and more computing resources. Meanwhile, the public and transportation agencies are not satisfied with conventional traffic analysis results, e.g., traffic congestion status. They have a higher demand for more detailed and various information, e.g., abnormal activity detection. The centralized computing framework may not catch up with the rapid expansion of the sensor network and increased sensing tasks. In contrast to worrying about how to process the enormous amount of traffic data, machine learning users always face the problem of insufficient annotated training data, leading to a poor generalization ability to transfer algorithms to different scenarios. Meanwhile, ITS applications' data quality is not stable because traffic sensing always suffers adverse conditions with low data quality. Motivated by designing efficient and robust computer vision algorithms for ITS applications, this dissertation first summarizes three representative challenges related to data quantity and quality.

1.2.1 High Data Volume for Analysis

Worldwide urbanization has encouraged the construction and upgrade of transportation facilities, e.g., roadways and transit centers. Monitoring these regions is necessary and significant to traffic safety and efficiency. Transportation agencies and commuters are eager to learn real-time traffic information. Thus, various traffic sensors, i.e., cameras, are installed for data collection. Additionally, different regions may require distinct sensing tasks, e.g., vehicle counting and pedestrian tracking. Broader monitoring coverage and sensing tasks indicate higher data volume and workload on the computing platform. For example, there is

a growing need for enforcement surveillance cameras at high-risk locations for traffic safety monitoring[296]. Besides roadside sensors in ITS facilities, autonomous vehicles have a high demand for mobile sensors. Argo AI has installed two LiDAR, two forward stereo cameras, and eight ring cameras on their autonomous vehicles. The sample rate for LiDAR is 10Hz, and for cameras is 20Hz[34]. As a result, data volume from sensors rises steeply and places a high workload on the existing system. The situation is even worse for autonomous vehicles because they rely on onboard computing, which has a relatively weaker computing power than cloud platforms. One straightforward solution is enhancing the computing power by adding more computing devices or upgrading these devices. However, it is almost impossible to match the computing power on the server-side to the growing traffic data limited to the current framework. As stated in Section 1.1.3, most traffic sensing systems use the framework of cloud computing. Cloud computing-based traffic sensing systems have two procedures on separate platforms – data collection & transmission and cloud analysis[299,305]. Field cameras capture the raw data and transmit it back to servers for the next-step analysis and decision-making. More sensors ask for broader transmission bandwidth and more computer resources. The upgrading speed of transmission bandwidth and computing resources cannot catch up with the expansion speed of traffic sensors. The delay in data processing will become unignorable and intolerable. For time-sensitive applications like parking status monitoring and abnormal-event detection[302], the transmission and inference time is significant and worth thoughtful evaluation.

In addition to the growing demand for transmission bandwidth and computing resources, it is necessary to reduce the dependency on high-performance computing machines by improving the algorithm efficiency, especially for applications without such machines. For ex-

ample, there are around 13,000 signalized intersections in New York City. If 10% of them are monitored using four surveillance cameras, the demanded number of servers with Nvidia V100 or equivalent GPUs is approximately 160 for lively processing. When the coverage rate increases, the inference speed will drop correspondingly and cannot reach in real-time. Another example is autonomous vehicles and ADAS, which process data onboard because the transmission latency between vehicles and servers is unacceptable. What is more, the instability of communication is a hidden safety danger. Meanwhile, the computing power of in-vehicle computing devices is weaker than that of cloud computing devices.

Constrained by limited computing power, the algorithm should be as efficient as possible. However, the algorithm conversion from conventional machine-learning-based to deep-learning-based increases the model complexity conversely and worsens the situation. Compared to the conventional machine-learning-based algorithms using hand-crafted feature descriptors, deep-learning-based algorithms, e.g., Faster R-CNN[76] and FCOS[144], require extra computing resources to process the same amount of data. Furthermore, Using high-resolution images and the feature pyramid can improve the accuracy of detecting tiny objects in transportation scenarios[258,336]. High-resolution images will increase the operation number and further reduce the inference speed. Most previous deep learning and relevant transportation works focus on algorithm accuracy from the framework aspect and ignore the algorithm efficiency. As data volume keeps rising, algorithm efficiency is becoming as important as accuracy in ITS applications. Currently, there is no deployment workflow for machine learning algorithms in ITS, when there are more machine learning applications in the transportation field. Researchers and practitioners always follow the system design process from the computer science field. The process may ignore or underestimate some unique

features in transportation applications.

1.2.2 Insufficient Annotated Traffic Data

Although roadside and onboard traffic sensors produce tons of data every day[193], most data is not applicable for model training and improvement because unlabeled data is useless for the supervised training process. Using limited annotated data for machine learning model training faces data imbalance and insufficiency problems, which will lead to low accuracy in minor categories and poor generalization ability when transferring to similar tasks in different scenarios[345]. Millions of learnable parameters in deep-learning algorithms are a double-edged sword by offering an unprecedented fitting ability but requiring sufficient data to avoid overfitting[347]. Massive data can yield promising results. One impressive research work from OpenAI is Contrastive Language-Image Pre-training (CLIP)[230]. CLIP is trained on the WebImageText dataset containing 400 million image-text pairs and has achieved remarkable results in multiple downstream tasks using zero-shot, e.g., image retrieval and action recognition. Transportation applications have their specialties. Although public datasets can contribute to the base model training, there is a domain knowledge shift between the public dataset and target datasets, e.g., parking space classification[21], and vehicle detection from Unmanned Aerial Vehicle (UAV)[283]. Training on public datasets cannot provide satisfactory performance on the target tasks. Available training data in the target task usually contains fewer samples and suffers category imbalance[194,27] and statistic bias[6].

For example, traffic sensing usually faces new or uncommon objects that appear in different transportation facilities, as shown in Figure 1.2. The left Figure 1.2a displays an animal appearing in the center of the road, which may happen in the wild. The right shows a set

(a) Animals on Road **(b)** Unusual Traffic Signs

Figure 1.2: Sample Images of Uncommon Traffic Objects

of unusual traffic signs. In other fields, missing these uncommon objects may only lead to a high miss rate in some categories. Missing may cause traffic crashes and traffic rule violations in the transportation field, leading to more severe impacts. In addition, it is challenging to detect tiny and occluded persons using computer vision[184]. Two sample images of pedestrian detection using surveillance cameras[209,61] are shown in Figure 1.3, where objects are marked by bounding boxes. The Left and right images represent outdoor and indoor detection results, respectively. Targets, e.g., pedestrians, captured by surveillance cameras are always tiny and occluded in crowded public facilities. Therefore, this dissertation addresses two shared difficulties for traffic object detection applications. One is category imbalance in that some categories only have limited samples. Another is lacking data containing small-scale objects[341].

Generally, deep learning models are primarily trained with supervised learning and require sufficient annotated training data[105]. Ideally, categories in training data should be in uniform distribution and cover all possible scenarios. The training process can have a similar probability of exploring all categories with such training data. However, the fact is just the opposite in practice. Firstly, the category distribution in practice is always unbalanced, and some cate-

(a) Outdoor Pedestrian Detection [209]

(b) Indoor Pedestrian Detection [61]

Figure 1.3: Sample Images of Pedestrian Detection with Surveillance Camera

gories only have limited samples. The appearance frequency is different, related to the actual number in that area. The deep learning model fine-tuned on such a dataset will pretend to focus on categories with more training data. Secondly, insufficient training data can easily lead to overfitting problems when transferring the detection model to a new application scenario. The fine-tuning strategy using such datasets should be carefully designed. Under the supervised learning framework, model training cannot directly use data collected from traffic sensing systems before annotation.

Unluckily, data annotation is costly in both budget and time. Multiple vendors can provide data annotation services like Amazon SageMake Ground Truth and Google Cloud Data Labeling Service. For example, the price of annotating bounding boxes using Google Cloud Service in September 2021 was $63 per 1,000 images per human labeler. Different labelers should annotate the same image multiple times to ensure the annotation quality because there are human annotation errors. Cross annotation can help reduce errors but increase both cost and time. A cleared annotation guidance is necessary to guarantee the work quality and efficiency. Additionally, some uncommon and novel categories only have a few samples in raw data. It is impossible to create such data out of the air for model training. Conse-

quently, utilizing the unannotated and limited annotated data for model training is essential for deep learning research and practice. However, transportation applications may not have many resources like big technology companies to implement effective but expensive methods to fundamentally improve deep learning performance, e.g., labeling new data. There is an urgent need for cost-effective solutions to promote the system with limited annotated data.

1.2.3 Poor Data Quality in Adverse Conditions

Compared to computer vision tasks in other fields, e,g., retail product recognition, vision-based traffic sensing faces more severe application scenarios and weather conditions, e.g., raining and foggy days, and is more likely to fail to detect objects. Missing an object will lead to severe traffic safety issues, e.g., false-positive detection in self-driving. Thus, Miss Rate (MR) is more suitable than precision as the metric to evaluate the algorithm performance. The definition of MR is shown in Equation 1.1, where FN indicates the number of false negatives, and TP indicates the number of true positives.

$$MR = \frac{FN}{TP + FN} \tag{1.1}$$

Robust traffic video sensing in adverse conditions is complicated. The adverse conditions mainly shorten the visible distance and reduce the contrast between the target and the background. Thus, extracting valuable information from data collected in such an environment is challenging, especially with conventional hand-crafted feature descriptors. Deep-learning-based computer vision algorithms have promising performance and even surpass humans in some sensing tasks[92]. In addition, RGB cameras with the advanced CMOS can capture images and videos with much higher Signal-to-Noise Ratio (SNR) information in adverse con-

(a) RGB Pedestrian Image **(b)** Infrared Pedestrian Image

Figure 1.4: Sample Images of Multispectral Pedestrian Detection[126]

ditions than ever. However, there is still much noise in those images and video frames. With the increasing distance between the target and cameras, such noise can have more negative impacts. Meanwhile, texture information is lacking in low-light conditions, e.g., trails and rural roads at night, because no light is reflected from the target. High noise and lacking texture information are a disaster for computer vision algorithms. One potential solution is sensor fusion through synthesizing multiple kinds of data to overcome the poor data quality when using a single sensor.

This dissertation investigates one common adverse condition in traffic sensing – low light. There is not enough city light to light up the road, even in urban areas. Drivers are also likely to ignore crossing pedestrians when entering the intersection in these conditions. Therefore, robust traffic sensing under low-light conditions is necessary to improve traffic safety. According to the previous discussion, the conventional RGB camera may be incompetent for this sensing task, and sensor fusion contributes to the overall performance. The infrared camera is studied to help distinguish the target from the background based on the target's infrared radiation. Two sample images from the KAIST multispectral dataset[126] are shown in Figure 1.4, where orange bounding boxes draw the persons' position. The left image Fig-

ure 1.4a displays the RGB image and the right Figure 1.4b displays the thermal image. It is difficult to manually localize targets from Figure 1.4a, and harder for machines. The RGB image is more likely to miss texture information because the object is drowned in the background noise. In contrast, the thermal image can provide more precise visual information such as object contour because the target has infrared radiation. However, infrared cameras and thermal images cannot replace RGB cameras entirely and still have drawbacks because they receive infrared radiation and are sensitive to environmental temperature. Influenced by the environmental temperature, the infrared camera is more likely to confuse the target and background during the daytime when the object and background have similar infrared radiation. What is more, infrared cameras have much lower resolution, such as 320×240 and 640×480, than RGB cameras capturing 1920×1080 images. When the target is distant, detection using infrared cameras is also difficult. Therefore, it is necessary to develop practical sensor fusion algorithms to utilize multiple sensors and improve detection accuracy in adverse conditions.

1.3 Research Objectives

Motivated by the rapid development of ITS and the emerging challenges of deploying the machine-learning-enabled traffic sensing system, this dissertation proposes a systematic and generalized workflow to optimize traffic sensing accuracy and efficiency. Founded on three transportation-specific challenges described in Section 1.2, this dissertation firstly tries to improve deep learning algorithm efficiency and deploy them on edge devices if possible. Meanwhile, this dissertation faces realistic problems of working with insufficient and low-quality data for model training and inference. Originated from the transportation field, this disserta-

tion studies several transportation scenarios and factors in priority and highlights differences in computer vision algorithms between general and ITS applications.

The information era brings massive data generated every second from various sensors, which raises concerns about existing computing resources and algorithm efficiency. This dissertation first explores deep learning compression and post-training techniques to reduce model complexity and simultaneously keep accuracy. The server can thus process more data and operate multiple tasks with the same computing power. After model compression, it is also possible to deploy algorithms on edge and further distribute computing workloads[334]. The compression and optimization techniques are also crucial parts of the remaining chapters, which consider applying deep learning algorithms on edge devices. Traffic sensing will meet different targets, ranging from small-scale persons to large trucks. Some objects may appear more frequently, while others have a lower appearance frequency. It is critical to detect all targets and minimize false negatives. However, insufficient annotated traffic data is a common situation, which brings data imbalance and leads to overfitting problems. More specifically, this dissertation focuses on training a robust and generalized model in the condition without sufficient training data of lacking annotations of small-scale objects and uncommon categories. Instead of utilizing unlabeled data as self-supervised learning, data augmentation and FSL are considered to explore more valuable information from existing annotated data. Compared with computer vision tasks in other fields, traffic sensing is expected to operate normally under adverse conditions. Adverse conditions bring poor data quality when computer vision models are likely to fail to extract texture information drowned in background noise. This dissertation considers implementing sensor fusion by combining multispectral data sources for object detection[163,167,12]. Therefore, the primary research objectives are sum-

marized in the following four topics.

a) Optimize deep learning model efficiency, especially on the edge device, to process massive traffic data. This dissertation explores model compression, i.e., quantization, and knowledge distillation strategies to simultaneously boost inference speed and keep high accuracy. They are also fundamental optimization strategies applied to the remaining parts of this dissertation. Furthermore, these optimization techniques are not limited to computer vision applications and have a great potential to expand to other machine-learning-related transportation applications.

b) Improve tiny traffic object detection, i.e., in the crowd monitoring scenario. Crowd monitoring plays a vital role in traffic safety with the development of urbanization. However, crowd monitoring always faces challenges of localizing persons on a small scale. Small-scale object detection is significant as it provides essential information for crowd motion detection and group clustering. This study designs a novel data augmentation strategy to enhance tiny object detection accuracy. This strategy explores valuable information by increasing the tiny object sample number from existing training data.

c) Promote rare and novel traffic object detection accuracy. This dissertation develops a few-shot object detection algorithm to improve the detection accuracy of novel categories, e.g., wild animals on the road and unusual traffic signs, in the transportation scenario. Compared to the previous research objective of enhancing tiny object detection, this part concentrates on rarely appeared traffic categories in all scales. This dissertation tries to force the model to learn more features from uncommon or novel

categories with limited samples, thus promoting the detection accuracy in those rare categories while keeping detection accuracy in base categories. Few-shot object detection can also help data mining and reduce the workload of data annotation.

d) Improve vision-based traffic object detection accuracy in adverse conditions, especially low-light conditions. The biggest challenge in detecting objects in such an environment is poor data quality, which cannot be solved using the RGB information because almost no visible light is reflected from the object's surface. Sensor fusion can utilize the strengths of different sensors and generate high-quality features. This dissertation proposes a multispectral detection algorithm for 24/7 traffic object detection through fusing RGB and thermal images.

1.4 Dissertation Organization

This dissertation focuses on designing efficient and robust machine learning algorithms for challenging traffic video sensing applications. To be more specific, this dissertation works on three typical but tough challenges stated in Section 1.2 to demonstrate the effectiveness of the proposed algorithms. The overall framework is shown in Figure 1.5. There are seven chapters, including the introduction and conclusion chapters. In Figure 1.5, the left blocks show the chapter's primary topic, and the right texts describe their corresponding challenges. The summary of each chapter except the first is listed below.

Chapter 2 Literature Review will review the existing works about deep learning and computer vision applications in ITS, including deep learning models and optimization strategies. More specifically, this chapter will introduce state-of-the-art object detection algorithms and their applications in ITS. These algorithms can be divided into two subcategories based

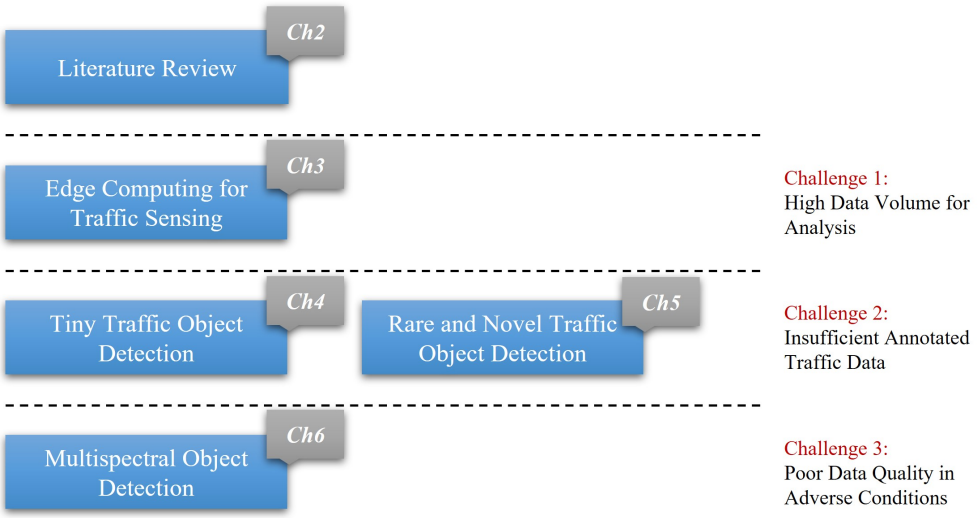


Figure 1.5: Dissertation Framework

on whether they have multiple input sources – monovision and multispectral algorithms. Monovision implies using a single input source, such as RGB images. In contrast, multispectral algorithms fuse different input sources to generate embeddings for classification, localization, and segmentation. Besides fundamental detection algorithms, research works in vision-based crowd monitoring will be reviewed, involving object detection and tracking. Then the second section of this chapter will summarize current solutions to two challenges – massive input data and insufficient annotated training data. For the high data volume challenge in Section 1.2.1, this chapter reviews two kinds of methods, e.g., deep learning compression and edge computing by reducing the model complexity and distributing computing tasks to multiple edge devices. Compared to the high data volume to process, annotation training data is insufficient. This chapter introduces self-supervised learning, data augmentation, and few-shot learning. More importantly, this chapter describes the reason for selecting the latter two instead of self-supervised learning methods.

Chapter 3 Edge Computing for Traffic Sensing in ITS focuses on fast-growing data volume for analysis. Unprecedentedly massive traffic data from the sensor network increases workload significantly in terms of transmission bandwidth, storage capacity, and computing power. With the development of urbanization and the construction of transportation facilities, these resources will be further inadequate. The fundamental idea is to increase the algorithm efficiency and distribute computing tasks to multiple devices if possible. Deep learning compression is an effective tool for boosting inference speed without conducting considerable modifications on original models. Besides deep learning compression, transferring computing works to multiple devices will significantly ease the central server's workload and require narrower transmission bandwidth. The extreme case is computing on the sensor side and abandoning gathering raw data in one or several places. However, the limitation of edge devices is relatively low computing power. Simplifying existing models, e.g., adopting a lightweight model framework and compressing the existing model, becomes necessary for edge computing. Thus, this chapter focuses on deploying deep learning compression and knowledge distillation to promote model efficiency on edge devices and keep the same accuracy as the original model. This chapter studies the example of parking monitoring as an ideal scenario for edge computing. More importantly, this chapter innovatively assesses how efficiency influences overall accuracy in parking monitoring through real-world data and simulation.

Chapter 4 Tiny Traffic Object Detection takes the crowd monitoring as the scenario and focuses on the problem of lacking tiny object data. Crowd monitoring is essential for traffic sensing as crowd gathering is common in urban regions, e.g., transit centers, and may raise severe safety disasters. Thus, accurately estimating crowd numbers and motion is meaningful.

Nevertheless, persons usually occupy a small portion of the image because of the cameras' installation position. Tiny-scale objects have much fewer training samples than normal-scale objects and deficient texture information. These practices make detecting small-scale persons difficult. Besides, tiny object detection can also contribute to object tracking and reidentification in public facilities and intersections because their inputs include object detection results. Inspired by the need and challenge, this chapter proposes an accurate and efficient crowd monitoring framework, including crowd counting and motion detection. It should be particularly emphasized that this chapter designs a Zoom-Stitcher to increase the tiny sample number without collecting and annotating new data. This chapter integrates individual detection and density estimation models to cope with the crowd counting in low- and high-density situations.

Chapter 5 Rare and Novel Traffic Object Detection explores a solution to detect uncommon and novel categories in transportation scenarios with a few annotated training samples. Rare object detection will contribute to traffic safety, especially for collision avoidance in autonomous vehicles and ADAS. It is unacceptable to classify rare objects in the background because they are less common in the training data. Although the core of missing rare objects is still lacking annotated data, improving rare object detection may not follow the strategy discussed in Chapter 4. One reason is that small-scale persons have similar textures while these different rare categories have different latent features. Another is that rare categories may only have several training samples, much fewer than small-scale persons. Therefore, this chapter proposes a few-shot training strategy and an effective and lightweight detection model to detect novel traffic objects and find corner-case data. This chapter conducts two experiments – general traffic object detection and traffic sign detection, to evaluate the

performance and significance of the proposed model and training strategies on rare and base categories.

Chapter 6 Pedestrian Detection in Adverse Conditions faces the challenge of poor data quality when the environmental light is dim, and the visible distance is short. Different from general computer vision in other applications, e.g., retail product classification, transportation applications do not have constant illumination. It is essential to have a robust detection algorithm in different illumination conditions for traffic monitoring. Improving detection accuracy in adverse conditions can help increase the safety level of autonomous vehicles. Since conventional RGB cameras can only export images with high noise and low texture information in low-light conditions, the involvement of infrared cameras can assist in providing available information by sensing targets' radiation. This chapter thus designs a multispectral object detection model for 24/7 pedestrian detection by dynamically adjusting the fusion weights of RGB and thermal embeddings. Besides improving detection accuracy, this chapter considers the model efficiency by selecting the lightweight model structure and deep learning compression techniques introduced in Chapter 3 for the self-driving platform with limited computing power.

2

Literature Review

2.1 Overview

This chapter will review recent progress in computer vision and deep learning related to transportation applications. This chapter has four primary sections covering fundamental computer vision algorithms, i.e., object detection, and specific training and optimization strategies, e.g., FSL and deep learning compression. Section 2.2 reviews milestones and current

progress of deep-learning-based computer vision algorithms. This section splits these algorithms into two categories – monovision and multispectral based on input sources. Section 2.3 introduces ITS applications utilizing computer vision technologies, e,g., traffic flow detection and crowd monitoring. The following two sections focus on data challenges of high data volume and insufficient annotated training data. Section 2.4 concerns dealing with massive traffic data more efficiently in analysis and transmission by compressing deep learning models in a low precision format. Additionally, this section introduces edge computing by distributing computing works to edge devices, which is suitable for large-scale traffic monitoring. Section 2.5 reviews research works on limited and imbalanced annotated data, including self-supervised learning, data augmentation, and FSL. More importantly, this section points out the major drawback of applying self-supervised learning in ITS applications and suggests adopting data augmentation and FSL.

2.2 Computer Vision and Deep Learning

Deep learning has achieved great success in multiple fields and surpassed conventional machine learning algorithms. Computer vision has benefitted from the development of deep learning with solid analysis and generalization ability. It will not be constrained by previous hand-crafted feature descriptors and uses learnable feature extractors. Therefore, this section will go through representative works in general computer vision algorithms, including conventional and deep learning algorithms, before diving into vision-based traffic sensing in Section 2.3. Although computer vision has many branches, this section concentrates on object detection algorithms as one fundamental task in computer vision. Besides, object detection also provides metadata to other computer vision tasks, e.g., multi-object tracking and

motion prediction. Thus, this section reviews object detection algorithms based on input sources – monovision and multispectral. Monovision object detection is the most common by applying a single camera. Multispectral object detection implies fusing RGB images with other spectral images. Since this dissertation works on vision-based traffic sensing, this section does not explore studies of fusing cameras and other sensors, i.e., LiDAR.

2.2.1 Monovision Object Detection

Traditional object detection methods involve two steps – hypothesis generation and hypothesis verification [275]. Haar [250] and Histogram of Oriented Gradients (HOG) [10] are two conventional feature descriptors to transform raw images to feature maps. Then the sliding window will traverse the whole image with a fixed stride. Each window location will be classified to generate a classification score. The main drawback lies in hand-crafted feature descriptors whose performance is constrained to detection targets. Initially, deep learning-based object detection models inherit this structure but replace the hand-crafted descriptors with neural networks for feature extraction. AlexNet [160] can be seen as the first widely-applied deep learning model for feature extraction. After this model, several milestone models have been proposed to increase the feature extraction ability as well as reducing the computing complexity, e.g., ResNet [110], MobileNet [118], and EfficientNet [282]. In ResNet, Kaiming He, etc., designed the residual block that increases the model depth significantly from less than 20 layers to more than 100 layers. The shortcut connection between layers helps solve the vanishing gradient problem, the foremost hurdle of increasing the model depth. In MobileNet, Andrew G. Howard, etc., replaced the traditional convolution operation with a separable convolution operation that dramatically reduces computing complexity. In EfficientNet, Mingxing

Tan and Quoc V. Le proposed a scaling method to scale dimensions of depth, width, and resolution uniformly. The above vanilla models only work for classification tasks.

The detection task requires extra structure, including regression and classification layers on different feature levels to predict bounding boxes and corresponding categories[97]. Object detection models have two categories – one-stage and two-stage based on the detection process. Two-stage models follow the same structure as the traditional detection methods. Representative two-stage models are the R-CNN series including R-CNN[96], Fast R-CNN[95] and Faster R-CNN[236]. Faster R-CNN conducts the region proposal on the feature map, while Fast R-CNN conducts the region proposal on the original image. In the deep learning era, the fully convolutional structure helps jointly combine the region proposal and prediction. Such kinds of models are called one-stage. Compared to two-stage detection models, one-stage models simultaneously predict the proposed region and corresponding classification scores without the region proposal step. Typical one-stage models include YOLO[234], SSD, and RetinaNet[181]. The first version of the YOLO model predicts objects on the last feature map, while SSD does this on multiple feature levels to cover more object scales. Meanwhile, the improved SSD called DSSD applied deconvolution to increase the model depth[83]. Another difference between YOLO and SSD is that SSD accepts inputs with fixed dimensions like 300×300, but YOLO accepts random multi-scale inputs. Two-stage models are generally more accurate than these one-stage models but have lower inference speed. To further promote one-stage models' accuracy, anchor-free models are designed[71]. The anchor-free models use keypoints to depict the objects instead of anchors. One representative work is FCOS[288] which proposed an effective method to suppress the low-quality predicted boxes by introducing a new concept called centerness, which measures the normalized distance from

the location to its responsible object center.

In practical scenarios, objects in the same image may different scales due to their actural sizes and distance to cameras. Previous works like YOLO and Faster R-CNN predict the last feature map with a large receptive field. The large receptive field is more likely to ignore tiny objects because tiny objects only occupy a small portion of the feature map. Therefore, it is necessary to conduct the prediction on multi-scale feature maps. Although SSD produces predictions on multiple feature maps, it does not perform well in tiny object detection because the shallow layer does not have strong feature representations. One straightforward solution is using image pyramid[140]. The traditional image pyramid is inefficient since the basic feature extraction process will repeat several times. Yanwei Pang, etc., designed an efficient image pyramid network by using a lightweight convolution block to generate feature maps from downsampled images[223]. Another solution is using the feature pyramid, and the typical work is Feature Pyramid Network (FPN)[180]. Enhanced by FPN, the average precision of Faster R-CNN has improved by around 20%. YOLO v3 builds a similar feature pyramid structure and predicts objects at three levels with a feature pyramid structure[235,24].

2.2.2 Multispectral Object Detection

Compared to monovision object detection stated in Section 2.2.1, multi-sensor fusion is applied to improve the performance in adverse conditions, i.e., low-visibility. One representative work is multispectral detection fusing RGB and thermal information. As described in Section 1.2.3, data collected in adverse weather conditions is poor with a lot of noise and low contrast when only using RGB cameras. Adding infrared cameras as another input source can enhance the sensing ability in low-visibility conditions. Multispectral detection can over-

come shortages of detection using either RGB or thermal images, which significantly benefits traffic sensing, e.g., pedestrian detection, and improves traffic safety. Before reviewing state-of-the-art detection algorithms, there are two public multispectral pedestrian datasets – OSU Color-Thermal Database[59], and KAIST Multispectral Pedestrian Dataset[126]. The OSU dataset uses a fixed camera to collect data, while the KAIST dataset uses the onboard camera commonly installed in connected and autonomous vehicles. The primary difference is whether the camera is fixed at one location. Thus, the frequent change of background and illumination almost disables the application of conventional object detection algorithms like background subtraction[329], and increases the difficulty of distinguishing the object from the background as well.

Multispectral pedestrian detection also has conventional and deep learning algorithms like monovision detection. The advantages of conventional algorithms include explainable structure and easy deployment. One traditional algorithm is combining the Aggregated Channel Features (ACF)[64] and thermal HOG features[57] for feature extraction and applying a classifier. In the KAIST dataset, Soonmin Hwang, etc., built the baseline with this algorithm and published its MR as 64.76%. MR is the shared evaluation metric in multispectral pedestrian detection and is defined as the ratio of false positives and all positives. Besides, they found that thermal images contribute less to instant pedestrian detection since the context information in thermal images is not as sufficient as RGB images. However, hand-crafted feature descriptors are out-of-date and limited to application scenarios, which cannot meet the demand for higher accuracy and robustness in transportation circumstances. Meanwhile, deep learning succeeded in object detection, as described in the previous section.

Based on fusion strategies, Multispectral detection algorithms generally have three fusion

branches – early fusion, middle fusion, and late fusion. Early fusion indicates stacking RGB and thermal images as the inputs. Middle fusion implies fusing feature maps from different sources within the feature extraction process[54]. Late fusion means combining the feature maps at the highest level. Previous works have revealed that late fusion is always more accurate than early fusion but it will bring extra computing costs. In Jörg's research, MR of early fusion was 10% higher than late fusion[294]. Daniel Konig, etc., built their detection model founded on Faster R-CNN and compared different fusion strategies[158]. Their experiment results showed the middle fusion performed very closely to late fusion, and their MR was about 9% lower than early fusion. Furthermore, Yongtao Zhao, etc., introduced the attention mechanism to the fusion process[351]. Instead of concatenating layers directly, they utilized local and global attended decoding for layer fusion and got an impressively low MR. Yunfan Chen, etc., fused RGB and thermal features at different levels for region proposal to improve detection performance on multiple scales[49].

Besides evaluating different fusion strategies, impacts of the illumination factor are also considered. Since the idea of fusing RGB and thermal images comes from reducing the effects of illumination, computing the mean value of two inputs at the fusion stage may not perform as well as applying dynamic weights based on current illumination conditions. Therefore, research works have also explored designing an Illumination-Aware (IA) network to compute fusion weights, notably[174]. Dayan Guan, etc., designed an IA network that uses the feature map from the concatenation layer of RGB and thermal streams[100]. Output weights of the IA network work for fusion in both classification and localization networks. Their experiment results displayed that applying an IA network reduces MR to 29.62%, which is 3% lower than no IA network. However, another factor of environmental temperature may also generate

impacts on the final results via affecting thermal images[55]. Only a few pieces of research consider this temperature factor in the fusion process.

Although detection accuracy is a vital evaluation metric for object detection, its efficiency is also meaningful in practice. As discussed in Section 1.2.1, cloud computing has suffered challenges for high data volume for analysis because of large-scale sensors. The key is reducing the model complexity. Most state-of-the-art multispectral detection models belong to two-stage concentrating on accuracy but sacrifice efficiency. Considering some mobile applications, e.g., pedestrian detection in autonomous vehicles, more efficient models are needed. As a result, Yali Hou, etc., built their multispectral pedestrian detection algorithm based on SSD and fused RGB and thermal embeddings at different feature scales[117]. Compared with other two-stage algorithms[100], its inference time to process a single image decreased to 0.03 seconds from 0.25 seconds by 88% on the same GPU instance. Their experiment demonstrated that it is possible to deploy multispectral detection algorithms on the edge side at a real-time speed. However, a considerable accuracy gap exists between this and other two-stage models with over 50% higher MR.

2.3 Computer Vision in ITS

Traffic sensing works as the frontend to collect and analyze traffic data using multiple sensors, e.g., loop detector, magnetic sensor, and camera. As stated in Section 1.1.2, cameras can capture more detailed information compared to other sensors. Additionally, the collected visual data is close to human observation and convenient for fast verification. The vision-based analysis also benefits from deep learning and has impressive performance in complicated transportation scenarios. In ITS, computer vision plays an increasingly vital role by

providing real-time traffic status information from stationary and mobile platforms. Previous works pay attention to road safety. As urbanization advances, ensuring crowd safety is another critical issue because of more opportunities for group gatherings, e.g., public facilities during events. Vision-based crowd monitoring is an effective tool to observe crowd safety and detect abnormalities in time. Thus, this section will review computer vision applications in the transportation field, such as object detection from the sky and crowd monitoring.

2.3.1 Object Detection for Traffic Sensing

Vision-based traffic detection can be generally classified into vehicle detection and pedestrian detection in public facilities based on detection objects. Vehicle detection also has multiple subsets according to the location of the camera: roadside, onboard, and aerial. The roadside camera is usually applied for traffic surveillance by transportation management department, e.g., vehicle recognition[214,35], automatic license plate recognition[166,41], vehicle logo recognition[222], and vehicle type recognition[298,153]. The onboard camera usually refers to the advanced driver-assistance system or autonomous vehicle[370,162]. Road lane and traffic sign detection are two fundamental tasks in these applications. The road lane detection is always related to image segmentation that marks each pixel to one class. The segmentation algorithm always consists of encoder and decoder[210]. Pingrong Chen, etc., proposed a lane marking detector, similar to SegNet[14], to group lane pixels. Then they applied a 3rd order polynomial to fit the lane. To apply higher-order polynomial to fit the lane, Davy Neven, etc., designed a separate neural network called H-Net to compute this transformation[215]. The traffic sign detection is similar to the vehicle and pedestrian detection, but it faces some unique challenges, e.g., multiple appearances of sign and motion artifacts[295]. Domen Tabernik and Dani-

jel Skočaj applied Mask RCNN [108] for accurate sign localization. There are several strategies to improve traffic sign detection, including Online Hard-Example Mining [260] and selecting the same number of Region of Interest (ROI) for each presented object. The popularity of UAV provides another view for vehicle detection [349,147]. The UAV has strong mobility and broader views than the roadside and onboard cameras. Seyed Majid Azimi proposed a ShuffleDet to monitor the open public space using UAV images [202]. However, the major drawback of UAV-based traffic flow estimation is its short operation time and it can only work for temporal monitoring. Based on the traffic flow estimation, traffic speed can also be estimated [77].

Besides vehicle detection, accurate pedestrian detection also makes sense to ensure transportation efficiency and safety. Based on the person number, it can be divided into single-person and crowd detection [26]. Single-person detection is similar to vehicle detection, and the major difference is detecting persons instead of vehicles. Since the output of deep learning models is not a binary classification, general video surveillance [276], autonomous vehicle [106,267], and UAV [3,188] may detect the person object at the same time depending on the design of the classifier. One emerging application related to COVID-19 is the social distance detection [114]. Onur Karaman, etc., built a smart camera system for social distance detection based on variants of Faster R-CNN and SSD [189,136] using bird's-eye view [145]. In addition, Zhenfeng Shao, etc., implemented the algorithm on UAV images for social distance detection in open space [252].

2.3.2 Vision-Based Crowd Monitoring

Urbanization creates more scenarios of crowd gathering, e.g., transit centers during peak hours. Crowd monitoring has been attracting considerable priority in non-motorized traf-

fic research due to the lessons learned from the previous crowd events and disasters[111]. The fundamental task in crowd monitoring is crowd counting. Crowd counting s two branches based on counting methods. The first one is detecting each individual and accumulating the total number. As a result, the overall performance largely depends on object detection accuracy. Deep learning methods have greater advantages than traditional methods using hand-crafted feature descriptors[211,313]. In addition, the object detection results are inputs for motion detection and prediction. However, this kind of method is constrained by whether the algorithm can extract each individual precisely. Occlusions are common in public facilities and decrease detection accuracy. When the crowd density is relatively high, and there are more occlusions, crowd counting following the abovementioned procedure will have a high estimation error.

To overcome the limitation raised by high crowd density, another method of predicting the density map thus becomes popular. Similar to semantic segmentation[217], density-estimation methods also utilize Fully Convolutional Network (FCN) like U-Net[242] to produce heatmap. The difference is that the output of each grid is the predicted crowd number instead of the classification scores. Inspired by FPN of predicting objects on multiple scales, one method is to enlarge the receptive field, e.g., applying convolutional filters with different kernel sizes in parallel for feature extraction[353]. In contrast, a larger kernel size leads to computing complexity. Dilated convolution can increase the receptive field while keeping the algorithm efficiency[177,352]. Besides upgrading convolution operation, Weizhe Liu, etc., used average pooling to get feature maps of different sizes[190]. What is more, other researchers upgraded the model framework instead of replacing operators by selecting more robust backbones and adopting the attention network[300,308]. Considering that the density distribution

may not be uniform in one scenario, researchers also divided the whole image into multiple patches for analysis. Deepak Babu Sam, etc., passed each patch into different networks to predict the crowd number based on the density score using a separable network called Switch-CNN[246]. Similarly, Usman Sajid, etc., proposed a zoom-in-and-out strategy to estimate the patch density[245]. However, density estimation methods usually have a higher error when the density is low, or crwods are sparse. It is necessary to combine individual detection and density estimation models for crowd counting to cope with all scenarios. Mingliang Xu, etc., combined object detection and density map estimation to get a more robust crowd counting method[326]. They selected the method based on the target distance and used image segmentation to estimate the depth map[72]. Then, they used the object detection method to count the near-view person number. One concern is that the near-view crowd may also have high crowd density, which is the opposite of this study's basic assumption.

Besides counting the crowd number, it is significant to obtain crowd motion characteristics, e.g., movement direction and speed. Previous research works had two kinds of methods to detect crowd motion – flow-based[356] and object-detection-based[137] methods. Flow-based motion detection mainly considers the crowd motion as the optical flow and estimates the flow based on anchors extracted with different feature descriptors. The quality of anchors can determine the motion detection accuracy. Several assistance methods have been applied to improve flow-based motion detection. Saad Ali and Mubarak Shah utilized various floor fields to calculate the movement probability and to force the crowd to follow specific rules[7]. Mikel Rodriguez, etc., employed Correlated Topic Model to describe the crowd behavior[241]. They computed the future movement via the observed historical trajectories and the predicted positives. Similarly, Louis Kratz, etc., adopted the hidden Markov model to consider

the spatial-temporal correlation[159]. Flow-based methods worked well in dense conditions because the algorithm can easily capture anchors for flow computation. However, these methods assume that the brightness is consistent, which may not be guaranteed in a practical scenario, especially in outdoor facilities. Additionally, these methods largely depend on the anchors. When the image contrast is low, hand-crafted descriptors may fail to capture the gradient change at the edge of objects. It is common for these descriptors to find anchors in the background.

Since flow-based methods may not suit low-density situations and cannot distinguish the background effectively, object-detection-based methods can make up for these drawbacks. Object detection-based methods first track each individual and then group individuals based on their movement patterns. As a result, the most critical part is multi-object tracking by finding the appearance similarity for the same target between adjacent frames. Chenghao Kuo, etc., designed an Online Learned Discriminative Appearance Model that used AdaBoost to select discriminative features to compute the link probability between tracklets[164]. Considering the tremendous success of the Kalman filter in various fields[206], Alex Bewley, etc., proposed a simple but effective tracking method, Simple Online and Realtime Tracking (SORT), that applied the Kalman filter to the object tracking task[20]. The data association was computed by intersection-over-union between detection and predictions. Compared to previous tracking methods, SORT offered state-of-the-art accuracy but a much higher inference speed in its publishing year of 2016. Much improvement works based on SORT, e.g., utilization of deep embedding features, have been conducted as well to reduce the impacts of occlusions[319,312,207]. After completing the multi-object tracking, the next step is grouping these tracking objects by their motion patterns. Riccardo Mazzon, etc., applied the social force

model to constrain the crowd behavior[205]. Feng Zhu, etc., built a hierarchical tree structure for grouping in terms of motion and location coherence[364]. What is more, Aniket Bera and Dinesh Manocha designed a hybrid motion model considering both flow and individual detection[19]. When comparing bounding boxes in object-detection-based methods with anchors in flow-based methods, each has pros and cons. Object-detection-based methods are more robust to the complicated visual background because they can learn the appropriate targets for tracking from training data. Flow-based methods can offer more reference points for crowd grouping.

2.4 Dealing with Massive Data

With more sensors installed in ITS, the data volume for analysis is much higher than ever. Meanwhile, the analysis algorithms are more complicated for different traffic sensing tasks. The existing system may not process data as efficiently as before to meet the growing demand for a better transportation system. How to ease the workload has become a hurdle for researchers and practitioners. There are generally two kinds of quick solutions. The first is accelerating the model inference speed. Instead of directly adopting a simpler model, deep learning compression can make users adopt the state-of-the-art but complex model while in a relatively small size. Thus, deep learning compression will quickly transfer a well-trained model to deployment without much framework modification. The second is edge computing which conducts computing on edge devices instead of transmitting data to servers for centralized processing. The edge device only sends metadata or results back to servers, saving bandwidth and computer resources on servers. Due to limited computing power, edge computing requires extremely high algorithm efficiency. Deep learning compression to help the

pre-trained model run smoothly on edge devices.

2.4.1 Deep Learning Compression

Simplifying deep learning models is a significant step before the practical deployment. As described in Section 2.2.1 and Section 2.2.2, selecting a lightweight model can improve the efficiency significantly. Besides this one, model compression is another effective tool to reduce the model size and boost the inference speed[53]. Several prevalent model compression methods include parameter quantization, pruning, and knowledge distillation. Parameter quantization is a recently developed and simple method to solve the problem that deep learning always consumes much GPU memory and computing resources[52]. The basic idea is to reduce the parameter precision. The most common quantization operation is replacing 32-bit floating-point parameters with 8-bit integer parameters[133], which can compress the model size by 75% in theory. Benoit Jacob, etc., conducted the experiment results and showed that a proper quantization operation could increase the inference speed significantly while having a slightly negative impact on accuracy. To further reduce the adverse effects, Yuhui Xu, etc., designed a multi-level quantization method by introducing incremental layer compensation to quantization layers iteratively[327]. More remarkably, researchers even studied binary quantization for extreme compression. Mohammad Rastegari, etc., proposed a binary CNN that saves memory by more than 32 times and increases the inference speed by 58 times compared to the original model[232]. Although binary CNN has a standing performance on saving memory and inference time, the accuracy drops by over 10%. Binary CNN is more suitable for applications that require minimal memory but are not sensitive to accuracy. Transportation applications cannot sacrifice much accuracy closely related to safety, which indicates a

carefully designed training strategy is necessary when adopting Binary CNN. The parameter quantization helps reduce the model size by decreasing the parameter precision. In addition to quantization, model pruning reduces connections between neurons by applying the mask at the training and inference stages. In the inference process, the forward process ignores the pruned connections and reduces computing complexity[284]. Michael H. Zhu and Suyog Gupta explored the efficacy of pruning[366]. They compared the performance of large-sparse and small-dense models and found that the overhead in sparse matrix storage diminishes the achievable compression ratio. The large-sparse model was consistently more accurate than the small-dense model. In addition, Qiangui Huang, etc., proposed a pruning agent to remove unnecessary convolutional filters in a data-driven way instead of following the pre-designed rules[122].

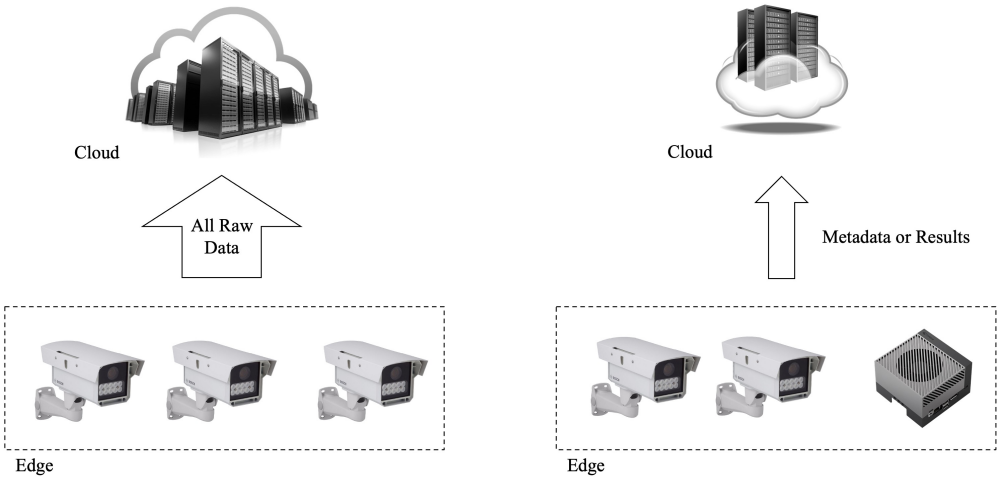
Deep learning quantization and pruning compress a large model into a smaller one. These compression techniques will cause some bias and lose a little accuracy. Retraining after compression becomes a necessity. However, the compressed small model has fewer parameters and is more likely to suffer an underfitting problem. Knowledge distillation came out to train a better small model,[98] and has gradually become a general framework that helps promote the model performance[303]. Knowledge distillation transfers knowledge from a teacher model, which is always a large model, to a target model by learning data distribution output via the softmax function[113]. Knowledge distillation targets to enhance the model generalization ability by learning from its teacher model. Based on knowledge types, there are three different categories. The first category is response-based knowledge referring to the response from the last output layer. Guobin Chen, etc., applied knowledge distillation in the object detection task, where the response included logits and offsets of bounding boxes[36]. The second

category is feature-based knowledge referring to representations from the last and intermediate layers. Sergey Zagoruyko and Nikos Komodakis proposed an attention map from original feature maps to represent embeddings[343]. The last category is relation-based knowledge which further explores the relationship between different layers. Junho Yim, etc., computed correlations between feature maps using the inner product as the knowledge[335].

2.4.2 Edge Computing

Deep learning has a powerful data processing ability because of its adequate transformations, e.g., CNN, and MLP, that put a higher requirement for computing power. As models become complicated with more parameters, they have a stronger fitting ability but become larger and more complicated at the same time.[290]. For example, ResNet can have 152 convolutional layers, which is much deeper than VGGNet with up to 19 layers[262]. Although deeper models and powerful servers can help process the raw data effectively, the rapid growth of sensors has raised a new challenge in analyzing massive data for ITS. Section 2.4.1 reviews works of compressing these deep learning models to promote the inference speed on the server. Some transportation applications have to process data on the edge side, e.g., self-driving and ADAS. The transmission latency and potentially unstable wireless connection are unacceptable. Edge computing is also a beneficial choice for roadside sensors with the fast-expanding sensor network[257,38]. Transportation agencies can save budget on upgrading the data transmission route and servers.

The framework comparison between cloud and edge computing is shown in Figure 2.1, where the left figure represents the cloud computing framework, and the right denotes the edge computing framework. In Figure 2.1, the width of the arrow between edge devices and



(a) Cloud Computing Framework **(b)** Edge Computing Framework

Figure 2.1: Framework Comparison between Cloud and Edge Computing

cloud refers to the data volume and corresponding bandwidth. The width of cloud computing is wider because raw data is transmitted back to the cloud. The scale of servers represents the computing resources required in the cloud. Comparing Figure 2.1a and Figure 2.1b, it can be found that edge computing asks narrower bandwidth and fewer servers. The edge devices are mainly cameras in this dissertation and can extend to other IoT sensors in different scenarios. Edge devices are only responsible for data collection and transmission for cloud computing without any computing task. In contrast, edge computing will conduct computing tasks on the edge side. Thus, edge devices have restrained computing power. When there is data for analysis with the development of urbanization, cloud computing demands more resources. Comparably, edge computing conducts the distributed data processing, where the computing power grows as more deployment of edge devices. In conclusion, there are three advantages of deploying edge computing in ITS: latency, scalability, and privacy.

Firstly, edge computing can shorten the **latency** significantly especially in the large-scale

network[74]. Kiryong Ha, etc., compared the latency when conducting speed and face recognition[102]. They found that the latency using cloud computing was several times higher. Since the raw data transmission is unnecessary, the data transmission time can thus save. Benefit from the algorithm compression[53], and hardware acceleration like TensorRT[208], the inference speed on edge can be as fast as the cloud. Secondly, edge computing has a strong **scalability**[249]. The node expansion can be more convenient without upgrading the cloud or central server because the edge device will conduct most computing tasks. In addition, transmitting all data back to the cloud is also inefficient in terms of internet utilization[38]. The video stream belongs to bandwidth-sensitive data in the transportation field, where edge computing can help save the bandwidth significantly. Lastly, **privacy** is attracting more and more concerns in the transportation field, e.g., facial and speech information. The cloud computing-based system will transmit all data back to the cloud for analysis and storage, which is riskier of being leaked in the process[231]. However, edge computing can guarantee privacy when the firewall on edge is appropriately set up because almost no sensitive data will be transmitted.

Although edge computing has various advantages, its development still has constraints due to its relatively weak computing power. Considering the practical situation, including cost, device size, and power consumption, deploying a computing machine as powerful as servers on edge is almost impossible. In particular, this dissertation will not discuss professional computing or AI chips though they are always powerful and efficient[240,70]. Those chips should have peripheral circuits and are task-specific. Instead, The general edge computing platform, e.g., Raspberry Pi and Nvidia Jetson, are concerned. There is an insurmountable gap between edge devices and servers. For example, one of the most potent edge devices in the market – Nvidia Jetson Xavier NX, whose integer performance is 21 Tera Operations Per

Second (TOPS) at 15W power consumption, is much weaker than the Nvidia A100 GPU, whose integer performance is 624 TOPS. For floating-point computing, the performance gap will become even more extensive. Previous experiment [121] showed some popular detection models, e.g., Faster R-CNN and SSD, could have an inference time of 110 and 50 milliseconds respectively to process one image. These models' backbone was Inception v2 [278] and input resolution was 300 $\times$ 300 pixels. The inference speed was not real-time even using a discrete GPU. The testing platform used the GPU of Nvidia Titan X GPU, whose integer performance is 26.4 TOPS. The inference time would increase enormously if these models were directly deployed on Nvidia Jetson Xavier NX without optimization.

Currently, edge computing is lively discussed in multiple fields, including ITS [175,178]. Yue Cao, etc., proposed a mobile edge computing-based system for the electric vehicle charging planning [30] via implementing data mining and aggregation in a decentralized way, which can reduce cloud processing. Similarly, Zhenyu Zhou, etc., developed a framework called robust mobile crowdsensing by integrating both data validation and local processing functions [363]. The proposed system filtered the irrelevant images on edge with deep learning and then transmitted the filtered images back to the server. In addition, edge computing was used in trajectory prediction [354], path planning, and autonomous control [80,212]. The roadside unit also uses edge computing that is responsible for communication with vehicles, data processing, and data uploading to servers [124,161,185]. Besides application in connected and autonomous vehicles, intelligent transportation facilities also benefit from deploying edge computing [86,81], e.g., parking garages [151,171]. One substantial component of a modern parking system is parking space monitoring. The current parking space monitoring is cloud-based and suffers challenges from limited computing resources and storage space [125], which are similar to other

cloud computing-based applications. Victor Kathan Sarker, etc., designed the parking system with IoT-based sensors using a vibration sensor and magnetometer[248]. Cong Zhang, etc., built the surveillance system with fish-eye cameras to cover a larger parking area in the parking system[28]. In addition to parking monitoring, other researchers explored traffic flow detection. Guanxiong Liu, etc., proposed a two-tier edge computing model for both congestion and speed estimation using the surveillance camera[183]. Firstly, they adopted the Global Foreground Modeling method[255] to extract potential candidates. Then, the Bayes classifier was applied to classify each pixel as either background or foreground.

2.5 Learning from Limited Annotated Data

Currently, most deep learning applications use supervised learning with well-annotated data for loss calculation. Although massive data is generated from different sources, annotated data always occupies a tiny part and is valid for deep learning model training. Once the training data is labeled, it is updated frequently, considering the budget and time consumption. Base categories in common scenarios can have sufficient annotated data from various public datasets. However, some categories have low appearance frequency and thus have fewer samples, which leads to data imbalance. In ITS applications, there are always minor categories that are less common, e.g., rare traffic signs. Missing these objects will lead to safety issues and potential violations of traffic rules. Deep learning models trained on such imbalanced datasets will have a considerable bias. They will tend to give a higher prediction score to those categories with more samples. Therefore, this section will review three effective strategies to face this insufficient annotated data challenge – data augmentation, self-supervised learning, and FSL. Data augmentation is the most common strategy in deep learning training to

increase data variance and improve model generalization. The general workflow of data augmentation is randomly doing operations, e.g., affine transformation, on existing data. Self-supervised learning may solve this challenge fundamentally by utilizing unlabeled data in the training process. Nevertheless, self-supervised learning requires a carefully designed training strategy and sufficient computing resources, making it not ideal for transportation applications. FSL relies on prior knowledge from base categories and forces the model to learn sufficient features from minor categories. More details will be discussed in the following part.

2.5.1 Data Augmentation

Deep learning has achieved incredible success by relying on the tremendous number of transformations that require sufficient training data. Without sufficient training data, there will be overfitting problems [156] that constrain the model performance on testing data. Additionally, the model generalization is an important attribute that refers to the ability to adapt appropriately to unseen data [216]. The poor generalization ability will lead to overfitting, whose phenomenon is that validation error keeps unchanged while the training loss decreases. There are several strategies to increase the generalization ability including dropout [270], batch normalization [132], transfer learning [317] and data augmentation [259]. Dropout is one regularization strategy that sets the activation values to zero of randomly selected neurons in the training process. In the inference process, the output of these neurons will be multiplied by the dropout probability. Dropout can force the model to learn more robust features using part of neurons. Batch normalization is another regularization strategy that standardizes input, accelerating the training process with a larger learning rate. In addition, transfer learning indicates first training the model on a large dataset and then fine-tuning it on a target dataset.

Transfer learning works well when datasets share low-level spatial characteristics. The foundation of transfer learning is using more data for model training. As a result, the most effective way is to increase the training data. Data augmentation does not require additional data annotation and can increase the target data size as one method to increase data variance. The fundamental assumption of data augmentation is that it can help extract more information.

Several widely-applied data augmentation strategies include flipping, rotation, cropping, translation, color space transformation, and noise injection. Among these strategies, rotation, cropping and translation may lead to label changing for detection tasks. Guoliang Kang, etc., proposed a Patchshuffle regularization [142] by randomly swapping pixel values within a sliding window. Hiroshi Inoue innovatively mixed two sample images by computing the mean value of each pixel [131]. The label of the new image is the same as the first sample image. To get a more generalized mixed-sample strategy, researchers applied a non-linear method by conducting the image concatenation vertically and horizontally at the same time [273]. Inspired by the dropout regularization, Zhun Zhong randomly erased part of images and filled that position with a random pixel value [361]. This strategy was especially for detection tasks to mimic the object occlusion. Meanwhile, some research works have also focused on evaluating the combination of these data augmentation strategies [285]. Ekin D. Cubuk, etc., designed a framework to help learn the data augmentation strategies from data [56].

This dissertation considers the category imbalance a challenge for deep learning, especially for tiny object detection. One straightforward method is just copying and pasting small objects multiple times within the images [155]. This method requires object masks to copy and paste objects precisely instead of coarse bounding boxes. Another way is conducting the image stitching by stitching four images into one image and then resizing to the original dimen-

sion[50]. Since the stitched images come from different scenarios, it may lead to the scenario inconsistency problem for objects existing at the boundary of images. What is more, Xuehui Yu, etc., found that a scale match between the pre-trained dataset and target dataset can be helpful[340]. They aligned the object scales between these datasets to utilize better existing labeled data.

2.5.2 Few-Shot Learning

Insufficient annotated data is a considerable challenge. Data augmentation makes excellent contributions to increasing data variance but cannot generate new annotated data. Thus, researchers have started to work on utilizing the unlabeled data with pseudo labels. Those methods are called self-supervised learning[193]. Self-supervised learning does not need ground truth as supervision compared to supervised learning. With unlimited unlabeled data, self-supervised learning can help train a robust backbone for downstream tasks and reduce the parameter number trained in the fine-tuning stage. In general, self-supervised learning has generative and discriminative algorithms. Generative algorithms target to generate or predict pixels in the input space[44,63]. Discriminative algorithms learn representations using functions similar to functions in supervised learning but perform pretext tasks. Contrastive learning, as one of the discriminative algorithms, achieves promising results[13]. Ting Chen, etc., proposed an asymmetry framework for contrastive learning called SimCLR and its successor[42,43]. In the first version of SimCLR[42], they found that a more substantial data augmentation can benefit the model training. A nonlinear transformation between the representation and contrastive loss could substantially improve the representation quality. In the improving version[43], MLP replaced the single nonlinear transformation for better performance. Big-

ger self-supervised models were more label-efficient and performed significantly better when fine-tuned on small datasets[107,45]. To reduce the batch size, BYOL is proposed by applying an extra linear layer on the online encoder to predict the representation and comparing it with the representation generated from the target encoder[99]. According to the experiment setting of these self-supervised learning or contrastive learning, the required computing resources are thrilled because they should go through a super larger dataset and ask for a large batch size to obtain a globally optimal value. For example, the relatively efficient contrastive learning framework BYOL still requires a batch size of over 500 with 64 cloud TPUs. Thus, it is almost impossible to deploy self-supervised learning in transportation applications.

Comparably, FSL only works on a few annotated samples and is easily implemented with limited training resources. As the name suggests, FSL makes machine learning algorithms learn features and representations from limited training samples. General machine learning requires a vast amount of balanced training data. It becomes difficult to apply machine learning algorithms in practice when involving novel categories[1,146]. When implementing FSL to improve the training process, the dataset will have two parts – base D_{base} and novel D_{novel} categories that have no overlapping. Base categories D_{base} have sufficient training data, and novel categories D_{novel} only have a few training samples. There are k samples for each novel category in a K-shot object detection task. If training on the novel categories directly, there is an overfitting problem leading to poor generalization[37]. Thus, FSL always adopts a two-step training strategy. The algorithm is firstly trained on base categories D_{base} with sufficient data and then fine-tuned on a balanced small dataset with novel categories[157]. Many few-shot object detection algorithms utilize meta-learning to learn how to predict novel categories. Bingyi Kang, etc., added a reweighting module on YOLO[139], and Xiaopeng Yan, etc., designed a similar

module named predictor-head remodeling network for Faster R-CNN [328] for few-shot object detection. The shared feature was the meta learner with a small number of support images and their bounding boxes. To simplify inputs without bounding boxes, Yu-Xiong Wang, etc., proposed a weight prediction meta-model that predicts category-agnostic parameters from base categories and category-specific parameters from novel categories [309]. Xin Wang, etc., further improved this process by removing external networks and introducing a two-stage fine-tuning approach (TFA) [304]. Based on their experiment results, the performance in novel categories was even better. As mentioned above, a few research worked on few-show traffic object detection. Anay Majee, etc., applied FSL in traffic object detection [201]. They adopted Faster R-CNN to work as the base detection algorithm. Then they evaluated two few-shot architectures – feature-similarity-based following Xin Wang's work and auxiliary-network-based following Yang Xiao's work [324]. Their experiment further demonstrated the outperformance of TFA. Faster R-CNN belongs to a two-stage detection algorithm and cannot meet the efficiency requirement in traffic object detection deployed on roadside units and autonomous vehicles.

2.6 Chapter Conclusion

This chapter has reviewed state-of-the-art computer vision works in ITS, including traffic object detection algorithms, deep learning compression, data augmentation, and FSL. Compared to general computer vision tasks in other fields, e.g., retailed product classification [268], and short-clip classification [78], transportation applications have its own specialty for computer vision algorithms. Traffic sensing has a higher requirement of robustness and efficiency in all-day conditions to monitor and ensure traffic safety. With the development of urban-

ization, more traffic sensors have been deployed and generate an extremely high data volume. Thus, the analysis algorithms should be more efficient and process incoming data as soon as possible. IoT technologies contribute to the broader deployment of sensors with smaller sizes and good communication abilities. Advanced processors enable edge devices with some but limited computing capabilities. Edge computing has become popular in ITS to distribute computing tasks on central servers and promote system efficiency. The direct transfer of computer vision models from servers to edge devices may not work well because of the different computing power. Appropriate algorithm optimization is necessary to let edge devices run deep learning algorithms. In conclusion, this chapter has summarized research works faced with high volume, lack of annotations, and poor quality challenges. More importantly, this chapter points out the limitations of existing works in dealing with these challenges. The coming four chapters will propose innovative and practical machine learning methods for challenging traffic video sensing applications.

3

Edge Computing for Traffic Sensing in ITS

3.1 Overview

This chapter focuses on efficiently processing the rapidly growing traffic data and reducing heavy workloads on central servers. One straightforward solution is improving the machine learning algorithm efficiency by implementing model compression. After the model compression, deploying the compressed algorithm on the sensor side for edge computing becomes

possible because computing power on the edge side is limited. Edge computing can fundamentally ease the cloud's workloads. This chapter proposes an optimization workflow to improve the algorithm efficiency without losing accuracy. To demonstrate the effectiveness of the proposed workflow, this chapter studies the parking monitoring application to reduce the search time for available parking spaces. Since parking garages have a higher capacity in metropolitan regions and transit centers, edge computing is better than cloud computing for camera data analysis by saving transmission bandwidth and server resources.

3.1.1 Background

The city development has promoted the construction and upgrade of transportation facilities, including roadways and parking garages. The higher urban population has raised more concerns about traffic congestion and crowd safety problems. It is significant to monitor these regions to ensure traffic efficiency and safety. Nowadays, IoT-based traffic sensors can be easily deployed for multiple purposes, e.g., vehicle counting and collision avoidance. These sensors provide real-time data to the transportation system and enhance its sensing ability. Transportation agencies can make accurate traffic predictions and optimized operations based on received and analyzed data. Commuters can learn in-time traffic information and adjust their travel plans. However, such data also brings high pressure on transmission bandwidth, computing resources, and storage capacities. With the development of technologies, ITS will have more sensors and data in the foreseeable future. Additionally, deep learning algorithms in ITS also make the situation worse. Though deep learning is far more potent than conventional machine learning and statistical methods, its structure is complicated and requires specific hardware acceleration. As discussed in Section 1.2.1, cloud computing-

based framework with deep learning algorithms has a hard time catching up with the rising data volume and sensing demands. Therefore, it is necessary to improve the system efficiency to accommodate more input data and processing tasks.

Generally, there are two kinds of solutions. The first one is improving the efficiency of deep learning algorithms via different compression methods. Deep learning compression can only shorten inference time but cannot reduce the transmission latency. The second one is edge computing by distributing computing tasks to edge devices, usually integrated with sensors. As reviewed in Section 2.4.2, edge computing has advantages in latency, scalability, and privacy over cloud computing. Edge computing processes data on the sensor side and only transmits metadata or results to servers for deeper analysis and storage, saving transmission bandwidth. Since some locations have poor network connections, edge computing can contribute to deploying ITS in those areas. Each edge computing device is relatively independent and can flexibly work together to form a large-scale network.

3.1.2 Application Scenario – Smart Parking Monitoring

The rapid vehicle population growth worldwide has caused severe traffic congestion, especially in urban areas. Urbanization increases travel distance for daily commuting. Commuters may prefer to drive to workplaces or transit centers, which increases the demand for available parking spaces[15,11]. Constructing the large parking garages and upgrading the parking management system become necessary to accommodate the incremental demands[331,154]. Considering the scarce land for parking usage in urban areas[254], especially in central business districts of cities, applying advanced parking management strategies, e.g., dynamic pricing and shared parking, gains more feasibility for implementations. The advanced parking man-

agement system can also help drivers to find available spaces in a large parking garage in a short time and indirectly ease traffic jams. The collected historical parking information can help parking availability prediction and optimize parking management strategies. Basically, such a system requires real-time parking availability information [316].

Currently, multiple sensing technologies are available for collecting real-time parking occupancy data [239], e.g., ultrasonic sensing [170], magnetic sensing [261], and vision-based sensing [196]. Compared with video-based sensing technologies, ultrasonic and magnetic sensors rely on detecting physical status changes and do not require complex analysis algorithms. They are less likely affected by weather and light conditions. However, each parking slot should have at least one sensor. The parking space for large vehicles, e.g., trucks, even needs two and more sensors to ensure detection accuracy and avoid missing parking events [330]. Applying magnetic or ultrasonic sensors indicates higher costs of hardware devices, installation, and maintenance [220]. Magnetic sensors are easily impacted by surrounding magnetic disturbance. In contrast, one camera can simultaneously monitor multiple parking slots. Thus, the vision-based parking space monitoring system, benefitting from powerful machine learning detection algorithms, is the most popular.

Since there is still a technical gap between the growing parking demand and insufficient parking spaces, this chapter works to design a dedicated space availability detection framework based on machine learning and deep learning. With the popularity of deep learning, vision-based detection algorithms for parking occupancy transfer from traditional computer vision algorithms, e.g., support vector machine with Harris-corner feature [291], to deep learning [9]. Consequently, the additional complexity of deep learning results in excessively high computational costs, which certainly hurts the real-time performance of the parking occu-

pancy detection. A powerful central server or a cloud platform is in charge of vision data processing tasks to address this concern. Even though one camera can detect multiple parking slots, large parking garages still need many cameras to ensure coverage. Cloud-based parking monitoring systems thus require broad data communication bandwidth and sufficient computing resources. It leads to an unacceptable cost of central servers and communication services. Thus, merely utilizing the computing power of edge devices to operate parking occupancy detection algorithms in real-time is undoubtedly in demand[257].

When implementing the deep learning-based detection algorithm on edge devices, the limitation is the computing power of edge devices. General detection algorithms will be too slow when transferring the algorithm directly to the edge side. The exacerbated inference speed will highly impact parking occupancy detection accuracy. For example, the surveillance system fails to traverse all parking slots and will miss parking status changes when several parking vehicles arrive and depart within a short period. In this situation, the detection algorithm efficiency is significant. With the implementation of higher-resolution and wider-angle cameras, those challenges negatively influence the validity of collected parking occupancy data.

3.1.3 Contributions and Organization

This chapter designs a deep learning algorithm optimization pipeline for transportation applications, especially on edge platforms. The proposed optimization pipeline can be easily transferred to other transportation applications requiring edge computing. According to previous studies mentioned in Section 2.4.1, model quantization is the most straightforward method to decrease complexity without modifying the model structure. Knowledge distillation can compensate for the accuracy loss after model quantization[225]. This chapter selects

automatic parking monitoring as the application scenario. Inspired by noticeable progress in deep learning and edge computing, the main research objective is to improve the inference speed and accuracy of the parking availability monitoring algorithm on edge devices based on model quantization and knowledge distillation. The primary contributions of this chapter are summarized in five points.

a) Design a deep learning model optimization pipeline to reduce algorithm complexity and maintain accuracy for edge computing in transportation applications. The pipeline is evaluated by optimizing the intelligent parking monitoring system's vision-based parking availability detection algorithm.

b) Develop an edge-based parking surveillance system, based on the improved parking availability detection algorithm, targeting two kinds of parking information – real-time occupancy of the entire parking area and parking duration of each parked vehicle.

c) Propose a deep neural network based on MobileNet for parking space status classification optimized by model quantization and knowledge distillation. Compared with other state-of-the-art algorithms on two public parking datasets, the proposed algorithm shows promising efficiency and accuracy.

d) Build a real-world parking dataset to evaluate the effectiveness of the proposed system in large-scale parking garage monitoring. According to experiment results, the proposed system certainly improves the accuracy of the target parameters due to higher inference speed compared with other methods.

e) Conduct a simulation to imitate fast parking status change situations to assess how the efficiency of parking status classification affects parking time duration and garage

occupancy calculation accuracy. This simulation supplements previous real-world experiments, considering long parking events.

The rest of the chapter is organized as follows. The organization of the remaining part is as follows. Section 3.2 firstly proposes the deep learning optimization pipeline as the foundation of the whole dissertation. Then two optimization methods applied in this pipeline are introduced – model quantization in Section 3.2.2 and knowledge distillation in Section 3.2.3. Based on the proposed pipeline, this section describes the framework of the parking monitoring system and the parking status classification model. Section 3.3 presents the experiment setting and experiment results to evaluate the efficiency and accuracy of the parking monitoring system. The experiments are conducted on public and self-collected datasets. Moreover, this chapter also designs and runs a simulation in Section 3.3.5 to assess how traversal speed affects the overall detection accuracy.

3.2 Methodology

3.2.1 Deep Learning Optimization Pipeline

This chapter targets to provide a deep learning optimization pipeline for transportation applications. The construction of ITS promotes broader applications of machine learning algorithms for various tasks, including traffic sensing, prediction, and operation. The cloud-based data processing structure may not catch up with the growing demand for analyzing data simultaneously. Edge computing can distribute computing tasks to edge devices and has three advantages over cloud computing – short latency, strong scalability, and high privacy. Nevertheless, edge devices have poorer computing power, restrained by their sizes and power

supply. Machine learning algorithms running on servers cannot transfer to edge devices directly without optimization. Therefore, a deep learning optimization pipeline for edge computing is necessary. The pipeline is shown in Figure 3.1, including three steps – pre-training, compression and post-training. In this chapter, the last two steps are emphasized. The pre-training refers to the general machine learning training process, e.g., supervised learning. After obtaining a well-trained model, the next step is compressing the model size and increasing model efficiency. In this chapter, the applied compression method is model quantization discussed in Section 3.2.2. Since the compression operation will decrease accuracy, it is necessary to compensate for the accuracy drop via post-training. Post-training can follows the training strategy in the first step. However, the compressed model has a weaker fitting ability, which indicates conventional training strategies may work as well as before. This chapter thus adopts knowledge distillation, described in Section 3.2.3, to force the compressed model to learn from data and the uncompressed model. More importantly, this pipeline is suitable for most transportation applications requiring machine learning on resource-constrained platforms.

Figure 3.1: Deep Learning Optimization Pipeline

3.2.2 Deep Learning Model Compression

According to Figure 3.1, deep learning model compression plays an essential role. Compared to simplifying the model framework and adopting more efficient operators, deep learning compression belong to plug-and-play methods. Section 2.4.1 has reviewed current com-

pression technologies. Model pruning and quantization are the two most popular methods. Model pruning requires extra searching time to determine which connection is pruned. Model quantization is more straightforward and only reduces the model precision. Therefore, this chapter adopts model quantization by reducing the parameter precision from 32-bit floating-point to 8-bit integer. The compressed model size becomes approximately $1/4$ of the original model.

Before quantization, the first step is fusing layers as suggested in Zhewei Yao's research [332], such as combining convolutional and batch norm layers. This layer fusion can improve the inference speed by reducing the computing complexity by simplifying two-step work to one-step work [8]. Based on their experiment results, the inference speed of the deep neural network increases by 15.8% after layer fusion. After layer fusion, the second step is parameter quantization. The general framework of the quantized neural network is shown Figure 3.2, including quantization and dequantization. In Figure 3.2, "Quant" and "Dequant" refer to quantization and dequantization modules, respectively. The input and output are still in their original precision, e.g., 32-bit floating-point, which are convenient for data loading, loss computing, and evaluation. All computation between "Quant" and "Dequant" uses lower precision, e.g., 8-bit integer. The input converts into the integer precision at the quantization module "Quant". The output transforms back to floating-point precision from integer precision at the dequantization module "Dequant".

There are two kinds of quantization – per tensor and channel asymmetric linear quantization. Per tensor indicates that all values within the tensor are scaled in the same way. And per channel implies that all values within the channel are scaled in the same way. The mapping function to convert floating tensors to 8-bit signed integer is shown as $Q(x) = \lfloor x/s + z \rfloor$,

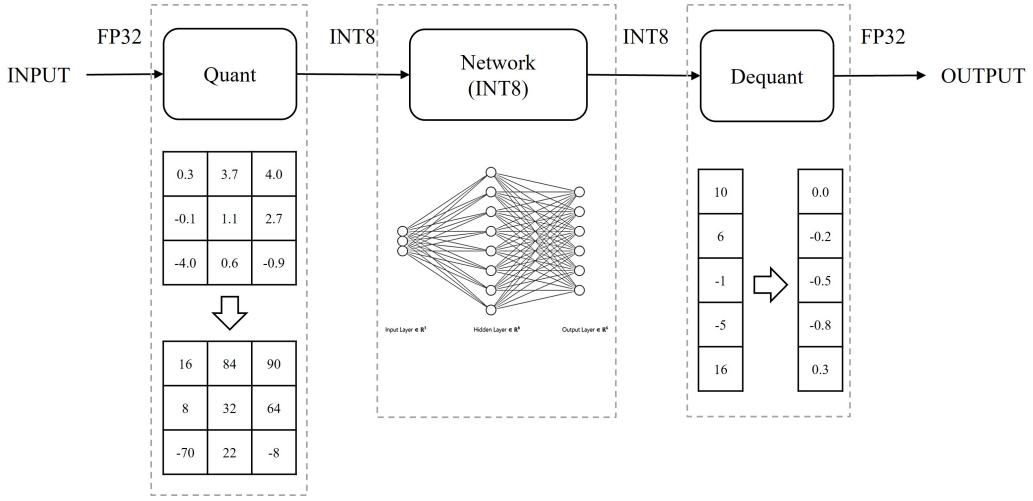


Figure 3.2: Quantization and Dequantization Framework

where x is the input tensor, s is the scale, and z is the zero point. Considering that the weights in floating-point precision are centering at zero, this quantization operation will not lead to much accuracy loss[287]. The post-training strategy involves knowledge distillation that is dispicted in the coming section.

3.2.3 Knowledge Distillation

Post-training is a fundamental step in keeping accuracy. It is unavoidable to lose some accuracy in deep learning model compression. The next step is to train the compressed model again to have similar accuracy to the original model. Training the compressed model on the training dataset cannot make the model learn as much knowledge as the larger model because of a poorer data fitting ability. As reviewed in Section 2.4.1, knowledge distillation is an effective tool to cope with this situation. The basic idea of knowledge distillation is to let the student model learn how the teacher performs on the target dataset instead of directing learning from data. The knowledge distillation framework is shown in Figure 3.3, where

"Soft Prediction" refers to the soft prediction defined as the output after the softmax activation function. "Hard Prediction" refers to the hard prediction defined as the direct output from the network's last layer. According to Figure 3.3, the teacher model is a large and reference model, which can be the original model or others. In this chapter, the teacher net is the original model.

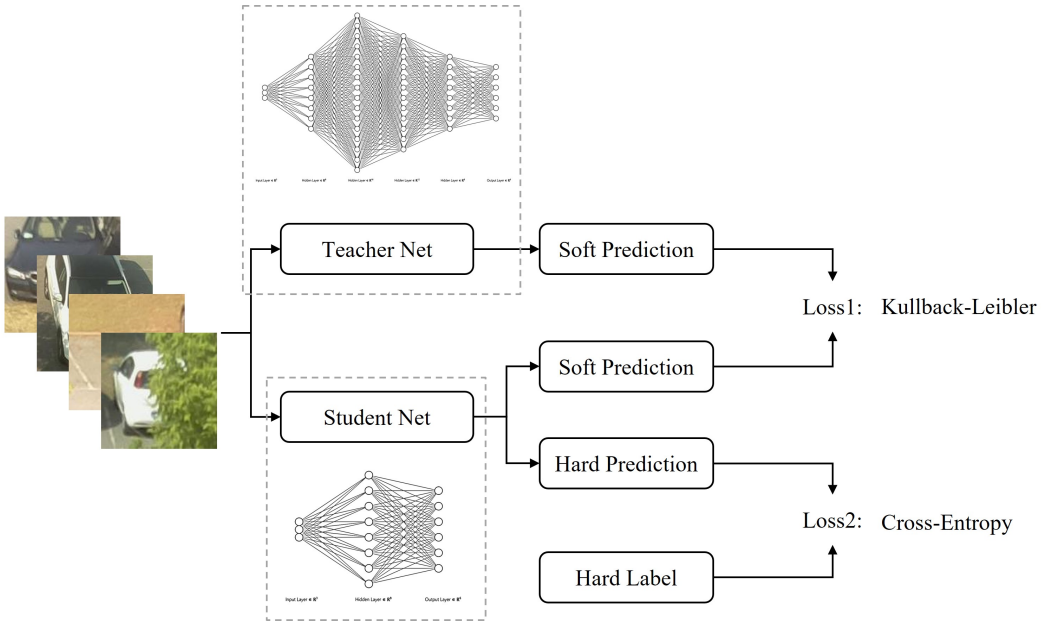


Figure 3.3: Knowledge Distillation Framework

Since knowledge distillation forces the student net to learn from the ground truth and teacher net, the loss function has two parts. The first loss function, written as $L1$, is the distillation loss and uses Kullback-Leibler divergence to measure the distance between predictions from the teacher and student models. $L1$ function is defined in Equation 3.1, where N indicates the size of mini-batch. y_i is the output from the teacher model, and x is the output from the student model. The second loss function, written as $L2$, is the cross-entropy loss to compute the difference between the student model's prediction and ground truth. $L2$ is

defined in Equation 3.2, where *cls* is the hard label. The overall loss is the sum of above loss function, $L1$ and $L2$, as $L = L1 + L2$.

$$L1(x, y) = \frac{1}{N} \sum_i y_i (\log(y_i) - x_i) \tag{3.1}$$

$$L2(x, cls) = -x_{cls} + \log(\sum_i \exp(x_i)) \tag{3.2}$$

3.2.4 Parking Monitoring System Framework

This chapter studies parking monitoring as an application scenario for the proposed optimization pipeline, and introduces an edge computing-enabled parking monitoring system. The system framework is shown in Figure 3.4. According to Figure 3.4, there are two steps for extracting real-time parking occupancy information from image sequences. The first step is to iteratively traverse all parking spaces and identify the parking status of each predefined parking slot. The parking status identification model is a classification model that classifies the image patches of all predefined parking slots into binary categories – empty and occupied. This classification model is built upon MobileNet by trading off accuracy and efficiency, whose framework will be described in Section 3.2.5. Then this chapter applies the proposed pipeline in Section 3.2.1 to optimize the parking status identification model. Based on the parking status classification results, the next step is to update two significant parking parameters in real-time, occupancy of the entire parking area and parking duration of each parking vehicle. The algorithm of the second step is presented in Algorithm 1 and corresponds to the right block in Figure 3.4. This system will update parking information after traversing all

spaces once. If the inference time of detecting the single slot is long, the whole latency will be high. Based on Algorithm 1, longer latency may lead to missing parking space change. For example, one vehicle leaves the slot, and another vehicle enters it in a short period before the system completes one traversal. This system will miss the parking space status change and calculates incorrect parking duration.

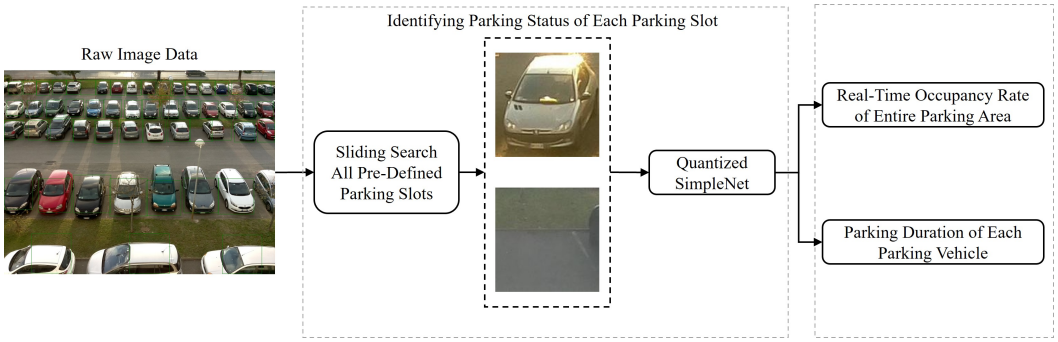


Figure 3.4: Framework of Parking Surveillance System

3.2.5 Parking Status Identification Model Framework

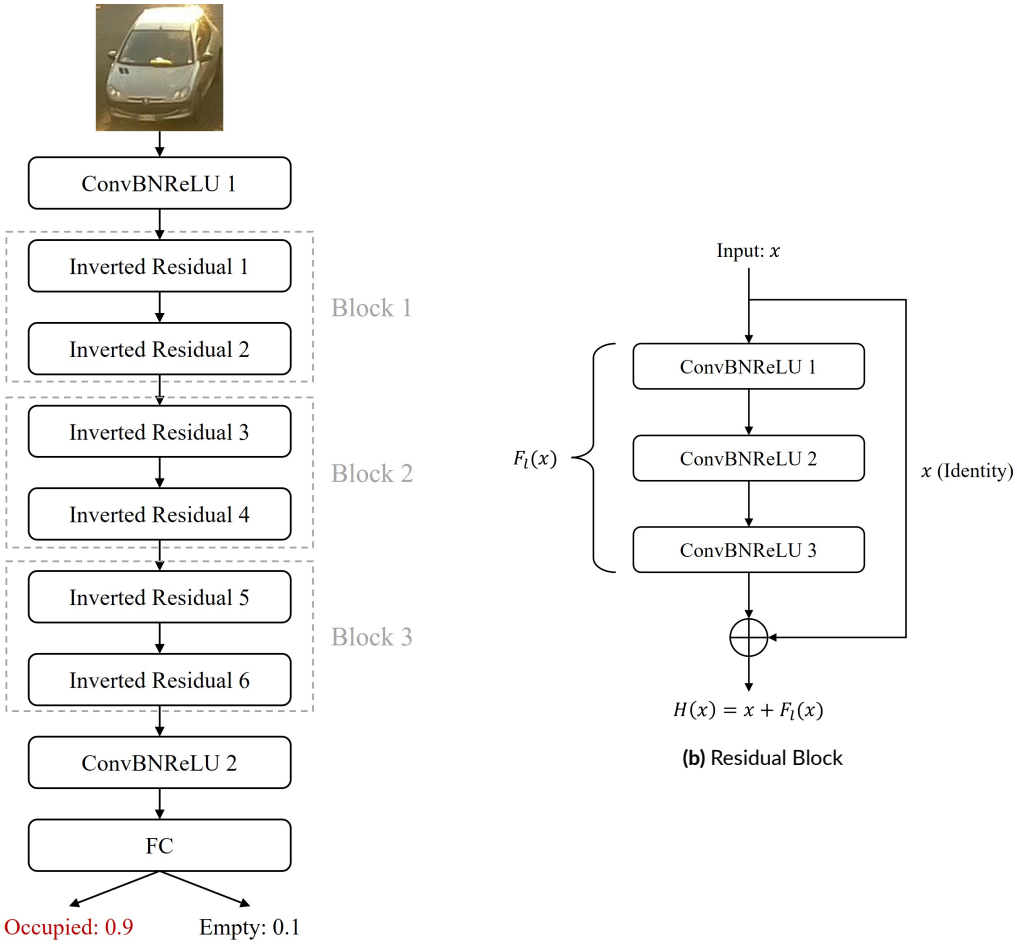
MobileNet appeared in 2017 with the highlights of depthwise separable convolution. Its improving version developed with inverted residual blocks and linear bottleneck[247]. In this chapter, MobileNet v2 works as the base model for parking status classification. Although MobileNet is very successful in improving efficiency, it is still overparameterized for parking status identification. Thus, this chapter proposes a simplified CNN based on MobileNet v2 called SimpleNet by removing some redundant layers. Compared with MobileNet v2, the total layer number of SimpleNet has reduced by 60%, whose inference speed expects to be one time faster than MobileNet v2 theoretically. The overall framework of SimpleNet is shown in Figure 3.5a, where "ConvBNReLU" and "Inverted Residual" blocks are inheriting from

Algorithm 1: Workflow of Updating Occupancy and Parking Duration

All predefined parking slots $P = (p_1, ..., p_n)$;
Parking duration of each parking slot $D = (d_1, ..., d_n)$;
Occupancy of entire parking area O_t;
The timestamp of last iteration $T = (p_1^t, ..., p_n^t)$;
for *i in n predefined parking slots in the image at time* p_i^{t+1} **do**
    Identify parking status s_{t+1}^i of parking slot i;
    if *current parking status* s_{t+1}^n *is occupied* **then**
        Update parking duration of slot i by $d_i += p_i^{t+1} - p_i^t$;
        if *previous state* s_t *is empty* **then**
            Update occupancy $O_{t+1} = O_t + 1$;
        end
    end
    else
        if *previous state* s_t *is occupied* **then**
            Update occupancy $O_{t+1} = O_t - 1$;
            Output: parking duration of slot d_i
            Reset the parking duration of slot i as $d_i = 0$;
        end
    end
end

MobileNet v2 with the same structure. "ConvBNReLU" block indicates a linear sequential convolutional, batch normalization, and ReLU layers. "Inverted Residual" block composes several "ConvBNReLU" blocks with separable convolution and residual connection.

The performance of residual blocks can reduce the negative impact of model degradation impressively [110] by the shortcut connection. The model can easily contain more than 50 layers with residual blocks and significantly improve detection accuracy. Compared with VGGNet model [262], the ResNet model reduces the error rate by 6%. The basic function of residual block is defined as $H(x) = x + F_l(x)$, where x is the input and $H(\cdot)$ is the output of this block. $F_l(\cdot)$ indicates the mainstream network. The shortcut connection can help the passthrough



(a) Overall Framework

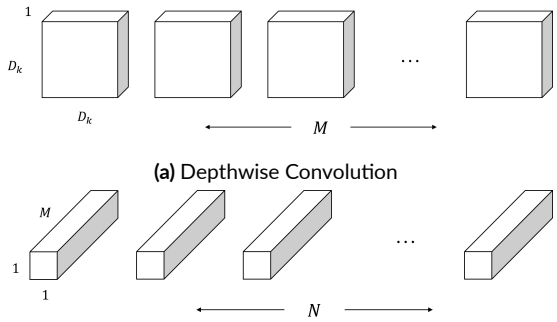
(b) Residual Block

Figure 3.5: SimpleNet Framework

of gradients effectively. The residual block in MobileNet v2 is shown in Figure 3.5b, where "Conv1" and "Conv2" are 1×1 convolutional layers, and "Conv2" is a 3×3 convolutional layer.

One key contribution of MobileNet is proposing the separable convolution, which decomposes the traditional convolution into two parts – depthwise convolution and pointwise convolution. The separable convolution is shown in Figure 3.6, where D_k is the con-

volutional kernel size. M is the input channel number, and N is the output channel number. Compared with a traditional convolution operation using the same kernel size D_k, the computation complexity of the new convolution is only $1/(D_k^2)$. Usually, the 3×3 kernel can decrease almost 90% complexity. Another important contribution is applying residual blocks.



(a) Depthwise Convolution

(b) Pointwise Convolution

Figure 3.6: Depthwise Separable Convolution

3.3 Experiments

3.3.1 Experiment Setting

This chapter has conducted experiments in both public datasets and a self-collected dataset to evaluate the algorithm proposed in Section 3.2. CNRPark[9] and PKLot[60] datasets are used for the experiment. The CNRPark dataset consists of 12,584 labeled parking spaces under different weather and occlusion conditions. The original image size is 1000×750, and the image of each slot is cropped from the original image and then resized to 112×112. The PKLot dataset consists of 695,899 labeled parking spaces from two parking lots under different weather conditions. The original image size is 1280×720. Similarly, the image of

each slot is firstly cropped and then resized to 56×56. An image covering all slots and two image patches of a single space is shown in Figure 3.7. Different from object detection, parking space monitoring works on classifying pre-defined locations. The primary reason is that object detection cannot guarantee all parking slots have been checked. The detected parking space number will not be stable. In addition, object detection may not accurately locate parking slots. The intra-class variance of empty spaces is too large because there are different kinds of backgrounds. This high variance will make it hard for the machine to learn valuable features. Object detection is more easily affected by occlusion and thus produces false negatives. The left part of Figure 3.7, all target parking slots are marked by green bounding boxes. The right shows image patches of occupied and empty spaces.

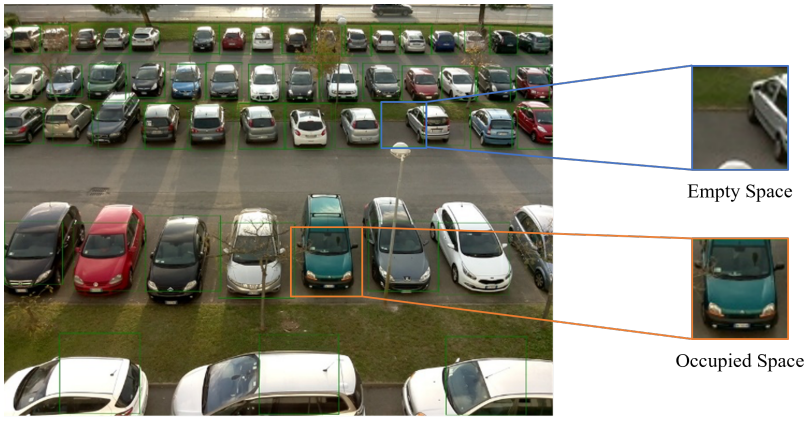


Figure 3.7: Sample Image from CNRPark Dataset[9]

The data distributions of occupied and empty slots are presented in Table 3.1. The status distribution under different weather conditions, e.g., sunny, rainy, and cloudy, is also listed. According to Table 3.1, the sample number of occupied and empty spaces is almost the same, which implies that there is no data imbalance problem. CNRPark and PKLot datasets merge into one dataset for training and validation in this chapter. Considering the limited samples,

images are divided into four parts equally for cross-validation. The training set size is 630,646 and the validation set size is 210,216.

Table 3.1: Parking Datasets Information

Datasets	Occupied	Empty	Total
CNRPark	79,307	65,658	144,965
Sunny	37,513	25,665	63,178
Cloudy	23,176	21,067	44,243
Rainy	18,618	18,926	37,544
PKLot	337,780	358,117	695,897
Sunny	186,511	180,312	366,823
Cloudy	87,735	141,396	229,131
Rainy	63,534	36,409	99,943

Besides the public datasets for evaluation, this chapter also builds its own dataset by capturing data at the parking lot at University Village, Seattle, US. The primary motivation is the long interval between two adjacent images in public datasets. The previous two datasets can only provide static images with a interval of more than 5 minutes. It is hard to observe inference speed's impact on parking occupancy and parking time computation accuracy. Therefore, this chapter further conducts experiments on evaluating the impacts using continuous videos. A snapshot from the video is shown in Figure 3.8. The collection time is two afternoons on July 14th and July 21st, 2020, when this plaza is bustling, and most parking slots are occupied. In total, 15 video clips are captured. The video resolution is 1920×1080 and frame rate is 30 Frames Per Second (FPS). The parking status of each slot is labeled manually. Following rules in the CNRPark dataset, the image patches of each parking slot are resized to 112×112.

The training platform is a PC with GPU of Titan XP to acclerate the training process. Since one major highlight is designing and optimizing a parking space monitoring system for

Figure 3.8: Sample Image at University Village, Seattle

edge devices. The proposed system is evaluated on both cloud and edge platforms. There are two testing platforms. One is the same as the training platform and acts as a cloud computing device. Another is Raspberry Pi 3 with a 1.4GHz quad-core CPU, which acts as an edge device with limited computing power. Although other general edge platforms and accelerators are available to boost inference speed, they will bring much more extra power consumption. For example, the peak power of Raspberry Pi 3 is 5W, while another popular platform Nvidia Jetson Nano consumes 10W. Besides, other sensors, e.g., cameras, connected to the same power supply will put extra pressure on the power supply and limit the application scenarios. Thus, Raspberry Pi is an ideal platform for evaluating the proposed model. The teacher model is pre-trained on ImageNet[62] and fine-tuned on the parking dataset. In the training process, the initial learning rate is 0.001 with a momentum of 0.9 and cosine decay. The total training epoch number is 30.

This chapter evaluates both accuracy and efficiency of the design system. Thus, this experiment section can be divided into four parts. Section 3.3.2 evaluates the accuracy of parking status identification on single parking slot under different weather conditions. Section 3.3.3

presents evaluation results of the model efficiency on both single parking slot and the whole parking area. Section 3.3.4 conducts ablation tests to demonstrate the effectiveness of optimization strategies – separable convolution, residual connection, and knowledge distillation. Section 3.3.5 evaluates how the proposed model can improve the overall parking monitoring accuracy by conducting a field test. Meanwhile, a simulation is conducted to illustrate the impacts of parking status identification efficiency on parking monitoring accuracy in a more general perspective. The baseline classification models include AlexNet, VGGNet, ResNet, and MobileNet v2. To be more specific, VGGNet refers to VGG-16, and ResNet refers to ResNet-18 in consideration of the balance of accuracy and efficiency. All these models are undergoing the same procedure described in Figure 3.4 and Algorithm 1 to traverse every single slot and then to compute the occupancy rate of the whole parking area.

3.3.2 Evaluating Accuracy of Parking Status Identification

The accuracy evaluation experiment operates on a PC and has two parts – single parking space and overall parking lot occupancy detection accuracy. The first part focuses on the accuracy of detecting each parking slot under different weather conditions, e.g., sunny, cloudy, and rainy. Based on the previous research [321], weather condition generates considerable impacts on the detection results. For example, straight sunshine light or raindrops on the lens can lower the detection accuracy. The single parking status detection results on the public dataset are shown in Table 3.2, where the unit is the percentage. According to Table 3.2, the accuracy of all algorithms is over 90%, which indicates that deep learning algorithms can handle this task very well under different conditions. ResNet has the highest accuracy in both CNRPark and PKLot datasets among these algorithms. The proposed SimpleNet and its

quantized version Q-SimpleNet have comparable accuracy to this algorithm. There is almost no accuracy difference between SimpleNet and MobileNet v2. The primary reason that the performance in the CNRPark dataset is higher than PKLot is that the former dataset has a higher resolution. After comparing different weather conditions, the overcast weather has the highest accuracy. However, its difference is less than 0.5% which can be ignored in most cases.

Table 3.2: Detection Accuracy Comparison for Single Slot

Dataset	AlexNet	VGGNet	ResNet	MobileNet	SimpleNet	Q-SimpleNet
CNRPark	91.4%	95.5%	**99.6%**	99.2%	98.9%	98.6%
Sunny	91.3%	95.3%	**99.4%**	99.2%	99.1%	98.8%
Cloudy	91.5%	95.6%	**99.8%**	99.4%	98.5%	98.3%
Rainy	91.1%	95.2%	**99.6%**	98.9%	99.0%	98.2%
PKLot	90.4%	94.1%	**98.1%**	98.0%	97.7%	97.5%
Sunny	90.1%	94.0%	97.9%	**98.0%**	97.4%	97.3%
Cloudy	91.0%	94.2%	**98.5%**	98.3%	98.0%	97.8%
Rainy	89.9%	93.8%	**97.9%**	97.8%	97.5%	97.3%

The second part focuses on the accuracy of the whole parking lot, whose results are shown in Table 3.3. The total parking slot number in CNRPark and PKLot is 320 and 168, respectively. The evaluation metric is Mean Square Error (MSE), whose computation function is shown in Equation 3.3, where $O(t)$ indicates the ground-truth occupancy at time t, and $\hat{O}(t)$ indicates the detected occupancy at time t. T refers to the entire detection period. Based on Table 3.3, these base models all have a low error rate in computing the occupancy rate. Additionally, the MSE in PKLot is slightly lower. However, there is no statistical difference between them, which also confirms the eligibility of deep learning models, including the pro-

posed Q-SimpleNet.

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(O(t) - \hat{O}(t))^2 \tag{3.3}$$

Table 3.3: Detection Accuracy Comparison for Parking Lot

Dataset	AlexNet	VGGNet	ResNet	MobileNet	SimpleNet	Q-SimpleNet
CNRPark	2.1e-3	2.0e-3	**1.8e-3**	**1.8e-3**	1.9e-3	1.9e-3
PKLot	1.1e-3	1.1e-3	**9.8e-4**	9.9e-4	1.0e-3	1.0e-3

When comparing Table 3.2 and 3.3, deep learning algorithms have reached high accuracy in this parking status detection. Since each parking space is stationary in an image after camera installation, this status detection is a classification task with a lower difficulty than general object detection tasks. It does not need to predict localization coordinates, which are pre-defined when installing the system. Therefore, these algorithms are all capable of handling the parking detection challenge. This chapter adopts the MobileNet v2 model as the base model to create the SimpleNet model considering satisfying accuracy and outstanding efficiency, even though it does not have the highest accuracy. Besides accuracy, another significant factor affecting the system performance is efficiency. The next part will compare the efficiency between the proposed model and other classification models and demonstrate how efficiency influences parking duration estimation accuracy.

3.3.3 Evaluating Efficiency of Parking Status Identification

Since parking occupancy and parking duration of each parking vehicle are updated based on parking status identification results, the identification algorithm efficiency is critical for the real-time performance of the entire parking surveillance system. The algorithm with high

efficiency would greatly benefit from deploying the identification algorithm on edge devices with limited computing power. Thus, in this section, the efficiency of the proposed algorithm for parking status identification is evaluated on both PC and Raspberry Pi 3. The efficiency comparison results for the single parking slot are shown in Table 3.4, where the unit is a millisecond. In Table 3.4, the second column shows the inference speed using CPU on PC. The third column shows the inference speed using GPU on a PC. The last column shows the inference speed on Raspberry Pi 3. Since two datasets have two kinds of image resolution, the experiment results also show two values for each evaluation metric under two patch sizes. Specifically, the first value in each cell is using the image resolution of 112×112. The second value is using the image resolution of 56×56. According to the results, VGGNet consumes almost one second to process a single slot image patch. If it runs to detect a parking area with 60 slots, it will take more than one minute to finish one iteration, which makes this algorithm unable to deploy on edge devices. The proposed SimpleNet takes almost half the inference time of MobileNet v2 since SimpleNet's parameter number is also 60% less than MobileNet v2. Compared with other algorithms, the inference speed on CPU devices of the proposed Q-SimpleNet is over two times higher than the non-quantized model and at least four times of AlexNet and ResNet. In addition, the inference speed of the Q-SimpleNet using CPU is almost as fast as SimpleNet using GPU on PC. Since the Raspberry Pi 3 has a weak CPU and small RAM, the Raspberry Pi 3 is almost six times lower than the PC when conducting the same task. Meanwhile, Q-SimpleNet on Raspberry Pi 3 can still keep a real-time inference speed.

The efficiency comparison results for the whole parking area are shown in Table 3.5, where the unit is second. It is testing one application scene with 100 parking slots from the PKLot

Table 3.4: Inference Speed for Single Parking Slot (Unit: Millisecond)

Algorithms	PC (CPU)	PC (GPU)	Raspberry Pi
AlexNet	32.7 / 23.4	9.0 / 8.9	250.1 / 170.3
VGGNet	139.3 / 81.2	15.5 / 13.4	1134.5 / 733.2
ResNet	31.3 / 22.1	11.5 / 11.3	244.6 / 162.1
MobileNet v2	26.1 / 17.8	9.5 / 8.3	200.1 / 155.5
SimpleNet	15.1 / 10.7	5.0 / 3.5	110.5 / 80.7
Q-SimpleNet	5.9 / 4.0	- / -	**40.8 / 30.2**

dataset. To evaluate the impacts of different image sizes. A single slot image is resized to 112×112 and 56×56 after image cropping from the raw frame, similar to the previous experiment. However, the detection time for the whole parking lot is not equal to the inference time of a single slot by multiplying the total parking slot number according to Table 3.5. Considering the time consumption of reading and cropping images, which is called preparation time, the total time will be a bit longer. In this chapter, the average preparation time is about 40 milliseconds for each slot. It can be found the loading and cropping time takes almost half the total processing time when the inference time is close to the preparation time. When the AlexNet or ResNet is applied to detect single parking slot availability, the time difference between Q-SimpleNet and AlexNet or ResNet will be at least 20 seconds under the resolution of 112×112 and 14 seconds under the resolution of 56×56. If adopting the low-resolution image, the proposed algorithm can complete the detection work in 7.8 seconds to iterate all 100 parking slots. Based on Algorithm 1, a shorter inference time can significantly help reduce the interval between iterations and thus increase the reliability of occupancy detection results. Q-SimpleNet will put almost no negative impact on parking time computation in the perspective of inference time. All other algorithms will generate adverse effects on the computation error. As a result, the proposed quantized parking occupancy detection system

Table 3.5: Inference Speed for All Parking Slots (Unit: Second)

Dataset	AlexNet	VGGNet	ResNet	MobileNet	SimpleNet	Q-SimpleNet
112×112	29.2	118.4	29.6	24.4	15.9	**8.6**
56×56	21.8	78.0	20.2	19.7	12.8	**7.8**

has a great potential to be implemented on the edge device with low computing power.

3.3.4 Ablation Tests on Parking Status Identification Model

The SimpleNet inherits separable convolution and residual connection from MobileNet v2. One ablation test is conducted to evaluate the effectiveness of these two modules. The test dataset is PKLot, and the test platform is Raspberry Pi. All input images are resized to 112×112 before the test to avoid the disturbance of data loading and dynamic resizing time. The test results are shown in Table 3.6, where the unit for inference speed is millisecond. In Table 3.6, the second row indicates the model without the separable convolution by replacing the separable convolution with the conventional convolution. The third row indicates the model without the residual connection by removing the residual shortcut in SimpleNet while keeping other structures unchanged. The last row indicates the model without both separable convolution and residual connection. According to Table 3.6, it can be found SimpleNet achieves the best trade-off between accuracy and efficiency. The inference speed difference between the first and second row shows that promising acceleration brought with separable convolution. Comparing the first row and third row, the residual connection improves accuracy significantly but has almost no negative impact on inference speed. Another ablation test is evaluating the effectiveness of knowledge distillation compared to the general training strategy applied in the pre-training step. When using SimpleNet as the base model, the accuracy using knowledge distillation for post-training is 98.1%. Without knowledge distil-

Table 3.6: Ablation Test for Separable Convolution and Residual Connection

Algorithms	Accuracy	Inference Speed
SimpleNet	98.1%	110.5ms
w/o S.	**98.5%**	223.2ms
w/o R.	95.7%	**108.1ms**
w/o S. and R.	96.3%	220.7ms

lation, the accuracy drops to 95.4%. The test results show that the knowledge distillation dramatically improves the model by increasing 2.7% accuracy.

3.3.5 Evaluating Impacts of Efficiency on Accuracy

According to Algorithm 1, occupancy and parking duration detection are updated based on iterative parking status identification results. The time interval between iterations certainly affects occupancy and parking duration detection accuracy. For example, if the employed identification algorithm is inefficient, the time interval between iterations becomes more prominent, it is highly likely to miss the parking vehicle changes. The accuracy would be much lower when the algorithm is deployed on an edge device with limited computing power. The previous experiments confirm that the proposed Q-SimpleNet has a high efficiency without losing any accuracy compared with the state-of-the-art algorithms. Thus, in this section, a field test is conducted to demonstrate how much the proposed Q-SimpleNet improves the parking surveillance accuracy. In addition, a simulation is conducted to evaluate how the parking status identification efficiency can affect parking surveillance accuracy.

The field test results provide statistical results of state change interval and the detection accuracy in continuous videos. A parking vehicle change event is defined as changing one parking vehicle to another that could be the same or not. The statistical results are shown

Table 3.7: Parking State Change Interval and Frequency

Date	July 14th, 2020	July 21st, 2020
Mean Interval	19s	21s
Max Interval	26s	43s
Min Interval	12s	13s
Change Frequency	1.4/min	0.9/min

in Table 3.7. According to the results, the mean interval during busy hours is less than 21 seconds, which is very close to the minimum inference speed for all parking slots of AlexNet, ResNet, and MobileNet in Table 3.5. It indicates that the detection system using these four algorithms as the base model has a high probability of missing a parking vehicle change. In contrast, the inference speed for all parking slots of Q-SimpleNet is even shorter than the minimum vehicle change interval. As a result, the parking vehicle change interval is short and frequent during busy hours. A very efficient detection system is crucial to track each slot's parking vehicle and parking duration. Thus, the proposed Q-SimpleNet can handle this task well.

Meanwhile, the occupancy accuracy for the entire parking area and the parking duration accuracy are also evaluated. The accuracy of occupancy and detection detection is shown in Table 3.8. The evaluation period for parking occupancy detection accuracy is 5 seconds, which indicates comparing the detection results with the ground truth every 5 seconds. The evaluation platform is Raspberry Pi 3, and the evaluation metric is MSE shown in Equation 3.3. According to the results in Table 3.8, MSE of Q-SimpleNet is much lower than other algorithms. Since AlexNet, ResNet and MobileNet have a similar inference speed, these algorithms have identical MSE.

The parking duration accuracy focuses on how accurately the system detects the parking

Table 3.8: Evaluation of Parking Occupancy Accuracy

Algorithms	MSE (%)	Duration Error (Second)
AlexNet	0.095	35
VGGNet	0.750	61
ResNet	0.095	35
MobileNet	0.086	33
SimpleNet	0.066	22
Q-SimpleNet	**0.014**	**5**

duration for each slot. The evaluation metric is the average time error for instances with parking vehicle change with the unit of second. Since collected videos cannot cover all parking periods for each instance, the parking time is thus shorter than the real parking time. The parking duration only considers these slots that have a parking status change. The experiment results are shown in Table 3.8. It can be found that Q-SimpleNet can significantly reduce the parking duration error by more than 80% compared with other models. However, limited by the length of videos, the impacts of inference speed with the longer parking duration cannot be assessed. Therefore, a simulation for long-time parking is necessary to evaluate the impacts, whose framework is shown in Figure 3.9.

This chapter applies Matlab Automated Driving Toolbox for visualization assistance. The scenario visualization is shown in Figure 3.10, where the blue vehicle will leave the slot, and the orange vehicle will occupy that space after the first vehicle leaves. This simulation mimics the busiest scenario when there is only one available slot. The orange is waiting to enter the space when it is available. So the exchange period should be as short as possible. In Figure 3.9, two colored curves show the parking status of two vehicles. Value 1 represents that the target vehicle occupies the space, and 0 is the opposite. The unit of the x-axis is a pre-defined time stride. There is no exact value for this stride in this figure. Vehicle 1 leaves the parking

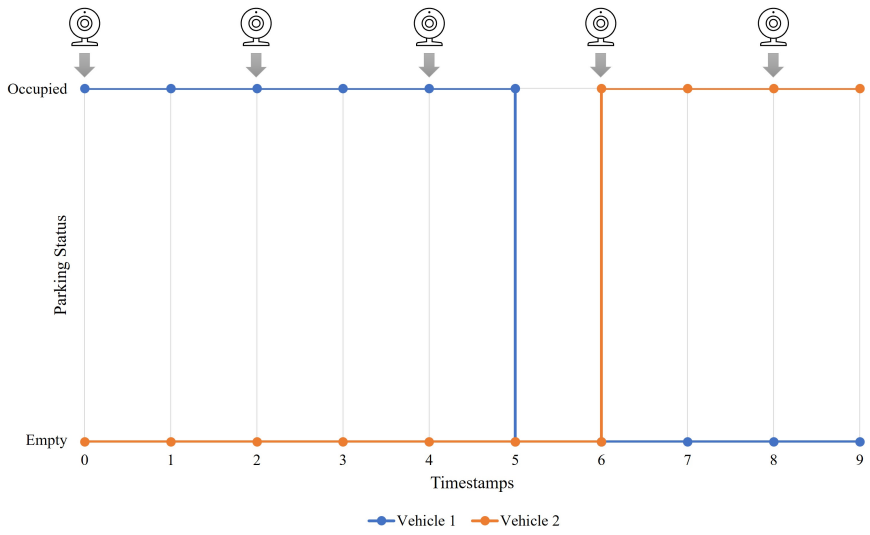


Figure 3.9: Simulation Schematic Diagram

slot at timestamp 5, and vehicle 2 enters it at timestamp 6. The interval between two cameras in the figure top represents the traversal time going through all parking slots. In Figure 3.9, the end of the first parking instance and the start of the second fall into the same traversal period, which means these two parking instances will be confused and considered as one instance. Therefore, the traversal period should be short enough to avoid this missing, which is equivalent to increasing the inference speed of the classification model. The basic idea of the simulation is to generate several parking instances randomly with a specific parking time, e.g., 30 and 60 minutes.

Figure 3.11 shows the simulation results of a single slot when the system has different inference times. According to the previous research work on indoor parking monitoring[148] and the statistics in Table 3.7, the state change interval from an occupied state to a new occupied state can be less than 15 seconds when vehicles are waiting for available spaces. The simulation is consistent with the fact. This simulation randomly selects the parking state

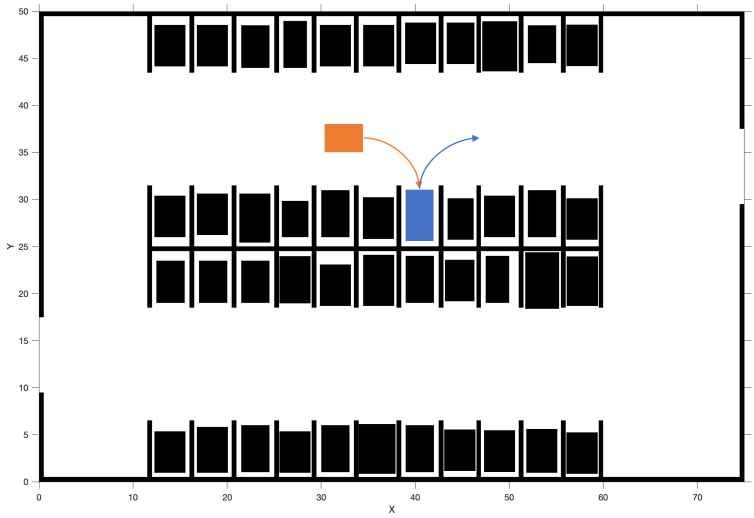


Figure 3.10: Simulation Schematic Diagram

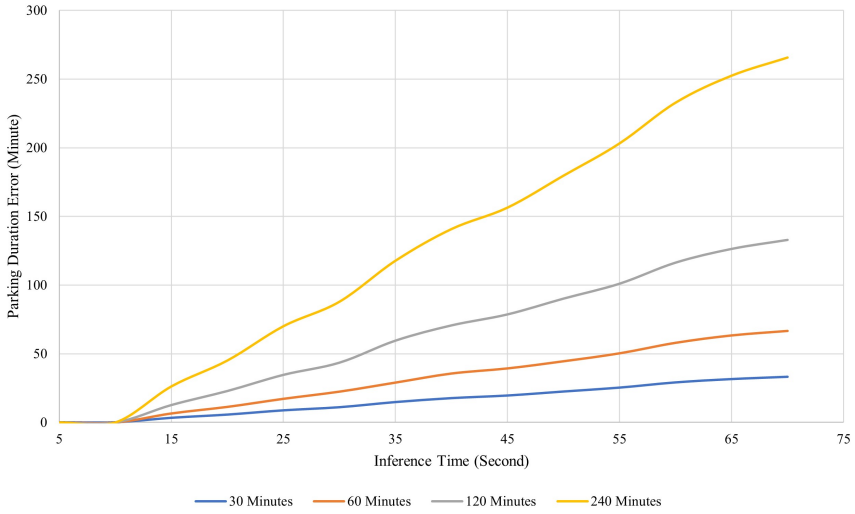


Figure 3.11: Simulated Parking Duration Error

change interval between 10 to 70 seconds based on uniform distribution. The duration of one parking instance is randomly set to 30, 60, 120, and 240 minutes. In Figure 3.11, the x-axis represents the inference speed of completing iterating all parking slots in an image. The y-axis represents the mean estimation time error, and its unit is minutes. Different colors rep-

resent the different mean parking duration. Based on the simulation results, the estimation error will increase significantly with increased mean parking time. When the traversal time reaches 60 seconds, the average error will be over 50 minutes when the parking duration is more than one hour. In addition, the broader coverage using one camera with more parking slots will have higher error by increasing inference time. This situation may be shared among large parking lots. For example, Angle Lake Station Parking Garage has 1,120 parking spaces. They are fully occupied during work hours because it is located at a transit center and accessible to the public. Daily commuters will park their vehicles in this garage and take the light rail to their workplaces. Each camera is expected to cover as many spaces as possible to cut down the hardware and operation costs of the monitoring system. While comparing different state-of-the-art classification models, efficiency plays an essential role in the overall system performance. Therefore, it is more significant to apply an efficient algorithm for parking detection than to adopt an accurate but complex algorithm. The proposed SimpleNet model can satisfy the demand of this task by providing good accuracy and impressive inference speed. Another potential solution is parallel computing in multiple threads and processing a batch of images simultaneously.

3.4 Chapter Conclusion

This chapter has explored the application of deep learning compression and knowledge distillation in ITS. Deep learning compression helps reduce the model size and boosts inference speed without changing the model framework. Thus, researchers and practitioners can pay less attention to model structure and adopt state-of-the-art models to meet their performance demands. Knowledge distillation is widely used in further improving model performance.

After model compression, there will be an accuracy drop because of losing parameter precision. Knowledge distillation will compensate for this drop.

Smart parking monitoring is an ideal scenario for the demonstration. The parking shortage has been troubling both commuters and management agencies, especially in transit centers and urban areas. It is urgent and necessary to build intelligent parking monitoring to access real-time parking information and provide it to users. Vision-based monitoring is optimal for large-scale parking spaces compared to other sensors, e.g., magnetic detectors. However, if transferring raw data back to servers for processing, there will be a high requirement for transmission bandwidth and computing resources. Edge computing is suitable to solve this concern. The deployed deep learning algorithm should be as efficient as possible, considering the limited computing power on edge devices. Deep learning compression and knowledge distillation will reduce the model complexity and keep the accuracy, which makes applying powerful deep learning models possible.

In conclusion, this chapter designs a parking monitoring system empowered by the deep learning model. The experiment results on public and self-collected datasets demonstrate the effectiveness of the designed system with high accuracy and efficiency on edge devices. What's more, this chapter evaluates how efficiency influences parking duration error. This system has a great potential for deployment in actual practice to monitor parking spaces in real-time.

4

Tiny Traffic Object Detection

4.1 Overview

Most machine learning and deep learning algorithms rely on supervised learning, which indicates annotated data is required for the training process. However, data insufficiency in practical applications is always a problem hindering the deployment of deep learning algorithms. Imbalanced data leads to low accuracy and high bias in minority categories, which

may cause serious traffic safety issues. Chapter 4 and the following chapter focus on this challenge and propose methods to reduce negative impacts on traffic sensing accuracy for minor categories. This chapter applies data augmentation to increase data variance in small-scale objects and improve the accuracy of crowd counting and motion detection. Besides, this chapter proposes a multi-task crowd monitoring pipeline to achieve crowd counting in different densities, group clustering, and motion detection.

4.1.1 Background

Data insufficiency and imbalance are common in deep learning practice as the total training sample number is limited or some categories have fewer training samples. The direct outcome will be overfitting, which makes the training process difficult. Although there is massive traffic data for analysis, the training data is still lacking. The essence of this situation is the supervised model training, which requests annotated data. The collected traffic data is unlabeled, and an extra step of data annotation is needed before being used for model training. According to Section 1.2.2, data annotation services, e.g., Amazon SageMake Ground Truth, are not a cost-effective method for fast-peace transportation applications. Thus, there are two potential solutions: utilizing unlabeled data and enhancing annotated data. Section 2.5.2 has reviewed recent works in self-supervised learning [112], i.e., contrastive learning [46], which achieved promising results in various downstream tasks [47]. One shared feature of contrastive learning methods is the high demand for computing resources. Meanwhile, contrastive learning requires carefully designed training strategies to enable the backbone to learn valuable features from unlabeled data. These contrastive learning methods may not be an optimal choice in ITS. As a result, enhancing the existing annotated data has the most significant potential to

improve the accuracy of minor categories for transportation applications. This chapter and Chapter 5 work on two different methods to utilize the annotated data. This chapter applies data augmentation to increase the data variance. To be more specific, the data augmentation in this chapter is used to improve small-scale object detection accuracy. Small-scale object detection is vital for traffic sensing because the camera is usually far from the targets. Autonomous driving and crowd monitoring are two scenarios requiring high small-scale object detection accuracy. With the urbanization process, crowd monitoring is essential to public safety. The cameras are usually installed at a higher location to increase monitoring coverage but make targets small. A practical method to improve small-scale object detection accuracy is significant.

4.1.2 Application Scenario – Crowd Monitoring

The fast urbanization does not only provide convenient transportation services but also promotes more crowd gathering activities, e.g., large shopping malls and transit centers. In recent years, there have been many crowd disasters because of ineffective crowd monitoring and management strategies. For example, 21 people died, and more than 500 people were injured in the Love Parade in Duisburg in 2010. During the Spring Festival event in Shanghai in 2014, 36 people were killed, and 42 people were injured in the stampede accident[318]. Public management agencies might observe the potential abnormal crowd flow if a crowd monitoring system is employed. They could provide support to avoid such tragedy in time. Besides crowd disasters in public facilities, the crowd is also marked as one of the vulnerable road users. Injuries and fatalities are more likely to occur than in motorized traffic participants since vulnerable road users do not have a metal shell to protect their safety[73]. The infrastruc-

ture and traffic rules need to be designed with the priority of the crowd flow safety[87]. Crowd monitoring has been attracting considerable priority in non-motorized traffic research due to the lessons learned from past events and experiences. Generally, crowd monitoring includes crowd counting and motion detection. An efficient and accurate crowd monitoring method is valuable for such goals from multiple perspectives, including understanding crowd mobility patterns, enhancing infrastructure safety design, prioritizing investment in infrastructure, assessing health and environmental impact, and analyzing crowd safety[197]. According to the aforementioned two-fold evidence, an effective crowd monitoring method is in-demand for enhancing crowd safety and mobility.

Existing crowd analysis studies have two primary directions: crowd simulation and real-time monitoring. Crowd simulation is typically conducted on strategic, tactical, and operational levels[152]. Compared to strategic and tactical simulation, operational simulation, e.g., Cellular Automata (CA)[369], force-based model[143], and velocity-based model[200], can reflect more detailed information like pedestrian motion patterns[357], which is thus widely studied and applied. The fundamental idea of these microscopic models is understanding human behaviors and simulating pedestrian motion. For example, in CA models, pedestrians discretely move through grids at a transition probability computed by predefined rules[23]. Most simulation models are built on prior experience or observations. In the model construction process, complicated behaviors are simplified and refined, leading to bias between the simulation and the practice. The predefined hypothetical rules in the simulation models may cause a certain degree of difference from reality[357]. The popularity of carry-on smart devices offers a potential opportunity for crowd monitoring by sniffing wireless beacons. Smart devices that turn on Bluetooth and Wi-Fi will periodically broadcast wireless communication frames

containing Media Access Control (MAC) address, used as a unique identifier for tracking crowd movements[362,227]. However, privacy concerns have triggered the implementation of MAC address randomization, generating the difficulty of detecting and tracking crowds using MAC address[204]. Besides, localization using limited sniffers in an open area is not precise because environmental factors like object occlusion affect the wireless signal strength. Comparably, vision-based methods are more robust, reliable, and implementable for crowd monitoring in different transportation facilities[138].

In current studies, the vision-based crowd monitoring method mainly focuses on two tasks – crowd counting[263,264], and crowd motion detection[233]. Previous studies of crowd counting were naturally developed from object detection[67,272]. These works firstly used an object detector to locate pedestrians and then counted the bounding box number as the crowd counting results. The inevitable occlusions and small-scale objects increase detection MR when the crowd density is high, and objects are far from the camera. To cope with high-density crowd counting, researchers propose direct counting methods by regressing extracted features and outputting the crowd counting result directly[32,265]. However, the employed regression methods only compute the crowd number loss instead of embeddings in the training process. Besides counting the crowd number, the spatial distribution of the crowd is also necessary to find irregular gatherings. Thus, several studies explored the methods for generating an accurate crowd density map, used as the complementary information for crowd counting[323,88]. Density-map-based algorithms may not work when the crowd density is low, but objects are distant from the camera. Increasing the small-object sample number and applying the pyramid structure to multi-level features is still necessary to enhance the small-object detection ability. Tiny and small-scale objects are scarce, making deep learning models unable to learn

enough information from data. Data augmentation thus can improve this condition by utilizing existing data but increasing data variance.

Crowd counting targets to collect the static characteristics of crowd flow using a single frame, while crowd motion detection focuses on the dynamic characteristics using multiple consecutive frames together. Most existing motion detection works were based on the optical flow theory[115], which applied the Kanade-Lucas-Tomasi (KLT) tracker[365,191] to compute the crowd flow. In addition, crowd groups can be identified by clustering individuals or key-points sharing similar spatial and motion features[39,301]. One major challenge of the flow-based method arises from the hand-crafted feature descriptor for key-point extraction, which cannot find representative points effectively, especially in a complicated scenario.

4.1.3 Contributions and Organization

Crowd monitoring analysis targets to detect abnormal crowd behavior to avoid potential crowd events[4]. Several vital characteristics can describe the crowd attributes in different dimensions at the same time, including density, flow, and speed[333]. However, in the previous studies, only a few works focused on offering an integrated framework for simultaneously generating multiple crowd flow parameters. Considering the aforementioned technical disadvantages of the existing studies and the identified research gaps, the current study proposes a multi-task crowd monitoring framework for crowd counting, localization, and motion detection using deep learning techniques. This chapter adopts a density-based switch strategy to accommodate different crowd densities and adjust the crowd counting model. Inspired by the lack of integrated crowd-attribute detection works[237], this chapter further synthesizes multi-object detection and density map estimation to analyze crowd counting and localiza-

tion. The crowd motion detection module is clustering motion patterns of individuals captured from multi-object tracking. The contributions of this chapter are summarized in the following four points.

a) Propose A multi-task crowd monitoring framework using deep learning techniques, including crowd counting, localization, and motion detection. The framework combines both density map estimation and multi-object detection for crowd analysis and adopts different detection strategies determined by the crowd density. this chapter innovatively integrates multiple detection modules for end-to-end crowd attribute detection.

b) Establish an efficient multi-scale crowd counting model based on the FCN. Instead of using the multi-column structure, the proposed model cascades dilation blocks with different dilation rates as an encoder to explore features at multiple scales. The proposed network achieves high accuracy without extra complicated network structures.

c) Propose a crowd motion detection strategy using multi-object tracking which supports the analysis of both individual motion patterns in low-density situations and crowd motion patterns in high-density situations. The crowd motion detection fully utilizes multi-object trakcing for effective target detection and tracking and surpasses traditional flow-based methods when the implementation scenario is complex.

d) Compare the crowd counting accuracy in both low and high crowd density conditions with state-of-the-art algorithms. The evaluation datasets cover individual detection and density estimation, e.g., TinyPerson, and UCF-CC-50. The evaluation re-

sults demonstrate the superiority of the proposed crowd counting models in different crowd densities.

The remaining of this chapter will first introduce the framework of this multitask crowd monitoring framework in Section 4.2, covering individual object detection model, crowd density estimation model, group tracking strategies, etc. In Section 4.2.2, two crowd counting model is described for low and high-density crowd. When the crowd density is low, this framework adopts the object detection model to count the person number. Since the targets are small, specialized data augmentation is designed to increase the sample number of tiny objects. The density estimation model starts working when the density is higher than a threshold. Section 4.2.3 describes how to cluster groups and estimate their motion based on multi-object tracking results. Then, experiments on evaluating the whole framework and data augmentation strategies are shown in Section 4.3. Firstly, Section 4.3.2 compares the object detection and density estimation models with previous research works in terms of detection accuracy. The next section evaluates the overall framework in crowd motion detection and compares it with the optical flow method.

4.2 Methodology

4.2.1 Multitask Crowd Monitoring Model Framework

This study innovatively proposes an efficient and integrated framework for simultaneously addressing multiple tasks regarding crowd monitoring, including crowd counting, localization, and motion detection. The framework mainly has two parts in terms of functionality – crowd counting and motion detection. The overall framework is shown in Figure 4.1.

As shown in Figure 4.1, the input images will firstly pass through both the object detection model and density estimation model. Their details will be described in the coming part. Only the object detection model will work for crowd counting when the crowd density is low. Once the crowd density is higher than a threshold, the density estimation model will involve and calculate the crowd number. Under this condition, the output of bounding boxes from the object detection model is used for motion detection. Meanwhile, bounding boxes in adjacent frames will be associated based on feature appearance for multi-object tracking. The multi-object tracking results will then be integrated with density for clustering to compute the crowd motion. The final outputs of this framework include crowd counting results, density distribution, and crowd motion patterns. Crowd counting and motion detection will be elaborated in Section 4.2.2 and Section 4.2.3 correspondingly.

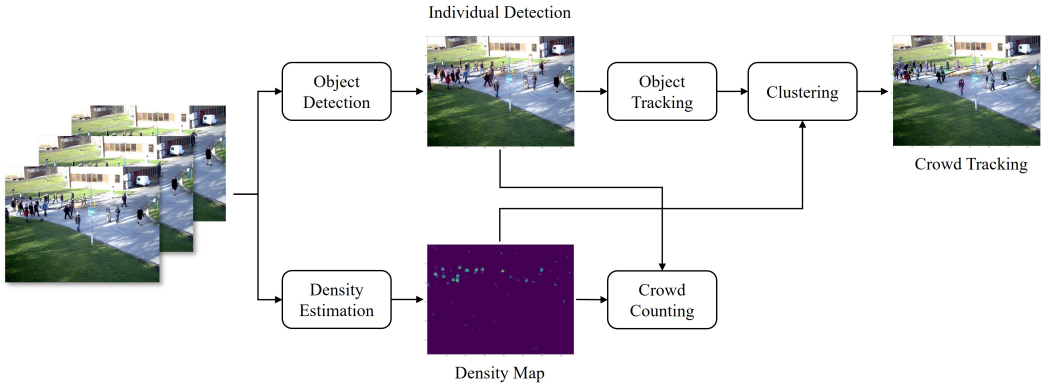


Figure 4.1: Framework of Crowd Counting, Localization and Motion Detection

4.2.2 Crowd Counting

According to Figure 4.1, the crowd counting module synthesizes both counting bounding boxes from the object detection model and summing density map from the density estima-

tion model. In general, counting with object detection performs better in the sparse-crowd conditions while counting with the density estimation is more suitable for dense-crowd conditions. Therefore, this study adopts a switch strategy that counts the crowd using object detection when the crowd counting is lower than a threshold. The approximate crowd counting for switch determination is computed using the density map. This strategy is explained in Equation 4.1, where N_{crowd} refers to the final crowd count. N_{bbox} refers the bounding box number from object detection. $\rho(x, y)$ refers to crowd density at location (x, y) in the image. t represents the threshold to determine which input is used to estimate the final crowd count. As a result, the accuracy of object detection and density estimation will determine the final accuracy of crowd counting.

$$N_{\text{crowd}} = \begin{cases} N_{\text{bbox}} & \text{if } \sum_x \sum_y \rho(x, y) < t \\ \sum_x \sum_y \rho(x, y) & \text{otherwise} \end{cases} \tag{4.1}$$

The individual object detection algorithm is built on YOLO v4, a typical one-stage deep-learning detector[24]. The outputs of the object detector offer crowd counting numbers under the sparse-density condition. Additionally, it provides anchors to track objects in the next step. Its accuracy and efficiency are equally significant. Crowd monitoring always requires real-time processing, especially when detecting abnormal activities. One-stage detectors are more suitable than two-stage detectors. YOLO v4 inherits the basic framework of YOLO v3 and absorbs advanced deep-learning components and training strategies to simultaneously achieve high accuracy and efficiency. Considering the application scenario that estimates the crowd number, the individuals are relatively small compared to general object detection. Thus, larger feature maps are more valuable than smaller feature maps. In

this chapter, only larger feature maps are activated for prediction. The optimized detection framework is shown in Figure 4.2. The primary difference between the optimized and YOLO v4 is using large feature maps for prediction. In this chapter, Only M1 and M2 layers work for prediction. Feature maps with larger receptive fields will not join in the prediction process. M1 and M2 have sufficient feature depth thanks to the feature pyramid structure.

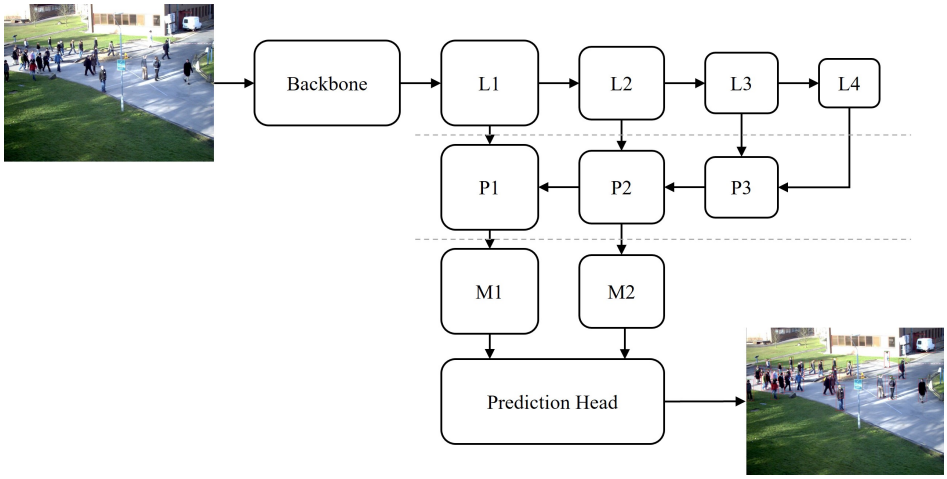
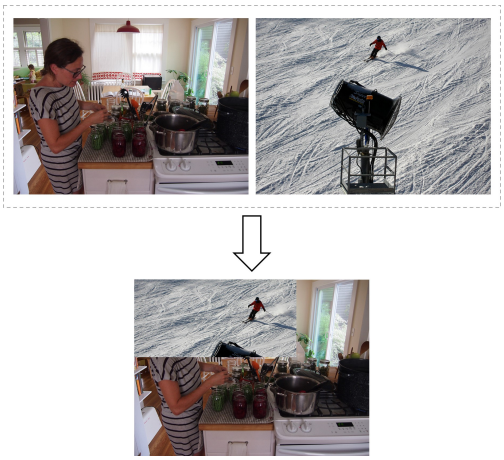


Figure 4.2: Optimized YOLO Framework

As described before, one reason leading to low accuracy is the lack of small-scale samples. Deep learning heavily relies on training data. Sufficient and balanced data is hard to obtain in practice. There are always various data issues, e.g., data imbalance and noise. In the crowd monitoring task, small-scale samples are lacking. It is well-known that collecting and labeling new data cost a lot of time and budget. Utilizing existing data by data augmentation is the most straightforward solution to increase data variance. Inspired by Stitcher[50] and CutMix[342] data augmentation, this chapter proposes a robust stitching strategy called Zoom-Stitcher. The basic idea is to conduct random cropping on four separate images and resize them into the same dimension. Then those images are composed to form a complete input

image for training. The framework comparison of Zoom-Stitcher and CutMix is shown in Figure 4.3. According to Figure 4.3b, CutMix conducts the random cropping on the base image and pastes an image patch of the exact location from another image. This operation tries to mimic the object occlusion situation and make the model learn knowledge from outside the base image. However, CutMix does not consider the scale change. Zoom-Stitcher, shown in Figure 4.3a, does not apply random cropping a small patch and then pasting considering the firm information inconsistency at the border of the image patch. Instead, Zoom-Stitcher composes four cropped images to generate a new image. The inconsistency between image borders can have regularity, which makes deep learning models easy to learn valuable information from composed images.

(a) Zoom-Stitcher Framework



(b) CutMix Framework

Figure 4.3: Data Augmentation Comparison

The density estimation model is based on CSRNet[177]. One critical component of CSRNet is dilated convolution that has been demonstrated in semantic segmentation[339] and object detection[186]. Dilated convolution can enlarge the receptive field while not increasing the computing complexity. Following previous work of MCNN[286], a multi-column structure is adopted to capture characteristics at different scales. Inspired by recent work of YOLOF[40], an encoder composed of cascaded dilated blocks with different dilation rates can generate features with multiple receptive fields while reducing the computing complexity significantly. Therefore, this chapter proposes a density estimation model called Multi-Scale Network with one-level Feature (MSNet-F), whose framework is shown in Figure 4.4. The feature extractor is based on ResNet-50. The down-sampling rate of the feature extractor is 8, which indicates the output density map dimension is 1/8 of the input image. The up-sampling module is used to map the dimension of the output density map to the input image. Meanwhile, the up-sampling module will also reduce the final channel number to one through 1×1 convolution operation.

One major contribution of this chapter is the innovative encoder with strong feature extraction ability and high efficiency. The encoder contains three consecutive residual blocks with dilation rates of 2, 4, and 6. The structure of the encoder is shown in Figure 4.5, where C_3 is the output feature map from the feature extractor, and P_3 is the output feature map from the encoder. 3×3 convolution is dilation convolution. The structure of one dilated residual block is very similar to the general residual block shown in Figure 3.5b. The difference lies in the usage of dilated convolution. Batch normalization and activation layers are omitted in Figure 4.5, which are following the convolutional layer. Although the feature pyramid structure is promising, it will increase extra computing costs. Compared to object

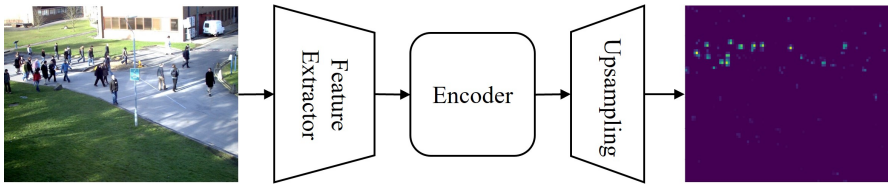


Figure 4.4: MSNet-F Framework

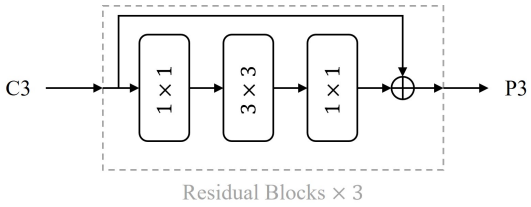


Figure 4.5: Dilation Encoder Framework

detection, density estimation has a lower requirement for localization precision. The cascaded blocks with multiple dilation factors are adequate to explore multiple feature levels. The loss function of MSNet-F is L2 loss, which is defined in Equation 4.2, where B refers to the batch size, and F indicates the density map.

$$L = \frac{1}{2B} \sum_{i=1}^{B} \left\| F_i - F_i^{gt} \right\|^2 \tag{4.2}$$

4.2.3 Crowd Motion Detection

Crowd motion detection involves multi-object tracking and clustering according to Figure 4.1. The multi-object tracking applies the SORT method that is super efficient without extra training process[20]. SORT uses the Kalman filter and Hungarian algorithm for tracking bounding boxes predicted from the object detection model. The outputs of SORT include tracking id as well as bounding box coordinates. The implementation detail can be referred to the original work, which will not be described further. A sample of tracking results using

Figure 4.6: Sample Image of SORT Tracking

YOLO and SORT is shown in Figure 4.6, where different colors represent different tracking ids. As discussed in Section 2.3.2, the object-detection-based method performs well in sparse-crowd conditions where each object can be easily recognized. However, occlusions are common in dense-crowd conditions. As a result, only partial objects in the image can be detected and tracked, which is also shown in Figure 4.6. This chapter thus proposes a strategy with multi-object tracking as well as density-map-based clustering to infer the crowd movement. Firstly, YOLO and SORT are implemented to track objects. Then tracked objects will be grouped using a hierarchical clustering method. Similar to crowd counting, the next step will depend on the crowd density. If the crowd density is low, the clustering results are used to represent the crowd motion. If the crowd density is high, a further clustering with density map will be conducted to estimate the crowd motion.

Hierarchical clustering is used to group objects recognized by the multi-object tracking method. A group of objects are more likely to have a similar motion pattern or act as a union in collective motion[221]. Both object location and movement are considered as clustering features. For a tracked object i, its bounding box coordinates of at time stamp t are defined as $(x^t_{i,1}, y^t_{i,1}, x^t_{i,2}, y^t_{i,2})$. Its center defined as $(x^t_{i,c}, y^t_{i,c})$ can be computed by mean value of bounding box coordinates. If it is also tracked at time stamp $t-1$, its movement d^t_i can be written

as $(x_{i,c}^t - x_{i,c}^{t-1}, y_{i,c}^t - y_{i,c}^{t-1})$. Therefore, the feature vector v_i^t of tracked object i at time stamp t can be finally written as $v_i^t = (x_{i,c}^t, y_{i,c}^t, \overrightarrow{d}_i^t)$. If object i is not tracked at time stamp $t-1$, d_i^t is equal to $(0,0)$. These objects will be clustered via this feature vector. Two kinds of distance measurements are applied, which are Euclidean distance $\mathrm{Dist}(i,j)$, and cosine distance $\mathrm{Cos}(i,j)$. Their expressions are shown in Equation 4.3 and Equation 4.4.

$$\mathrm{Dist}(i,j) = \sqrt{\left\|x_{i,c}^t - x_{j,c}^t\right\|^2 + \left\|y_{i,c}^t - y_{j,c}^t\right\|^2} \tag{4.3}$$

$$\mathrm{Cos}(i,j) = \frac{d_i^t \cdot d_j^t}{\|d_i^t\| \left\|d_j^t\right\|} \tag{4.4}$$

The detailed implementation of hierarchical clustering is shown in Algorithm 2. τ_d is the coordiante distance threshold and τ_c is the movement distance threshold.

Algorithm 2: Workflow of Hierarchical Clustering

```
for v_i^t, v_j^t (i ≠ j) do
    if Dist(i,j) ≤ τ_d then
        if d_i, d_j ≠ 0 and Cos(i,j) ≥ τ_c then
            i, j are clustered into group n;
            Update the feature vector v_n^t of group n;
        else if d_i = 0 or d_j = 0 then
            i, j are clustered into group n;
            Update the coordiantes (x_{n,c}^t, y_{n,c}^t)of group n;
        Match current and previous group ids by distance;
```

Hierarchical clustering can group detected objects in the neighborhood as shown in Figure 4.7, where circles indicate the clustering centers and different colors represent different cluster ids. When the crowd density is low, these groups can be seen as the final groups as output.

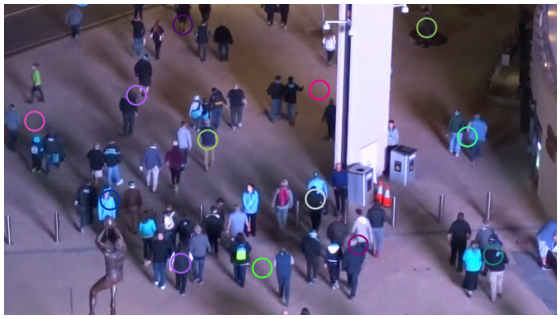


Figure 4.7: Sample Image of Hierarchical Clustering

However, partial objects are missing in the dense situation. It is almost impossible to infer the group based upon the detection results. Therefore, the density map is used to estimate the group. The fundamental assumption is that each group has one clustered object. The final group number is equal to the number of clustered objects. A thresholding-based image segmentation method is applied to distinguish pedestrian predictions from the background in the density map. Since ground truth density maps are generated from heads or pedestrian locations, the thresholding results can represent the approximate locations of pedestrians. The k-means clustering is then applied to group crowds, which only utilizes localization features. The cluster centers are initialized with the hierarchical clustering results. The framework of k-means clustering using density map and hierarchical clustering results is shown in Figure 4.8. The final outputs include the crowd localization as well as movement directions.

4.3 Experiments

4.3.1 Experiment Setting

This chapter works on applying data augmentation to increase small-scale object detection accuracy, especially for crowd monitoring. Since this chapter proposes a crowd monitoring

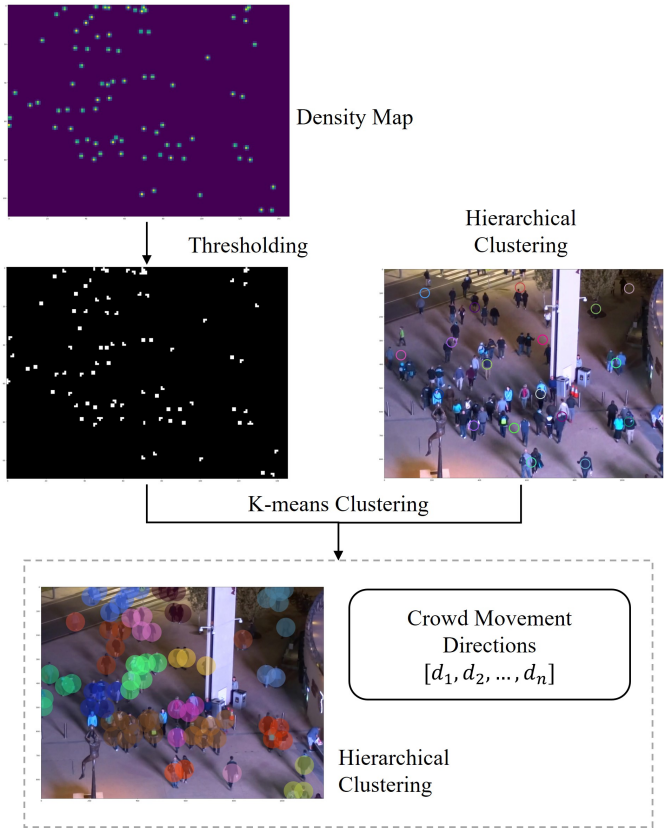
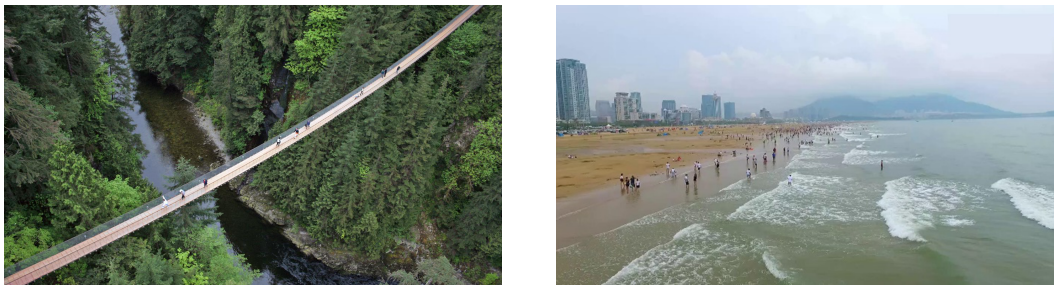


Figure 4.8: K-means Clustering Framework

pipeline responsible for crowd counting, group gathering, and motion detection, the experiment section has been divided into crowd counting and motion detection. Firstly, Section 4.3.2 focuses on crowd counting accuracy. This section introduces individual object detection and crowd density estimation. Individual object detection is the most important in this pipeline because its outputs are inputs for group gathering and motion detection. Evaluation of individual detection accuracy follows the rule in general object detection on two public datasets – COCO[182] and TinyPerson[340]. The ablation test of the proposed ZoomStitcher is conducted as well. The input resolution is 512×512 for both datasets. Based on

object sizes, there are small, medium and large objects corresponding to pixel sizes of $[0, 32^2]$, $[32^2, 96^2]$, and $[96^2, \infty]$ in the COCO dataset. The TinyPerson dataset is particularly built for tiny object detection. Compared to other detection datasets, most samples in this dataset are tiny and small objects. This dataset has a finer division of small objects than the COCO dataset. The sizes of tiny and small objects are $[2^2, 20^2]$ and $[20^2, 32^2]$. Two sample images from the TinyPerson dataset are shown in Figure 4.9 including persons in the forest and on the beach. It is a huge challenge to detect persons on that scale.



(a) Tiny Persons in the Forest **(b)** Tiny Persons on the Beach

Figure 4.9: Samples Image of TinyPerson Dataset

When considering the dense crowd condition, the proposed density estimation model MSNet-F is compared with other state-of-the-art crowd counting methods in terms of accuracy. One contribution of this chapter is designing the switch strategy to adopt the proper model based on the current crowd density. Thus, in the sparse crowd condition, the object-detection-based method is compared with the density-map-based method to prove the effectiveness of the proposed switch strategy shown in Equation 4.1. Three public dense crowd datasets are used for training and evaluation, which are UCF-QNRF[129], UCF-CC-50[128] and ShanghaiTech[353]. The image information of each dataset is shown in Table 4.1, including Gaussian kernel size and image scale. If one person only occupies one pixel or several pixels in

Table 4.1: Image Information of Different Crowd Datasets

Dataset	Kernel Size	Image Scale
UCF-QNRF	15×15	max(h, w) = 1024
UCF-CC-50	Geo-adaptive	max(h, w) = 1024
ShanghaiTech A	15×15	max(h, w) = 1024
ShanghaiTech B	15×15	768×1024

the heatmap, the heatmap information will be sparse, which is not suitable for model training. Gaussian blurring is applied to generate the ground-truth density map based on head annotations in data pre-processing[190,177]. Gaussian blurring function is shown in Equation 4.5, where δ refers to the ground truth in the image. G_{σ_i} refers to the Gaussian kernel with standard deviation σ_i. In Table 4.1, Geo-adaptive indicates the geometry-adaptive kernel, whose σ_i is computed via the average distance of k nearest neighbors. Considering that the dataset size of UCF-CC-50 is only 50, 5-fold cross-validation is adopted in the experiment as suggested in previous works[128,300]. The training strategy for density estimation models follows the C^3 framework[88]. The physical batch size is 4 in the training process. This chapter uses gradient accumulation with an accumulation step of 4 to enlarge the batch size for better convergence. The total training epoch number is 200. The initial learning rate is 0.0001 with a cosine learning rate decay.

$$D(x) = \sum_{i=1}^{N} \delta(x - x_i) * G_{\sigma_i}(x_i) \tag{4.5}$$

The crowd counting algorithm proposed in this chapter is designed for both dense and sparse crowd conditions. The crowd number varies during different time periods at the same place. It is necessary to evaluate the counting accuracy when the crowd density is low. This chapter utilizes MOT 2015 dataset[169] for the experiment. Its average crowd number in each

image is around 10, while the crowd number in the previous experiment is much higher. For example, the average crowd number is 815 in UCF-QNRF, and 501 in ShanghaiTech A. MOT 2015 dataset was originally used as the multi-object tracking benchmark. Since the training set has annotations of all appeared objects, it becomes possible to transform these annotations to generate density maps with Gaussian blurring for crowd counting evaluation. Following the strategy used in UCF-CC-50 dataset, 5-fold cross-validation is also applied here. What's more, the density estimation model is pre-trained on UCF-QNRF and fine-tuned on a mixture dataset of UCF-CC-50 and MOT 2015 to learn from both high and low-density datasets. In order to avoid overfitting, the fine-tune epoch number is constrained to 50.

Secondly, Section 4.3.3 is focusing on crowd tracking performance in terms of crowd coverage and detection accuracy. The proposed motion detection framework is compared with the optical-flow-based method that was widely used in previous crowd motion detection works. The dataset applied for evaluation is MOT 2020[61]. Compared to MOT 2015, this one has a denser crowd condition covering both indoor and outdoor spaces. The average crowd number of MOT 2020 is 150 that is multiple times higher than MOT 2015. Similarly, the training set, whose sample size is 8,931, is used for model training and evaluation since there are annotated bounding boxes for all objects.

4.3.2 Evaluating Crowd Counting

Since this chapter proposes a combined crowd counting method utilizing object detection and density estimation models, it is meaningful to evaluate each module separately. Thus, the proposed object detection model is compared with other state-of-the-art models on COCO

and TinyPerson datasets. The experiment results on these datasets are shown in Table 4.2 and Table 4.3, respectively. The proposed detection model is compared with with Faster R-CNN, RetinaNet, CornerNet[168], and YOLO v4 in terms of efficiency and accuracy.

Table 4.2: Individual Detection Accuracy in the COCO Dataset

Algorithms	Speed (FPS)	AP (%)	AP_S (%)	AP_M (%)	AP_L (%)
Faster R-CNN	7.0	36.7	21.1	39.9	48.1
RetinaNet	11.1	34.4	14.7	38.5	49.1
CornetNet	4.4	40.5	20.8	44.8	**56.7**
YOLO v4	31.0	**43.0**	**24.3**	**46.1**	55.2
Proposed	**40.2**	41.1	**24.3**	45.0	53.0

Table 4.3: Individual Detection Accuracy in the TinyPerson Dataset

Algorithms	Training	AP_{tiny} (%)	AP_{small} (%)
Faster R-CNN	Scale Match	43.5	56.7
Faster R-CNN	Zoom Stitcher	49.5	63.1
YOLO v4	CutMix	65.3	71.1
Proposed	CutMix	65.3	71.0
Proposed	Zoom Stitcher	**68.9**	**75.2**

According to Table 4.2, the proposed model has surpassed its base model YOLO v4 in the aspect of efficiency. The inference speed is almost 30% higher than YOLO v4 and several times higher than others. The proposed model has a slightly lower mean AP than YOLO v4 because it drops several feature maps. Its average AP is still higher than other models. When comparing accuracy for small objects, the proposed model has the same AP as YOLO v4, which implies that dropping these layers does not affect detecting small objects. For medium and large objects, the proposed one is not as accurate as YOLO v4 and CornerNet. The experiment results demonstrate that optimization is effective without sacrificing the accuracy of detecting small objects. Meanwhile, the inference speed is accelerated significantly. Ex-

periments on the TinyPerson dataset tries to evalaute the Zoom-Stitcher data augmentation strategy. Experiments on the TinyPerson dataset tries to evalaute the Zoom-Stitcher data augmentation strategy. Based on Table 4.3, the proposed mdoel trained with Zoom-Stitcher reaches the highest AP for both tiny and small objects. When comparing the last rows using the same model with different data augmentation strategies, the improved Zoom-Stitcher has a much better performance than CutMix by introducing scale variance. The first two rows using Faster R-CNN with Scale Match and Zoom-Stitcher indicate that Scale Match brings more minor data variance. One more thing to notice is that the newer one-stage object detection can exceed old two-stage models. YOLO models have significantly higher AP than Faster R-CNN.

After comparing the proposed object detection model and Zoom-Stitcher on the individual object detection accuracy, another experiment is conducted on the MOT2015 dataset for crowd counting accuracy when the density is low. Unlike the first experiment of general object detection, this part pays more attention to overall crowd counting accuracy and does not care about whether a single object is detected. The object-detection-based method is YOLO pre-trained on the COCO dataset. Since MSNet-F exceeds other crowd counting methods, MSNet-F is selected to represent density-map-based methods. The comparison results are shown in Table 4.4. According to the results, it can be found crowd counting by object detection works better in sparse crowd conditions. Although the absolute error using density map estimation is not large, the relative error matters more since the original crowd number is around 10, which means the relative error is more than 80%. The primary reason of generating the error using YOLO is the occlusion when persons gather together. One potential solution to this occlusion problem is increasing the sample number of occluded objects by

Table 4.4: Crowd Counting Accuracy in the MOT 2015 Dataset

Methods	Training	MAE	RMSE
MSNet-F	General	8.4	8.6
Faster R-CNN	General	4.6	4.8
YOLO v4	CutMix	4.1	4.1
Proposed	Zoom Stitcher	**3.8**	**3.9**

randomly masking parts of objects to simulate the occlusion scenario.

Besides the individual detection model optimized for small-scale object detection, the density map estimation model plays an essential in crowd counting when the density is high. This section compares the proposed density map estimation model MSNet-F with state-of-the-art methods including M-SFANet[286], SGANet[300], CAN[190], CSRNet[177], Switch-CNN[246], and MCNN[353]. The evaluation metrics are Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), defined in Equation 4.6 and Equation 4.7 correspondingly. C refers to the predicted crowd number, and C^{gt} refers to the ground-truth crowd number. N represents the sample size of the evaluation set.

$$\text{MAE} = \frac{1}{N}\sum_{i}^{N}|C_i - C_i^{gt}| \tag{4.6}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i}^{N}|C_i - C_i^{gt}|^2} \tag{4.7}$$

The experiment results of crowd counting in the dense crowd condition are shown in Table 4.5. According to Table 4.5, the proposed MSNet-F achieves the highest accuracy in most evaluation sets. The only exception is that the accuracy of MSNet-F is a little bit lower than CAN in UCF-CC-50 datasets under dense conditions. One possible reason is the small sam-

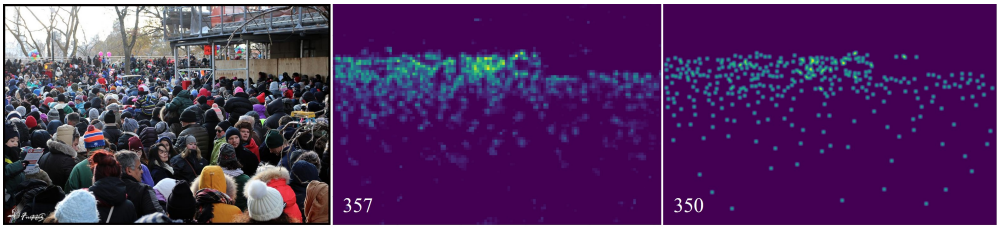
Table 4.5: Crowd Counting Accuracy in Public Crowd Datasets

Methods	UCF-QNRF		UCF-CC-50		ShanghaiTech A	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
M-SFANet	113	188	230	354	60	101
SGANet	103	178	225	315	58	100
CAN	107	183	**212**	**244**	62	100
CSRNet	115	190	266	398	68	115
Switch-CNN	228	445	318	439	90	135
MCNN	277	426	378	509	110	173
MSNet-F	**101**	**176**	218	306	**57**	**99**

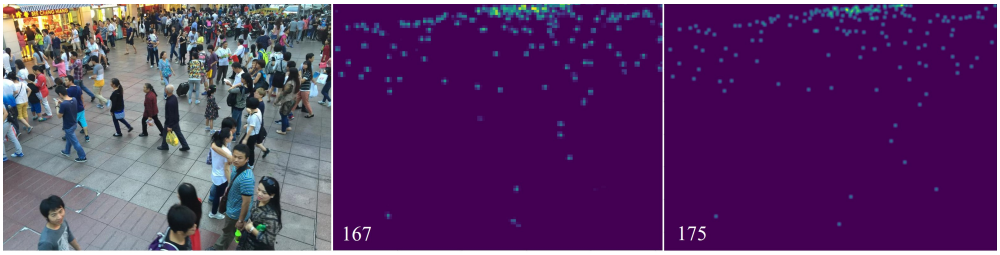
ple size which leads to overfitting. Pre-training on other crowd datasets may help solve this problem. However, the relative difference is less than 3%, which can be ignored in practical applications. As the result, the application of the dilation encoder is proved to increase the density estimation accuracy without complicated FPN or multi-column structure. The visualization results on UCF-QNRF and ShanghaiTech datasets are shown in Figure 4.10, where the left column displays the original image. The central column displays the predicted density map, and the right column displays the ground-truth density map. The white number in the image is the crowd number. In addition, the predicted density map is generated using the proposed MSNet-F model.

4.3.3 Evaluating Crowd Motion Detection

This section will compare the proposed crowd motion detection method with optical-flow-based methods[39,221]. The fundamental idea of optical-flow-based methods is applying optical flow to match key-points in adjacent frames and then group these key-points based on the spatial position and motion orientation. Three kinds of evaluation metrics are applied – Percentage of Crowd Coverage P_c and two kinds metrics of MAE. P_c is used to evaluate the



(a) Density Map Sample from UCF-QNRF Dataset



(b) Density Map Sample from ShanghaiTech Dataset

Figure 4.10: Visualization of Predicted and Ground-truth Density Maps

percentage of crowds that are detected, whose definition is $P_c = C_{trk}/C_{total}$. C_{trk} refers to the crowd count being detected and C_{total} refers to the total crowd count in the image. Since this chapter does not focus on object tracking, the evaluation metrics in multi-object tracking, e.g., multi-object tracking accuracy and precision are not used. Instead, two MAE metrics – MAE_v and MAE_d are used in this section and focus on perspectives of pixel velocity v_{ij} and movement direction d_{ij}. The corresponding definitions are shown in Equation 4.8 and Equation 4.9, where N refers to the frame number and C refers to the cluster number. The actual velocity can be estimated after obtaining the optical specifications of cameras in future work.

$$\text{MAE}_v = \frac{1}{NC}\sum_{i}^{N}\sum_{j}^{C}|v_{ij} - v_{ij}^{gt}| \tag{4.8}$$

$$\mathrm{MAE}_d = \frac{1}{NC}\sum_{i}^{N}\sum_{j}^{C}\left(1 - \frac{d_{ij}\cdot d_{ij}^{gt}}{\left\|d_{ij}\right\|\left\|d_{ij}^{gt}\right\|}\right) \tag{4.9}$$

The ground truth of crowd motion can be generated using tracking information, e.g., tracker id and bounding box coordinates, from MOT 2020 dataset. The individual motion pattern is computed via bounding box coordinates with the same tracker id. Then the crowd motion pattern is computed via clustering and averaging individuals with similar individual motion patterns as shown in Equation 4.10, where G is the individual number within one group, and d_i refers to the movement of individual i. The velocity of one group is the length of vector d^{gt}.

$$d^{gt} = \frac{1}{G}\sum_{i}^{G} d_i \tag{4.10}$$

The evaluation results of P_c is shown in Table 4.6. It can be observed that the proposed method has a much higher coverage of 90% than optical flow methods of 68%, which means optical flow methods cannot estimate the crowd movement objectively. The visualization is also shown in Figure 4.11, where the left one is the snapshot using the proposed method, and the right one is the snapshot using the optical flow method. Different colors in Figure 4.11 represent different group ids. The optical flow method can only extract key-points but not distinguish whether they belong to persons or other objects, which may lead to false clustering. In Figure 4.11b, key-points from the right building are also extracted, which are irrelevant to this task. The assistant classification work is thus required like crowd density map [141]. What's more, the optical flow method always depends on hand-crafted descriptors for feature extraction, e.g., Shi-Tomasi Corner [256], which cannot cope with the complicated

situations effectively compared to deep learning feature extractors. Some objects that do not have large intensity variations may be ignored.

Table 4.6: Crowd Coverage Percentage

Methods	C_{trk}	C_{total}	P_c
Optical Flow	150	102	68%
Proposed	150	**135**	**90%**

(a) Proposed Algorithm **(b)** Optical Flow Algorithm

Figure 4.11: Visualization of Crowd Coverage

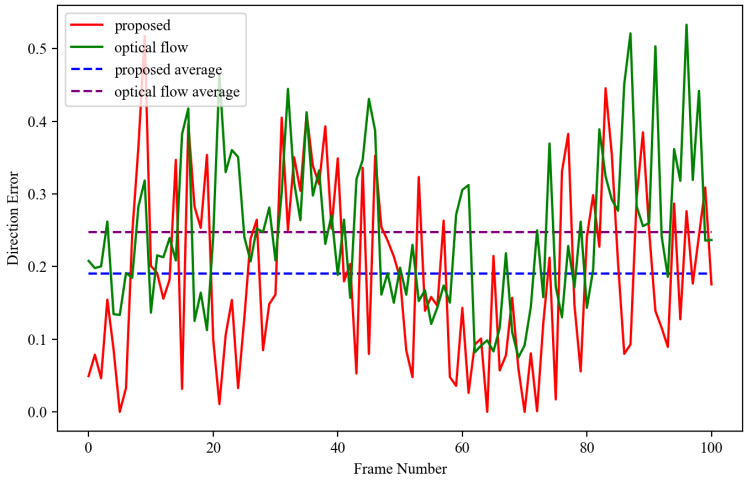
MAE_v and MAE_d are two metrics to evaluate the crowd motion detection performance. The ground truth motion data is calculated by averaging the motion pattern of all objects in each group. Groups are manually labeled based on whether they have a uniform motion pattern because a group of persons always have a similar movement pattern. Similar to object detection, a localization-based matching will be conducted between the ground truth and prediction. The ground truth will uniformly match only one predicted group. If the ground truth cannot match the prediction or the prediction cannot match the ground truth, the unmatched values of the group k for $MAE_{v,k}$ and $MAE_{d,k}$ can be written as $|v_k|$ and 1 correspondingly. The average speed of the ground truth is 2.6 meters per second in this dataset.

Table 4.7: MAE of Crowd Motion Detection

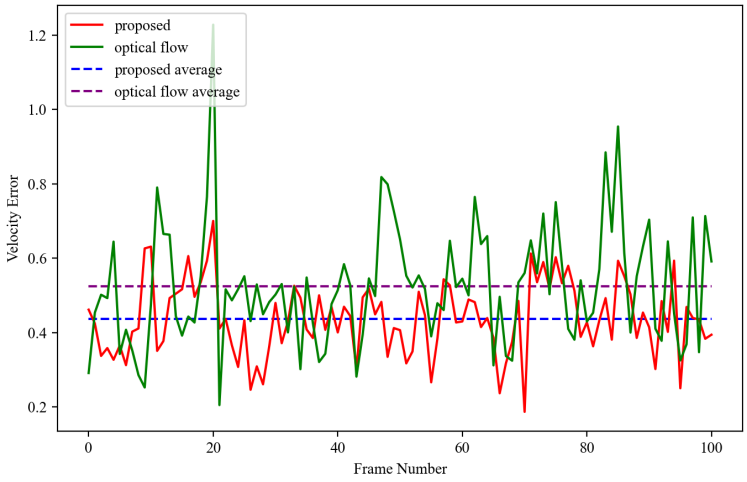
Methods	Zoom Stitcher	MAE_v	MAE_d
Optical Flow	-	0.58	0.29
Proposed	-	0.42	0.21
Proposed	✓	**0.40**	**0.20**

As a result, the evaluation results are shown in Table 4.7. The proposed method has lower MAEs than the optical flow method, indicating better detection accuracy in crowd motion detection. One reason that leads to low MAEs of the optical flow method is its low crowd coverage. Comparing the second and third rows, the proposed Zoom-Stitcher can benefit the motion detection accuracy in velocity and direction because SORT depends on the object detection results. In the future, more advanced tracking algorithms can further improve motion detection accuracy. This feature also implies the advantage of modular design by replacing one part to promote overall performance.

More specifically, this chapter provides a more straightforward way to observe the difference between the proposed and optical flow algorithms. Figure 4.12 includes two plots of direction and velocity errors based on 100 fixed-interval frames. This chapter randomly selects a starting point from an indoor MOT 2020 video and samples 100 frames at a sample rate of 10Hz. The observation duration is 10 seconds. The application scenario can refer to Figure 4.11. This scenario is complex because the environmental illumination is insufficient, and several groups have different motion patterns. Figure 4.12 depicts continuous values of velocity and direction errors besides static overall errors. The solid red line refers to the proposed method, and the solid green line refers to the optical-flow-based method. In addition, the average value, representing the overall error during the observation period, is shown in the dashed line, where the proposed method uses blue color and the optical-flow-based method



(a) Direction Error



(b) Velocity Error

Figure 4.12: Detection Error of the Proposed and Optical Flow Algorithms

uses purple color. Figure 4.12a shows the direction error and Figure 4.12b shows the velocity error. There is no additional smoothing operation on these plots. Comparably, the proposed method has a lower error and higher stability than the optical-flow-based method. This re-

sult produces similar conclusions to Table 4.7. The limitation of the optical flow mainly lies in the poor quality of feature corners. Hand-crafted descriptors cannot stably catch objects in complicated scenarios. They may be easily affected by high-contrast areas or regions with more texture information.

4.4 Chapter Conclusion

This chapter tries to improve the existing crowd monitoring system from the crowd counting, localization, and motion detection perspectives. More importantly, this studies data augmentation for tiny object detection, benefitting object counting and tracking directly. Crowd analysis plays a vital role in traffic safety and efficiency. Most previous works belonged to crowd simulation that succeeded in studying motion patterns in different facilities and crowd density and avoiding crowd disasters in advance. However, crowd simulation models are established based on various assumptions and predefined rules, which might not be as accurate as practical scenarios. Moreover, hyper-parameters in crowd simulation came from the real world. Thus, precise crowd detection methods are vital to the crowd simulation models. In addition, a real-time crowd detection method can help cope with emergent situations like abnormal crowd behavior detection. In general, crowd detection has three essential tasks – crowd counting, localization, and motion detection. However, previous methods only focus on one or two tasks mentioned above.

In conclusion, this chapter innovatively proposes an integrated deep learning framework using visual cameras for crowd counting, localization, and motion detection within one pipeline. This framework considers sparse and dense crowd conditions separately with different analysis strategies. This chapter designed two crowd counting models to handle different crowd

densities. A YOLO-based object detection model optimized for fast detecting tiny objects and crowd counting in a low-density situation. It promotes the inference speed by over 30% and keeps detection accuracy for tiny objects. This chapter develops a Zoom-Sticker strategy to enhance detection ability further and explore data variance at different scales. An efficient crowd counting model MSNet-F based on the FCN model with cascaded dilation blocks is designed for dense conditions and outperforms other state-of-the-art crowd counting methods in public crowd datasets. A switching strategy helps the framework adapt to different density conditions. In addition, this chapter expands the multi-object tracking method to detect the crowd motion with the assistance of a density map, which exceeds the optical-flow-based algorithm in both accuracy and stability.

5

Rare and Novel Traffic Object Detection

5.1 Overview

Chapter 4 has discussed one effective method to improve tiny object detection in the crowd monitoring scenario by applying Zoom-Stitcher data augmentation to increase the tiny object sample number. This chapter focuses on a more extreme situation when there are only a few samples for one category. In tiny object detection tasks, tiny objects still have hundreds

of samples. Additionally, tiny objects are more likely to belong to one category and only have scale differences. In the previous chapter, all small-scale objects are persons. However, some minor categories have different appearances and latent features. The data augmentation may not take effect in this situation because it can only increase a limited number of samples based on existing data. What is more, there are newly appeared traffic objects every day. It is impossible to give up the detection accuracy of these objects, which will lead to severe traffic safety events. Therefore, this chapter designs a FSL strategy to force the model to learn features from few-shot samples. Rare and novel object detection is challenging for the perception system in self-driving and ADAS. Improving detection accuracy on those objects can effectively elevate the safety level for autonomous vehicles. FSL can also be applied to other transportation analysis tasks facing extreme data imbalance.

5.1.1 Background

Data augmentation is one of the most common methods to increase data variance when the training data is insufficient. It has also become the standard operation in the deep learning training process. When there are only a few training samples for some categories, such as ten samples, data augmentation may not work. The deep learning model cannot learn sufficient information from those samples with the general loss function. The outcome will be extremely low accuracy in those categories even though the mean average precision is high. According to Cannikin's law, the system's overall performance depends on the weakest perspective. Low accuracy in minor categories will lower the robustness of the traffic sensing system. In the transportation application, false-positive detection may cause severe traffic safety issues. For example, it will be dangerous when autonomous vehicles fail to recognize

an obstacle as background when the barrier is rare in the training dataset. Deep learning algorithms have achieved impressive performance in the past ten years and surpassed conventional algorithms significantly. These deep learning algorithms even outperformed humans in some tasks. However, it is challenging to detect rare-category objects, such as wild animals in traffic datasets, that may appear only for a limited time in the training dataset. Unlike humans, who can learn a new object with a few samples, the machine requires more data to understand high-level representations. In this situation where partial categories have a few samples, the machine has a high probability of rendering a low classification or confidence score for those categories, leading to missing detection [320]. Data imbalance is common in machine learning studies because some categories lack annotated samples and are uncommon in the real world. One effective method is forcing the training process to pay more attention to rare categories, e.g., loss equalization [281]. Otherwise, detection algorithms tend to classify targets into those categories whose appearance frequency is higher.

5.1.2 Application Scenario – Rare Traffic Object Detection

Based on the platform's mobility, traffic sensing can have stationary and mobile modes. Previous two chapters, Chapter 3 and Chapter 4, discuss research works on the stationary platform of utilizing fixed cameras to monitor parking availability and crowd movement. This chapter will study traffic sensing on mobile platforms, e.g., autonomous vehicles. As representatives of the mobile platform, connected and autonomous vehicles have integrated advanced sensing technologies to detect the surrounding traffic environment and prepare input for behavior prediction and vehicle control. Besides fully self-driving vehicles, ADAS installed on conventional vehicles contributes to driving safety [370]. Self-driving or ADAS re-

lies on an accurate and robust perception system to provide detailed surrounding information [346]. Thus, incorrect detection can lead to false determination in the post-process and traffic safety risks. The perception system mainly has two components – sensing hardware and detection algorithm.

From the hardware perspective, cameras [176] are the most popular sensors compared to others, e.g., LiDAR [17] and radar [266]. The advantages of vision-based sensors are relatively low cost and mature support algorithms. Tesla only relies on vision-based sensors in their autonomous vehicles [280]. Waymo selects multi-sensor fusion by integrating camera and LiDAR [274]. After the cost of LiDAR decreases, sensor fusion may become mainstream because the point cloud from LiDAR can help reconstruct the 3D local environment for better scenario understanding. Nowadays, there are massive demands for accurate, robust, and efficient vision-based traffic object detection. The detection performance will affect post processing, e.g., trajectory prediction and obstacle avoidance. These operations are highly related to traffic efficiency and safety.

The primary reason causing low accuracy in rare and novel categories is the lack of annotated data for specific categories. Annotated data imbalance is a common phenomenon related to appearance frequency. It will take a lot of effort to collect and annotate these rare objects following conventional deep-learning training principles to balance the training data. Ignoring these rare objects is also unacceptable as they will increase traffic safety risks. Chapter 4 has stated that self-supervised learning is not suitable for transportation applications and designed a data augmentation for tiny object detection. This chapter works on a more challenging detection task of rare and novel object detection. This chapter pays attention to FSL, targeting to learn from a limited number of samples with supervised information [307].

FSL belongs to supervised learning but uses prior knowledge to improve performance on rare categories. Compared to self-supervised learning, FSL is more promising for transportation applications because it is easier to deploy and transfer to different knowledge domains. Specifically, the output of FSL can be directly applied for target tasks [120] without extra steps like self-supervised learning.

5.1.3 Contributions and Organization

Novel and rare traffic object detection is challenging but significant to traffic safety. Improving detection accuracy in these minor categories can boost the overall system robustness. Since there are only a few samples for them, the conventional deep learning training framework will lead to enormous bias on uncommon categories because they contribute little to the training loss. Deep learning models will try to learn features to reduce the total loss. Thus chapter proposes an effective few-shot object training framework and a lightweight detection algorithm to achieve accurate and efficient detection in common and rare categories for mobile platforms. Following Chapter 3 and Chapter 4, this chapter also consider efficiency as an important factor when constructing the deep learning detection model. There are three significant contributions of this chapter.

a) Develop A few-shot traffic object detection algorithm with a promising performance in accuracy and efficiency for simultaneously detecting common and rare categories. The few-shot detection algorithm contains the base model and training framework. The base model is a one-stage detector based on the recently released YOLO model. The training framework inherits from the conventional framework but adds one extra fine-tuning step on a balanced downsampling dataset with a specifically designed

similarity function.

b) Evaluate the effectiveness of FSL for detecting traffic objects when facing rare categories. This chapter has two experiments conducted on self-driving and traffic sign recognition datasets. The first experiment is the general traffic object detection task. Two kinds of animals on the road are added to this dataset as novel categories. The second experiment is the traffic sign detection task. Unlike the previous task, this one is the fine-grained classification task by recognizing the content of each traffic sign.

c) Explore intra-class variance's impacts on detection accuracy. Most previous research has ignored the effects of intra-class variance and worked on pre-defined categories using public datasets. However, it is crucial to determine whether the existing categories are appropriately defined. Thus, this chapter conducts a novel and practical evaluation about intra-class variance using t-SNE embedding visualization.

The rest of this chapter will develop as follows. Section 5.2 discusses the methodologies about the few-shot training strategy and an efficient object detection model. Firstly, Section 5.2.1 introduces a few-shot training pipeline to improve model accuracy on rare and novel traffic object detection. This training pipeline has two steps: training on base categories and fine-tuning on a small dataset containing all categories. Secondly, an accurate and efficient one-stage object detection model is proposed for multi-scale object detection in Section 5.2.2. The detection model adopts YOLO v5 as the base model and introduces cascaded dilated blocks to reduce the computing complexity but keep the accuracy. Section 5.3 will describe two experiments of traffic object detection and sign recognition. This section compares the proposed model and training strategy with state-of-the-art works. Experiments evaluate the

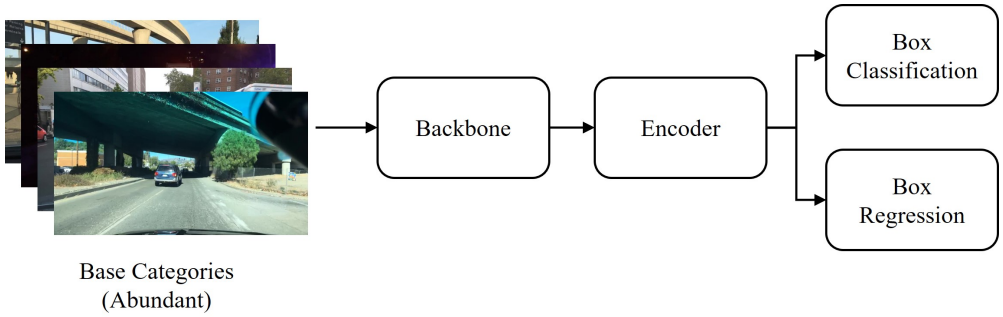
accuracy of the proposed model on base and rare categories. Since the proposed detection model targets to run on the edge side, the inference speed is also compared in the first experiment in Section 5.3.2. In addition, the impacts of intra-class variance on accuracy are explored in Section 5.3.4.

5.2 Methodology

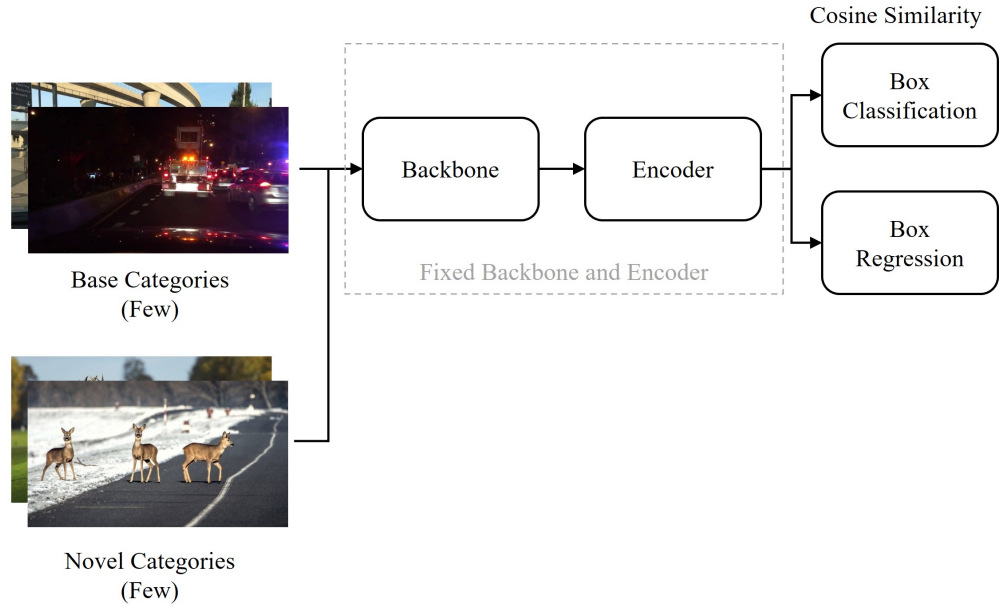
5.2.1 Few-Shot Training Framework

Previous research works in FSL are based on meta-learning that requires support images in a query. These algorithms compute the similarity between embeddings from the input and support images and make the model learn from these similarities. Such a framework is complicated because of adding a different input branch. Instead of modifying the network structure as meta-learning, this chapter adopts a two-step training strategy shown in Figure 5.1, where Figure 5.1a is the first step of training the model on base categories. Figure 5.1b is the second step of fine-tuning the pre-trained model on both base and novel categories using a downsampling dataset.

The first step is to train on base-category samples as shown in Figure 5.1a. It follows the same strategies as general object detection model training, including data augmentation and normalization. Since base categories always have sufficient data, this step does consider the extreme data imbalance situation. Focal loss [181] is still preferred as the loss function since the object number is the same. This step tries to help the model adapt to the target domain and learn enough base knowledge from training data. Thus, this model can rapidly learn new knowledge from limited samples in the next step. The second step is fine-tuning the pre-trained model on few-shot samples with specifically designed cosine similarity. Compared



(a) Training on Base-Category Samples



(b) Fine-Tuning on Few-Shot Samples

Figure 5.1: Two-Step Training Strategy for Few-Show Object Detection

to computing the distance between the prediction and ground truth, cosine similarity can better utilize prior knowledge obtained in the first step. There are three critical points in the second step as follows.

The first point is requesting **balanced few-shot sample images**. In Figure 5.1b, both base and novel-category objects will be used for model training. The sample number of each cat-

egory is the same. Previous works set that number as 1, 2, 3, 5, and 10 for evaluation. In practice, this number should be as large as possible. The second is **fixing backbone and encoder**. Since the fine-tuning samples are limited, it is impossible to unfreeze all. Otherwise, there will be an underfitting problem, and the training cannot converge. Only the final box classification and regression weights will be updated like other fine-tuning jobs. The third one is applying **cosine similarity**. Suggested by Spyros Gidaris[94] in few-shot classification tasks, the cosine similarity has an impressive performance and is applied in bounding box classification. Suppose the weight matrix of the box classification is $[w_1, w_2, ..., w_c]$, where w is the per-class weight. The classification outputs are scaled based on similarity scores between the i-th object proposal and the weight for class j written in Equation 3-1, where x is the input feature representation of i-th object proposal, and α is a hyper-parameter and set as 20 in this chapter.

$$s_{i,j} = \frac{\alpha x_i^T w_j}{\|x_i\| \|w_j\|} \tag{5.1}$$

5.2.2 Traffic Object Detection Model Framework

The object detection algorithm is based on YOLO v5, simultaneously achieving high efficiency and accuracy. It can be seen as a PyTorch implementation of YOLO v4[24]. YOLO series can provide an impressive inference speed and keep a satisfying accuracy. The newest version inherits its one-stage tradition but adds more novel features from other state-of-the-art research works. Its predecessor is YOLO v3 [235] that learns from the feature pyramid structure to achieve multi-scale object detection. There are several improvements in the network framework design and training strategies. Spatial Pyramid Pooling (SPP) is adopted to im-

prove multi-scale feature fusion using different pooling sizes[109]. SPP can generate the same dimension representations from different input sizes and increase the robustness to object deformation without extra computing cost. Beyond FPN, it used Path Aggregation Network (PAN) to generate multi-scale feature maps[187]. PAN can provide deeper feature maps for prediction compared to FPN. Besides the network itself, effective training strategies play a vital role. One representative strategy is improving CutMix[342] via mixing several training images to help detect objects outside their normal context. Furthermore, it can reduce the necessity for a large batch size because batch normalization computes activation statistics from 4 four different images.

YOLO v4 and v5 have great success in different computer vision tasks. Compared to other computer vision tasks, traffic object detection has fewer object categories. The targets mainly belong to traffic tools and signs. The newest YOLO may be still too complicated for this task. Inspired by YOLOF[40], this chapter introduces single-in-single-out structure with dilated convolution. YOLOF does not use feature pyramid structure but explore multi-scale features from outputs of the last layer of the backbone. The outputs from the backbone's last layer may be still too shallow. In order to increase the depth but avoid the complex PAN or FPN structure, Top-Down Modulation (TDM) is selected for multi-scale feature fusion[90]. As a result, the detection framework is shown in Figure 5.2, where the dilated module is described in Figure 5.3.

Backbone: This chapter adopts CSPNet as the backbone[297], which has fewer parameters than ResNet but stronger feature extraction ability.

Encoder: The encoder has two parts – TDM and dilated module. TDM uses concatenation instead of addition in FPN and PAN to fuse different feature maps. The dilated module

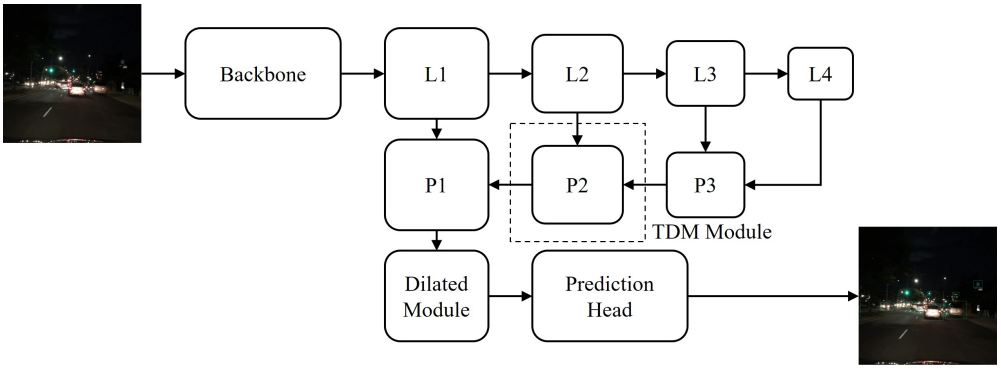


Figure 5.2: Proposed Detection Algorithm Framework

can explore multi-scale features in one single pass. Figure 5.3 shows the framework of the dilated module containing four successive residual blocks with different dilation factors. In this chapter, the dilation factors are 2, 4, 6, and 8. The first 1×1 convolution reduces the channel number to 1/4 of the original number. The 3×3 convolution is dilated convolution to increase the receptive field. The last 1×1 convolution recovers the channel number. Thus, the input and output have the same number of channels, which provides the convenience of inserting or appending other modules. This bottleneck design can significantly reduce computing complexity and is demonstrated in other models, e.g., ResNet.

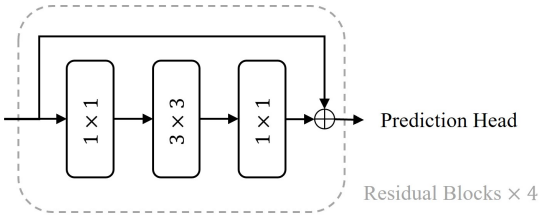


Figure 5.3: Dilated Module Framework

Decoder: The decoder is similar to YOLO v4 but only generates predictions from one feature map. Based on Figure 5.2, the prediction head accepts the output from the dilated module. The dilated module consists of cascaded dilation blocks as plotted in Figure 5.3.

The original YOLO model predicts different-scale objects from corresponding feature layers. Benefitting from cascaded dilation blocks, this model still explores multi-level features. Another minor change following FPN is the different number of convolution layers in classification and regression branches. The former has two convolution layers, and the latter has four layers.

5.3 Experiments

5.3.1 Experiment Setting

The experiment section focuses on general traffic object detection and traffic sign detection tasks using onboard cameras. Unlike previous research work in these two tasks, this chapter pays more attention to increasing the detection accuracy of rare categories without affecting major categories. Two experiments follow the same experiment procedure. Firstly, two experiments will compare the proposed method with state-of-the-art algorithms on the target dataset, excluding rare categories. Secondly, they will add rare categories and compare these algorithms in terms of accuracy on the rare or novel categories.

Traffic object detection using onboard cameras is essential for autonomous vehicles and is highly related to driving safety. The primary dataset for model evaluation is Berkeley DeepDrive (BDD), containing 100,000 well-annotated images including ten categories and covering different weather conditions and scenarios[338]. Figure 5.4 shows the category distribution in the BDD dataset. According to Figure 5.4, some categories occupy a small portion of all annotated objects. For example, *train* only occupies 0.01%. The long-tailed problem has already existed and should be addressed. However, some rare categories, e.g., animals, are still lacking in BDD and other similar datasets, e.g., KITTI[91] and UrTra2D[135]. Although it is

uncommon for medium or large animals to appear in the urban area, it will be super dangerous when failing to recognize them for autonomous vehicles. In addition, these animals have a higher probability of appearing on the rural road. This experiment has extended the BDD dataset by adding images of animals on the road. Instead of collecting large samples for training, only a few samples are used to simulate the practical situation of low appearance frequency in the training dataset. As a result, this chapter targets to provide a beneficial deep learning framework to detect novel categories with limited samples and model training efforts.

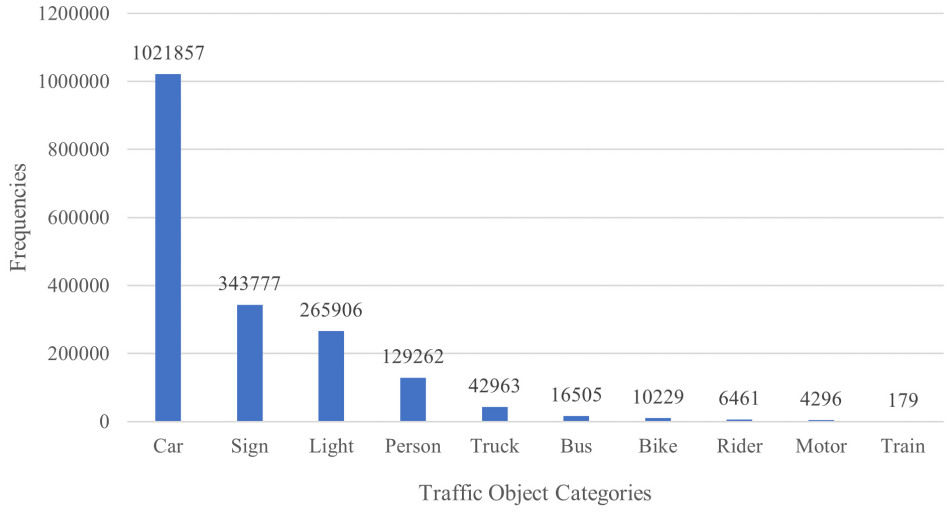


Figure 5.4: BDD Dataset Category Distribution

One research objective of this chapter is to focus on detecting rare objects in transportation scenarios, especially animals. Thus, 40 sample images containing animals in the street, e.g., deer and dog, are collected from the internet. Two sample images of small and medium animals on the road are shown in Figure 5.5, which do not show up in any autonomous vehicle dataset. These situations may be uncommon but vital to traffic safety for autonomous vehi-

(a) Small Animals on the Road

(b) Medium Animals on the Road

Figure 5.5: Sample Images of Animals on the Road

cles replying vision-sensor for object detection and obstacle avoidance. All collected images are cropped to 640 × 360 instead of resizing to ensure that no object is distorted.

The compared state-of-the-art algorithms include CFENet[355], Faster R-CNN, MultiNet, and DLT-Net[228] for this traffic object detection task. The evaluation metrics are Recall, Average Precision (AP), and inference speed, covering both accuracy and efficiency. The Intersection of Union (IoU) threshold for AP is 0.5. The training and testing platform equips with an Nvidia Titan XP graphics card. Limited by the GPU memory, the training process adopts gradient accumulation and mixed-precision strategies. The initial learning rate is 0.0001 with cosine annealing. The original image size from the BDD dataset is 1280 × 720 and is resized to 640 × 360 for training and evaluation. The data augmentation follows the same strategies in YOLO v4. The essential loss function for the detection model training is shown in Equation 5.2 and is composed of three parts – L_{cls}, L_{obj}, and L_{iou}. L_{cls} and L_{obj} are a loss for confidence scores and objectiveness scores, respectively, which are based on focal loss[181] addressing the data imbalance and forcing the algorithm to concentrate on hard samples. The definition of focal loss is shown in Equation 5.3, where γ is the hyperparameter and p_t is the predicted score. Training may become insufficient when using cross-entropy

loss since most samples belong to easy negatives. Those easy negatives make it difficult for a model to learn rich semantic information. Comparably, the focal loss can reduce this negative impact. Besides, L_{iou} is the loss for bounding boxes and adopts CIoU[359] considering overlap area, central point distance, and aspect ratio.

$$L = \alpha_1 L_{cls} + \alpha_2 L_{obj} + \alpha_3 L_{iou} \tag{5.2}$$

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{5.3}$$

The former experiment dataset concentrates on detecting base and novel objects on the road, which is essential to collision avoidance. Traffic sign detection and recognition is another fundamental detection task for autonomous vehicles and can contribute to the traffic asset management[368]. Autonomous vehicles can obtain road information from traffic signs and digital maps. However, that road information may be unavailable or not updated on digital maps. If vehicles only rely on digital maps to get road information, they are more likely to violate traffic rules without any notice. Benefitting from public street images, traffic agencies can collect and update the traffic sign status periodically. The automatic traffic sign recognition system can reduce the labor cost remarkably. One challenge for traffic sign detection is that some traffic signs are uncommon and have fewer samples for model training. Like traffic object detection in the previous dataset, the low appearance frequency may decreasee the detection accuracy for those uncommon traffic signs. There is also a need for improvement in uncommon traffic sign detection. Thus, the second experiment on traffic sign detection with few-shot samples targets to evaluate the proposed algorithm's performance in uncom-

(a) Traffic Sign in Urban Area **(b)** Traffic Sign in Rural Area

Figure 5.6: Sample Images of Traffic Signs

mon traffic signs. The traffic sign dataset for model training and evaluation is DFG Traffic Sign Dataset[279] containing 7,000 images with 13,000 annotated objects in 200 categories. Sample images of traffic signs captured in urban and rural areas are shown in Figure 5.6. The original images have two sizes of 720×576 and 1920×1080, and are preprocessed to the same size following different resizing rules. 720×576 images are firstly center cropped to 720×405 and then resized to 640×360. 1920×1080 images are directly resized to 640×360. The category distribution is displayed in Figure 5.7, where the maximum and minimum frequencies are 705 and 20. Most traffic sign categories have a low appearance frequency in this dataset.

5.3.2 Evaluating General Object Detection Accuracy and Efficiency

The proposed and other algorithms are trained and evaluated on the standard BDD dataset without considering novel categories. This experiment tries to evaluate the general performance of the proposed algorithm compared to other state-of-the-art algorithms. The training process is general deep learning training without FSL optimization. The experiment results are shown in Table 5.1, where the proposed algorithm achieved the highest accuracy and ef-

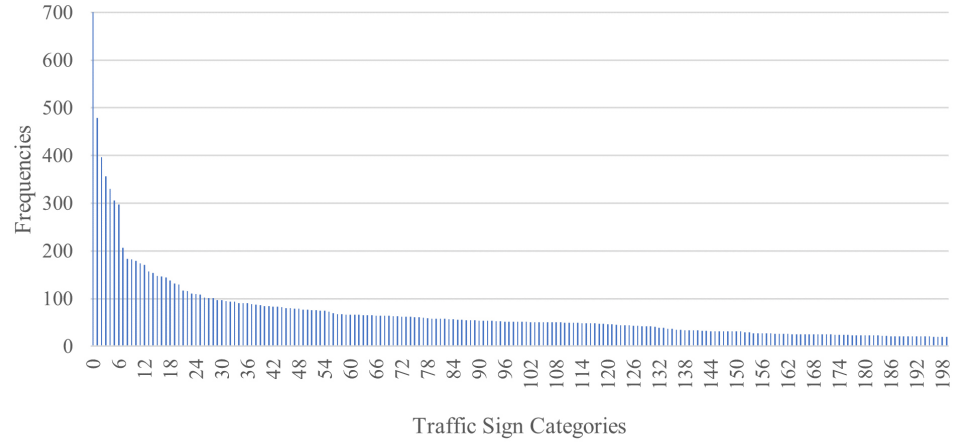


Figure 5.7: DFG Traffic Sign Dataset Category Distribution

ficiency. It outperforms its base model YOLO v5, which demonstrates the effectiveness of the dilated module in predicting multi-scale objects. To be more specific, the inference speed of the proposed algorithm is also 10% faster than YOLO v5. Recall and AP are 2.2% and 1.5% higher, respectively. Compared to the previous most accurate algorithm DLT-Net, the proposed one has slightly higher accuracy with an increased 0.3% in Recall and 1.3% AP. Meanwhile, the inference speed of the proposed model is over four times faster than DLT-Net.

Table 5.1: Comparison of Traffic Object Detection on BDD Dataset

Algorithms	Recall (%)	AP (%)	Speed (FPS)
CFENet	75.1	53.7	21.0
Faster R-CNN	77.2	55.6	8.8
MultiNet	81.3	60.2	8.6
DLT-Net	88.7	62.7	9.5
YOLO v5	86.8	62.5	45.0
Proposed	**89.0**	**64.0**	**50.1**

The visualization of detection results from a complex night scenario is shown in Figure

5.8, where Figure 5.8a is the ground truth while other figures represent three top algorithms – DLT-Net in Figure 5.8b, YOLO v5 in Figure 5.8c, and the proposed algorithm in Figure 5.8d. The environment light is dim and surrounding agents are small in these figures. This visualization matches the results in Table 5.1. The proposed algorithm can detect more objects and generate accurate target localization. In addition, it can detect partially occluded objects that are not annotated in the ground truth. Compared to its base model YOLO v5, the proposed algorithm is more accurate and can find hard samples. However, these algorithms fail to detect some traffic objects in the dark area due to a lack of texture information under this low-light condition. Detecting object in low-light conditions is always challenging but not the research objective of this chapter. Chapter 6 will discuss how to improve the accuracy in adverse conditions by applying sensor fusion.

(a) Ground Truth **(b)** DLT-Net

(c) YOLO v5 **(d)** Proposed Algorithm

Figure 5.8: Visualization of Traffic Object Detection Results

5.3.3 Evaluating Rare/Novel Object Detection Accuracy

The previous experiment compares the proposed algorithm with state-of-the-art algorithms and demonstrates excellent performance on conventional detection tasks. This part will evaluate the performance of novel categories, i.e., animals on the road, in this chapter. Most images in existing autonomous vehicle datasets are collected in the urban area. However, some animals may show up on the road in the wild or in rural areas and threaten traffic safety, especially when the visibility distance is short. Thus, this chapter considers animals as novel categories. Firstly, *train* is considered as the novel category according to Figure 5.4, while the remaining nine are base categories. Secondly, two extra categories, e.g., *dog* and *deer*, are added for this few-shot object detection experiment. These two animals are more common to appear on the road. As a result, there are nine base categories and three novel categories in total.

There are two training strategies in this experiment – general and fine-tune training. General training indicates training on mixed samples images containing base and novel categories. In this part, the total category number will be 12, including nine base categories and three rare categories. There is no particular attention to rare categories in loss function computation. Fine-tune training is a two-step strategy mentioned in Section 5.2.1. The first step only trains the algorithm on base-category sample images. The second step samples K images equivalently, also called K-shot, from each category to fine-tune the algorithm. Considering the limited sample number in this step, training the whole model will lead to the underfitting problem. Therefore, only the weights of the prediction head are updated. The backbone and encoder are fixed. The evaluation results using two training strategies on novel and base categories are shown in Table 5.2 and Table 5.3.

Table 5.2: Comparison of Few-Shot Object Detection on Novel Categories

Algorithms	Training	AP (%)				
Shot		1	2	3	5	10
CFENet	General	0.0	0.0	1.5	1.6	2.0
Faster R-CNN	General	0.3	0.3	1.9	2.0	2.7
MultiNet	General	0.3	0.3	1.9	1.9	2.6
DLT-Net	General	0.4	0.4	2.2	2.4	2.6
YOLO v5	General	0.3	0.3	2.0	2.2	2.6
Proposed	General	0.5	0.5	2.4	2.8	3.0
CFENet	Fine-Tune	25.2	30.3	33.4	36.7	40.1
Faster R-CNN	Fine-Tune	26.5	31.5	34.9	38.0	42.2
MultiNet	Fine-Tune	27.0	33.1	35.2	39.5	43.5
DLT-Net	Fine-Tune	28.3	35.5	37.1	39.9	43.2
YOLO v5	Fine-Tune	28.0	34.7	36.8	39.6	43.9
Proposed	Fine-Tune	**32.2**	**37.4**	**42.1**	**47.4**	**50.0**

According to Table 5.2, the proposed algorithm shows an extraordinary performance compared to previous algorithms. The experiment results demonstrate a necessity to apply the FSL strategy when there are novel or rare objects with a limited sample number. The comparing results between general and fine-tune training show the negative impact of data imbalance. In general training, the data is highly imbalanced, where novel-category objects occupy around 0.02% of all annotated objects. The accuracy is much higher when applying the fine-tune training strategy, even using the same number of novel-category samples. The primary reason is the balanced samples in the second step of training the prediction head. Figure 5.9 shows the relationship between a few-shot number and AP when applying different detection algorithms under the second training strategy. It can be found that the proposed algorithm has a much higher AP. More sample images used for training in the fine-tuning stage can significantly improve accuracy. In theory, the algorithm can reach the best when there are as many images of novel categories as base categories. However, requesting sufficient

sample images is contrary to the base of FSL. According to Figure 5.9, the AP rising speed becomes slower when the sample number is larger than five. A few-shot number of five in this experiment is a good option for model training.

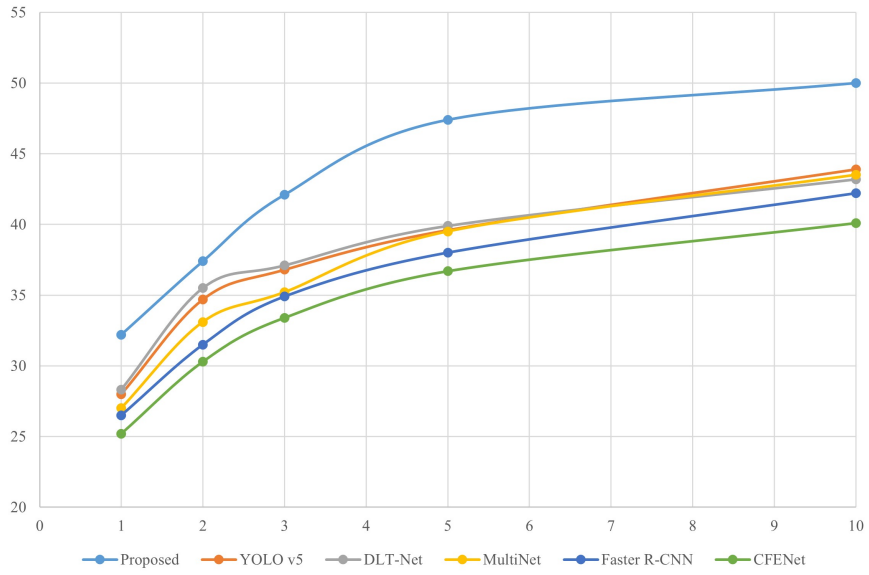


Figure 5.9: Few-Shot Number versus AP on Novel Categories

Table 5.3 shows the experiment results evaluated on base categories. Previous algorithms using the general training strategy have the best and most stable performance because models may ignore novel-category samples during the training process. These minor samples have almost no contribution to weight updates. The difference between base and novel categories is so significant that models have a higher probability of predicting the proper categories even when they randomly select base categories. Compared to Table 5.3 and Table 5.1, the accuracy of these algorithms increases due to the removal of *train* from base categories. The fine-tune training strategy performs worse than the general training strategy due to the limited number of samples to train the prediction head. With the increase of samples, their accu-

Table 5.3: Comparison of Few-Shot Object Detection on Base Categories

Algorithms	Training	AP (%)				
Shot		1	2	3	5	10
CFENet	General	53.7	53.7	53.6	53.6	53.6
Faster R-CNN	General	55.8	55.8	55.8	55.8	55.7
MultiNet	General	60.2	60.2	60.2	60.2	60.2
DLT-Net	General	62.7	62.7	62.7	62.7	62.7
YOLO v5	General	62.5	62.5	62.5	62.5	62.4
Proposed	General	**64.0**	**64.0**	**64.0**	**64.0**	**63.9**
CFENet	Fine-Tune	33.2	38.3	42.4	45.5	48.4
Faster R-CNN	Fine-Tune	33.5	39.6	42.0	45.7	49.5
MultiNet	Fine-Tune	35.0	39.5	43.1	47.9	52.3
DLT-Net	Fine-Tune	37.3	41.9	43.5	49.5	55.0
YOLO v5	Fine-Tune	37.3	41.6	43.0	49.1	54.9
Proposed	Fine-Tune	40.9	44.4	48.2	55.0	60.8

racy rises but is still lower than using general training. The applied fine-tune training strategy tries to shorten this difference. Under the 10-shot condition, the proposed algorithm with the fine-tuning strategy has a very close accuracy to its general training performance. The AP is only 3.2% lower, demonstrating that the proposed algorithm can keep the detection accuracy when introducing new categories.

The plot in Figure 5.10 shows the trend between the few-shop number and AP on base categories. AP is also low when the few-shot number is low because the new prediction head has little knowledge about these objects, similar to the phenomenon of novel categories. As the few-shot number rises, AP increases rapidly. From the few-shot number of one to five, AP increases by around 15%. After that, the increasing speed slows down. Comparing Figure 5.9 and Figure 5.10, the few-shot number of five is enable the model to have a satisfying performance on both base and novel categories. In general, more samples are always better. However, these plots can help determine how many samples should be collected and anno-

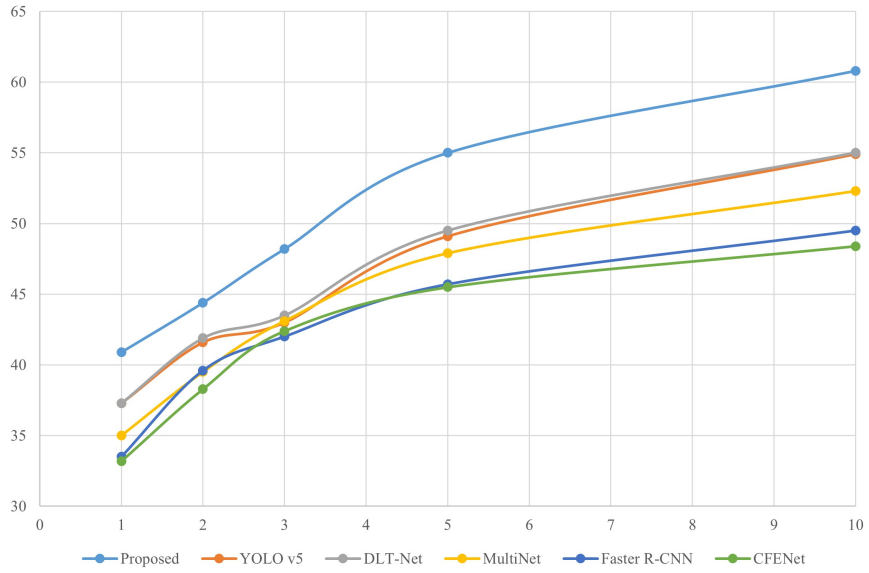
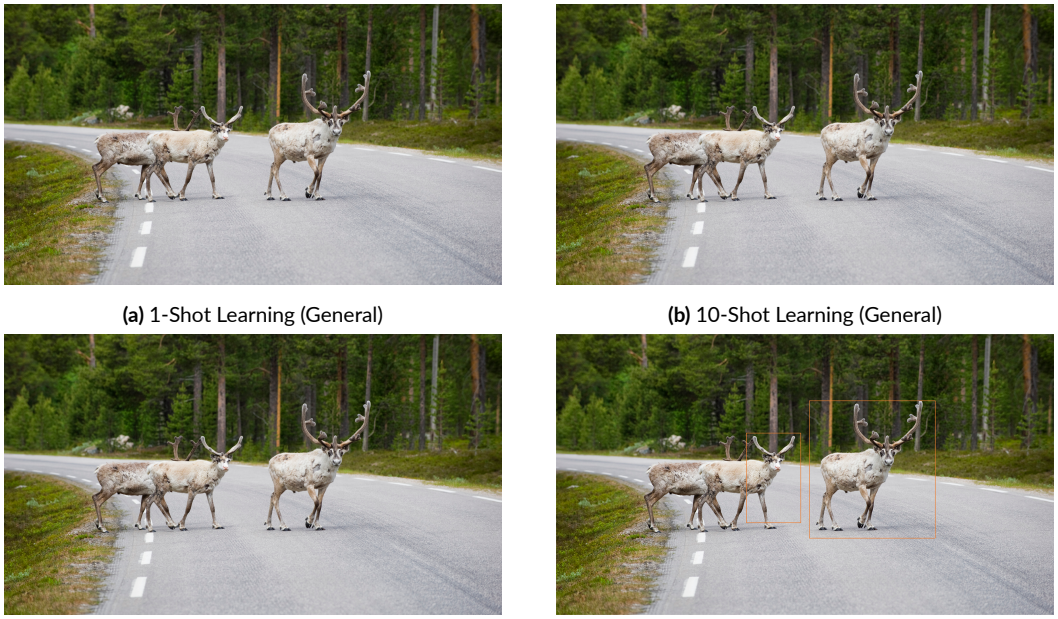


Figure 5.10: Few-Shot Number versus AP on Base Categories

tated under limited budgets and time.

Besides quantitatively compare model and training strategies, Figure 5.11 and Figure 5.12 also qualitatively analyze them by visualizing the prediction results. Figure 5.11 shows the detection results of DLT-Net on the novel category, where the first column is 1-shot learning and the second column 10-shot learning. Figures in the first row adopt the general training strategy, and others embrace the fine-tune training strategy. It is safe to conclude that the fine-tune strategy can significantly improve the detection accuracy using a few samples. Data balance is essential for model training when comparing the first and second rows. In the first row, novel-category objects only occupy a tiny portion of all training samples. The loss function easily ignores these samples even though the focal loss is applied. In contrast, the second row uses a small but balanced dataset for model fine-tuning. Different categories have a similar appearance probability in this downsampling dataset. Figure 5.12 shows the detection



(a) 1-Shot Learning (General)

(b) 10-Shot Learning (General)

(c) 1-Shot Learning (Fine-Tune)

(d) 10-Shot Learning (Fine-Tune)

Figure 5.11: Novel-Category Object Detection with DLT-Net

results with the proposed algorithm. Benefitting from the dilated module, it can generate more precise bounding boxes. When comparing Figure 5.11d and Figure 5.12b, two models, DLT-Net and the proposed model, are trained using the same strategy and can detect most targets. This visualization shows the generalization of the FSL training strategy, which can be deployed for different models. Additionally, the proposed model has a more accurate target localization and bounding box size. The selection of YOLO v4 as the base model and further modification is effective.

5.3.4 Exploring Intra-Class Variance

This chapter also explores the impacts of intra-class variance on accuracy. According to Figure 1.1 of building a deep learning system, the initial step includes data preparation. Data

(a) 1-Shot Learning **(b)** 10-Shot Learning

Figure 5.12: Novel-Category Object Detection Using the Proposed Algorithm

annotation is essential when facing specific categories that do not appear in previous self-collected and public datasets. Before starting to annotate data, it is necessary to make a clear data annotation rule. It is struggling to define how many categories should be created. Creating high-level categories, e.g., vehicles (all) and animals, can reduce the difficulty of this annotation task but increase the intra-class variance. In contrast, creating low-level categories increases labor costs and complicates the annotation rules. Thus, figuring out how intra-class variance influences accuracy is necessary. A simple experiment is conducted by merging some categories and then creating two new categories – *vehicle* and *animal*. The *vehicle* category includes *car*, *truck*, and *bus*. The *animal* category includes *dog* and *deer*. The experiment results are shown in Table 5.4 using the proposed algorithm with 10-shot samples. The evaluation metric is AP with an IoU of 0.5. According to Table 5.4, the proposed algorithm, after creating the *vehicle* category, has the highest accuracy in base categories. When creating the *animal* category, AP on novel categories drops remarkably. The experiment results reveal that high intra-class variance can generate negative impacts. Therefore, combining categories with low intra-class variance may promote performance, while combining categories with a high intra-class variance will lead to the opposite.

Table 5.4: Impacts of Intra-Class Variance on Accuracy

Algorithms	Base Categories (%)	Novel Categories (%)
None	60.8	50.0
vehicle	**61.5**	**50.1**
animal	60.8	47.6
vehicle & *animal*	61.4	47.8

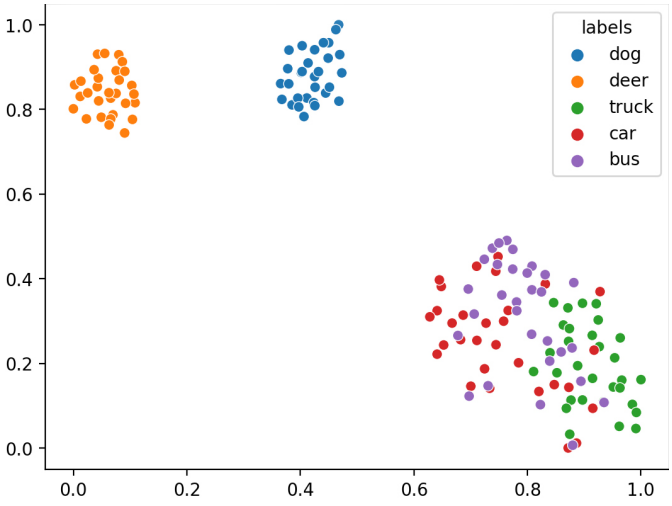


Figure 5.13: t-SNE Embedding Visualization

In addition, the visualization of feature embeddings from different categories are shown in Figure 5.13 using t-SNE[292]. These embeddings are extracted from the output of the classification module in the prediction head before the final fully connected layer. The object number of each category is around 30. In Figure 5.13, the *deer* and *dog* categories have longer distances than others, indicating a higher intra-class variance. The embeddings of *truck*, *car*, and *bus* in this figure are mixed and have a lower intra-class variance. Synthesizing Table 5.4 and Figure 5.13, there is almost no impact on AP when combining low intra-class categories.

5.3.5 Evaluating Traffic Sign Detection Accuracy

Besides the previous experiment in general traffic object detection, this experiment on traffic sign detection works as a supplement to further demonstrate the effectiveness of the proposed framework. Thus, this experiment evaluates traffic sign detection accuracy on all categories and novel categories following the same procedure. According to Figure 5.7, base categories are defined as traffic signs with a sample number greater than 30, and the remaining categories belong to rare or novel categories. Consequently, there are 152 categories belonging to the base category and 48 categories belonging to the novel category. To make it clear, the training strategy "General" represents training detection models on the complete dataset without FSL optimization. "10-Shot" is the two-step FSL training strategy described in Section 5.2.1 with 10 training images for each category. The evaluation metrics include Recall and AP with an IoU threshold of 0.5.

Table 5.5: Comparison of Traffic Sign Detection on DFG Traffic Sign Dataset

Algorithms	Training	All Categories		Novel Categories	
		Recall (%)	AP (%)	Recall (%)	AP (%)
Faster R-CNN	General	93.8	92.4	86.1	85.2
Proposed	General	**95.9**	**95.0**	88.0	87.7
Faster R-CNN	10-Shot	92.1	90.5	88.3	87.1
Proposed	10-Shot	95.5	**95.0**	**92.4**	**91.3**

The traffic sign detection experiment results on all and novel categories are shown in Table 5.5. Using the 10-shot fine-tune strategy, the proposed algorithm achieves the highest Recall and AP on novel categories. Comparing the results with the same detection algorithm, e.g., Faster R-CNN, but using different training strategies, fine-tuning the pre-trained model on a small balanced dataset can improve the performance on novel categories significantly. The

results also demonstrate that data balance is a vital factor contributing to final training results, especially for categories with fewer samples. In addition, the fine-tune strategy may slightly decrease the accuracy of base categories because the last layers for regression and classification are randomly initialized. The accuracy drop can be ameliorated by adding more data in the second step. The proposed training strategy can restrain such phenomena effectively and increase the overall accuracy. Two sample images of detection results in the same scenario are shown in Figure 5.14, the upper traffic sign belongs to the base category, and the lower one is the novel category. Figure 5.14a uses the proposed model and Figure 5.14b uses Faster R-CNN. Both models are trained with the 10-shot strategy. This sample image pair qualitatively compares Faster R-CNN and the proposed detection model. The proposed model can predict more precise bounding boxes. The Faster R-CNN model misses the novel category while the proposed algorithm has a more accurate detection. Furthermore, the accuracy difference between the general and few-shot training is not as large as the difference in the previous experiment. One possible reason is that the texture information is similar among different signs, indicating similar latent features.

(a) Proposed Algorithm **(b)** Faster R-CNN

Figure 5.14: Visualization of Traffic Sign Detection Results

5.4 Chapter Conclusion

The popularity of AI and deep learning have empowered the transportation system and built the foundation of ITS. Computer vision technologies have transferred and provided an automatic traffic sensing method and produced promising results in different sensing tasks to improve traffic safety and efficiency. Deep learning heavily relies on abundant training data to fit its tremendous number of parameters. Data sufficiency and balance can generate significant impacts on its final performance. However, some objects are uncommon and may appear a few times in the training data, which will lead to unavoidable ignorance. Failing to detect and recognize these objects may generate severe traffic safety problems. For example, autonomous vehicles distinguish animals into the background and do not take avoidance action. Such predicted situations are unaccepted. Therefore, this chapter focuses on detecting rare or novel-category objects with limited samples.

There are two research objectives – developing an effective traffic object detection algorithm and improving its performance on rare and novel-category objects. Standing on previous achievements in object detection and FSL, this chapter builds an accurate and efficient one-stage multi-scale object detection algorithm. A two-step few-shot training strategy is applied to cope with the challenge of limited training samples. The experiments have two parts – traffic object detection and traffic sign recognition, demonstrating the proposed algorithm's outperformance compared to state-of-the-art algorithms. In addition, this chapter also explores how intra-class variance affects the detection performance using the BDD dataset. This work can potentially expand to broader applications. Combined with the language model, e.g., CLIP[230], the proposed algorithm can achieve zero-shot object detection

to find new objects, contributing to data mining and semi-automatic data labeling.

6

Pedestrian Detection in Adverse Conditions

6.1 Overview

Traffic sensing in adverse conditions is essential but challenging since it operates 24/7 in all weather and light conditions. According to the previous discussion in Section 1.2.3, these adverse conditions will cause different negative impacts on sensing performance. However, traffic crashes are more likely to occur in those conditions. It is essential to deploy a robust

traffic sensing algorithm for stationary and mobile platforms in all conditions. Compared to other sensors, vision-based traffic sensing technologies are more sensitive to these adverse conditions. Specifically, low light will shorten the visible distance and add extra noise to RGB images. Objects will be overwhelmed by background noise when using RGB cameras. Multispectral image fusion will be an optimal solution to this challenge. Thus, this chapter designs an efficient and accurate multispectral object detection network – Illumination and Temperature-Aware Multispectral Network (IT-MN). The proposed multispectral detection model is built on the one-stage detector. There are four critical components in the proposed model – Dual-Input SSD (D-SSD), Fusion Weight-Computation Network (FWN), fusion network, and default box generation. Since multispectral object detection can contribute to onboard sensing and edge computing systems, model quantization is also applied to compress the model size and increase the inference speed by following steps introduced in Chapter 3.

6.1.1 Background

Compared with detection tasks in other fields, e.g., using face recognition to unlock the phone, traffic sensing always works 24/7 and suffers more harrowing situations, e.g., different weather and light conditions. Sensing in adverse conditions is always challenging but mandatory for traffic sensors because these conditions are regular in daily life. For example, Seattle has an annual number of rainy days of around 150 and shorter daylight hours than those southern regions. Robust traffic sensing technologies in adverse conditions are in high demand in Seattle. For conventional RGB cameras, adverse conditions bring low visible distance and lack texture information. This chapter studies pedestrian detection in low-light

conditions. According to Figure 1.4, images captured in low-light conditions have a lot of noise. Objects that appear in those images are also unclear. The detection accuracy will decrease steeply with the increasing distance between the camera and objects. It will be difficult for computer vision algorithms to detect objects when only using RGB cameras. The introduction of other sensors is necessary.

Pedestrian detection with roadside units and onboard perception systems is crucial for ITS associated with pedestrian safety and mobility. For example, pedestrian detection on the roadside unit can help optimize the intersection signal phase and timing, especially for disabled persons who have no access to the push button. Pedestrian detection for self-driving and ADAS contributes to collision avoidance and motion prediction. In general, detection accuracy and efficiency are two primary metrics evaluating the effectiveness of the systems. Usually, the systems' prediction and control actions are triggered according to real-time pedestrian detection results[370,69,314]. Firstly, sensors pass collected data to processors for analysis. Then, prediction and control modules will accept the analysis results. As the frontend of the whole system, the outputs of the perception system will be inputs of other modules, e.g., system control. False or untimely detection results can negatively impact next-step works. Different from the previous two chapters facing insufficient training data, this chapter at least has sufficient training data to avoid overfitting. Nevertheless, the training and testing data quality is not satisfactory. Chapter 5 has stated that both false and untimely detections are unacceptable in the collision avoidance system. In that chapter, low accuracy comes from the model not learning enough features from training data. This chapter faces the problem rising of similar textures to the background and high data noise overwhelming the target. Low SNR guides the model to learn more invaluable information from noise.

Besides accuracy, detection efficiency matters since most relevant applications are on board and require fast responses. Thus, an accurate and efficient pedestrian detection algorithm is significant for improving pedestrian safety and mobility.

6.1.2 Application Scenario – Multispectral Pedestrian Detection

Nowadays, multiple sensing technologies have been applied for pedestrian detection, including vision-based sensing[89], LiDAR[85], radar-based sensing[271], wireless sensing[119,226,227], etc. Among existing sensing technologies, the vision-based method has been demonstrated to be the most effective tool for pedestrian detection due to the affluent information and cost-effective features[123,26]. On the algorithm side, there are tons of efforts to improve the accuracy and efficiency of pedestrian detection using video or image data[293,5,101]. According to previous studies, one crucial factor limiting the detection performance is the environmental illumination[48,350,224]. Thermal images are suitable to mitigate the negative impacts of low illumination and are utilized as a complementary data source when RGB image quality decreases as illumination status drops[93]. Multispectral algorithms have been developed to fuse thermal and RGB images automatically. However, several specific disadvantages exist in the previous studies that considerably restrict the algorithm's accuracy and efficiency. Most existing studies only consider the illumination factor in computing the fusion weights regarding detection accuracy. Still, the temperature information extracted from thermal images is not utilized, which is intuitively viewed as a significant factor for pedestrian detection under low illumination conditions. Besides, previous studies do not optimize how to compute dynamic fusion weights, which may cause higher detection errors in multiple implementation scenarios. For the algorithm efficiency, even the existing neural network-based algorithms highly

improved the detection accuracy – the accompanying additional computational complexity impaired real-time performance. Since traffic sensing usually requires real-time processing, inference speed is also a critical evaluation metric.

6.1.3 Contributions and Organization

Motivated by the limitations of existing detection algorithms stated in Section 6.1.2, this chapter concentrates on proposing an accurate and efficient multispectral detection algorithm, especially for pedestrians in low-light conditions. Instead of directly concatenating RGB and thermal images as the network input, this chapter dynamically fuses RGB and thermal embeddings based on environment light and temperature information. Considering the potential implementation on edge devices and features of detected targets, the proposed network reduces the default box number and is further optimized with deep learning compression. The primary contributions of this chapter are summarized in the following five points.

a) Advance a one-stage detection network IT-MN for multispectral pedestrian detection in adverse conditions by fusing RGB and thermal images. This network has four highlight features – one-stage base model, dynamic fusion weight computing, a late-fusion strategy, and optimized default box generation. In the beginning, this chapter considers accuracy and efficiency equally important.

b) Design a CNN-based network FWN as a component of IT-MN dedicatedly for computing fusion weights from RGB and thermal images. Pre-defined fusion weights, generally the same value, consider an equal contribution from RGB and thermal images. This assumption simplifies the model structure and may not be real in every case.

Thus, dynamically adjusting the fusion weights makes more sense by online estimating current environmental conditions.

c) Design an effective late-fusion strategy by utilizing all feature maps for feature extraction and fusion on multiple scales to increase accuracy further. Benefit from separable convolution, late-fusion is as efficient as possible. Earlier fusion may reduce the computing complexity but lose more global features as the network goes deeper.

d) Optimize the default box generation by reducing the box number and selecting specific box aspect ratios. Based on empirical observations, the aspect ratio of target pedestrians is always less than one, which implies that default boxes with an aspect ratio greater than 0 will not generate true predictions. This chapter drops these boxes to reduce computing complexity.

e) Apply model quantization to reduce the model size and shorten the inference time, especially for edge computing with weak computing power. Thanks to post-training after quantization, this network is even faster and has a very close accuracy to its original. It is the first time exploring deep learning compression in multispectral object detection to our best knowledge.

The remainder of this chapter is organized as follows. Section 6.2 presents the proposed IT-MN in detail. Firstly, Section 6.2.1 depicts the general framework of IT-MN. The base model is SSD with two input data sources. These two towers do not share weights and have their separate backpropagation flows. The fusion strategy is late fusion, indicating fusion is conducted before classification and localization predictions. To fuse RGB and thermal embeddings with dynamic fusion weights, this chapter designs a CNN-based network for

computing fusion weights in Section 6.2.3. FWN considers the impacts of environmental light and temperature on the fusion process. What's more, Section 6.2.4 introduces one small but effective improvement for inference speed by removing partial default boxes. Inspired by the aspect ratio of pedestrians in the real world, default boxes with an aspect ratio greater than one are likely to contribute to final predictions. Section 6.3 shows the experiment results and discusses the corresponding implications. The evaluation metric is MR, defined in Equation 1.1. This experiment compares the proposed network with other state-of-the-art algorithms in MR and inference speed and finds it exceeds them in accuracy and inference speed. The plots of MR versus FPPI also demonstrate this conclusion. Besides, several ablation tests are conducted to evaluate different modules and strategies, including late fusion, FWN, and optimized default box generation. The experiment results show that all modifications are meaningful in the multispectral pedestrian detection scenario.

6.2 Methodology

6.2.1 Multispectral Detection Model Framework

Multispectral object detection can significantly improve detection accuracy in low light conditions, which will benefit autonomous driving and intersection monitoring. Edge computing is the optimal solution for these applications, considering the processing latency. As discussed in previous chapters, edge devices are constrained by their computing power. The running algorithm should be as efficient as possible. Thus, this chapter designs an efficient and robust multispectral detection algorithm for all-day detection tasks. There are two primary keywords for this algorithm – one-stage and multispectral. This chapter builds the proposed algorithm based on the representative one-stage model SSD with two independent

input branches. Since SSD is an anchor-based model, it requires default box generation to offer dense possible candidate regions. The prediction head will regress these region coordinates and classify them. The overall framework of the proposed multispectral detection model IT-MN is shown in Figure 6.1. The inputs include a pair of well-aligned RGB and thermal images. The outputs are target bounding boxes and labels. In Figure 6.1 and the following figures, the abbreviation *conv* represents the convolutional layer. To be simplified, activation and batch normalization layers are ignored.

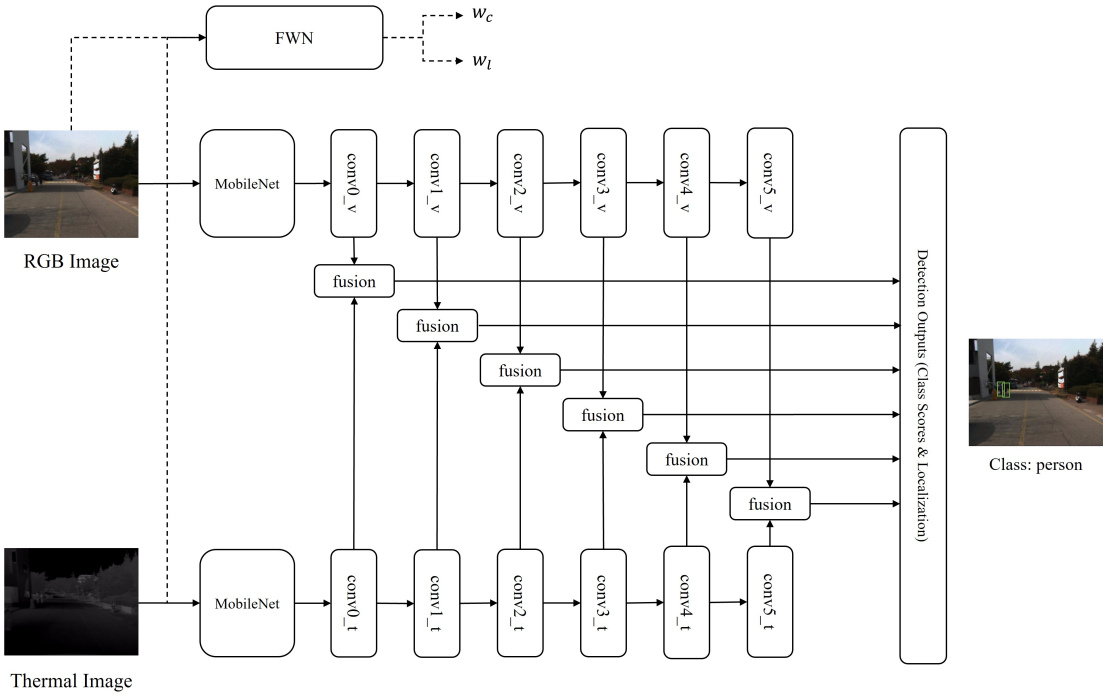


Figure 6.1: IT-MN Framework

The base model of IT-MN is D-SSD. The main difference between D-SSD and regular SSD is that the former has two input branches of RGB and thermal images The backbone of D-SSD refers to MobileNet, instead of VGG-16 and ResNet-50, by removing the last av-

erage pooling and fully connected layers. According to Chapter 3, MobileNet can reduce the parameter and operation number significantly because of the proposal of separable convolution. The computing complexity of MobileNet is only 1/8 of ResNet-50 and 1/39 of VGG-16. In Figure 6.1, The fusion strategy is late fusion, and fusion weights w_c and w_l in Figure 6.1 are computed through the FWN. w_c refers to the weight to fuse the classification score vector, and w_l refers to the weight fusing localization coordinate vector. The fusion net also shows in Figure 6.1, whose function is fusing feature maps from two inputs using pointwise addition. FWN calculates fusion weights based on RGB and thermal images, which will be discussed in Section 6.2.3. FWN is applied to promote the detection accuracy under various illumination and temperature conditions. The fusion net concatenates feature maps extracted from RGB and thermal images on six feature levels. After the fusion, the fused feature maps will get into the classification and regression layers. The detailed structure of the fusion net will be described in Section 6.2.2. According to Figure 6.1, the bounding boxes and classes prediction is conducted on six feature maps that are from *conv*0 to *conv*5 layers. The feature map sizes are 38×38, 19×19, 10×10, 5×5, 3×3, and 1×1 respectively. Each cell on selected feature maps will generate four default boxes as anchors. The coming contents will introduce feature fusion, default box generation, and FWN in order.

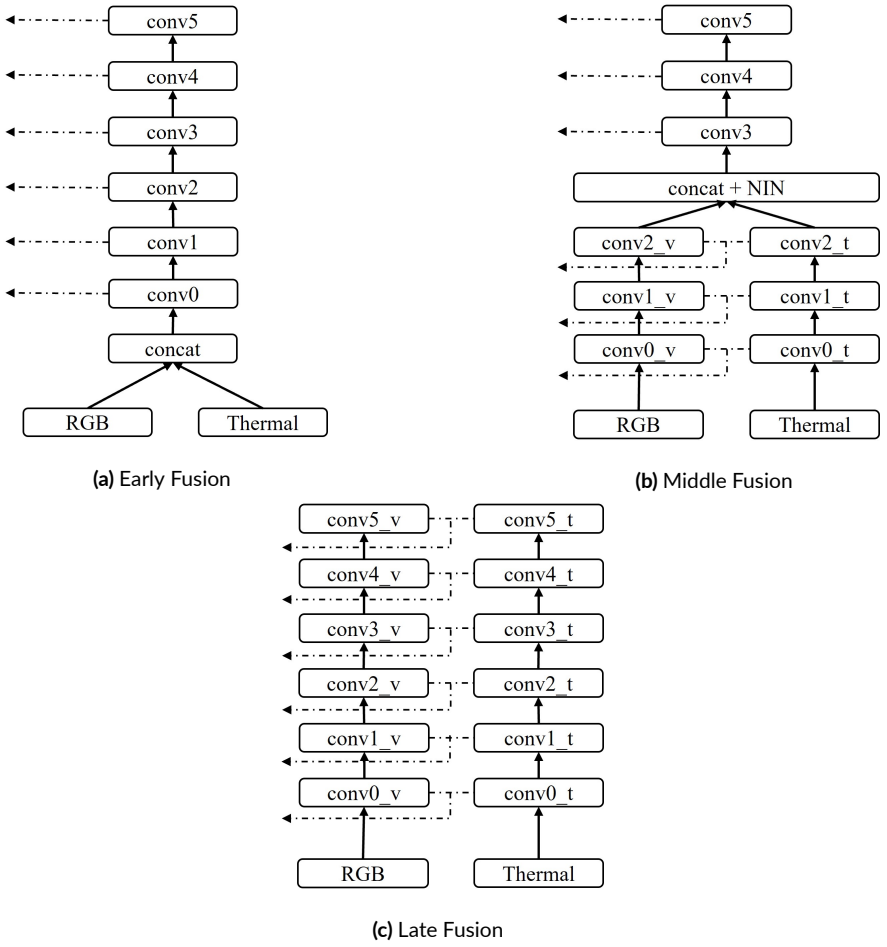
6.2.2 Feature Fusion Design

Based on previous research, the late fusion has higher accuracy than the early fusion[294,117]. Unlike the previous work, this chapter adopts a one-stage framework D-SSD using a two-stage detection network that first generates region proposals and then regresses and classifies each proposed region. Thus, the late fusion may not be the optimal strategy in the one-

stage detection model. Therefore, it is necessary to evaluate different fusion strategies based on the D-SSD skeleton. This chapter tries to find the appropriate fusion strategy and compares three kinds of fusion strategies – early fusion, middle fusion, and late fusion. These three frameworks are shown in Figure 6.2. The early fusion, shown in Figure 6.2a, concatenates original RGB and thermal images as the input for the detection network. The middle fusion, shown in Figure 6.2b, concatenate feature maps from two streams after passing through *conv*3*v*/*conv*3*t*. The concatenation process applies Network in Network (NIN)[179], referenced from Daniel's work[158]. The late fusion, in Figure 6.2c, does not concatenate feature maps directly during the feature extraction process. Instead, the fusion is conducted in the process of target prediction. This chapter only plots convolutional layers from *conv*0 to *conv*5 in Figure 6.2 to highlight the differences among these three strategies while omitting other layers, e.g., ReLU and Batch Normalization.

Different from early fusion in Figure 6.2a, other fusion strategies, i.e., middle and late fusion, will fuse latent features after passing through several convolutional layers. The difference between middle and late fusion is that middle fusion conducts the feature extraction separately for two inputs in low feature levels before fusion. Late fusion performs separate feature extractions for two inputs in all feature levels. From *conv*0 to *conv*3 in middle fusion and *conv*0 to *conv*5 in late fusion, the fusion process on a pair of feature maps at the same level keeps the same. In Figure 6.3, x_v and x_t refer to feature maps from RGB and thermal inputs respectively. y_0 and y_1 are outputs of feature fusion, which are also the input of prediction heads. *Cls* refers to the object class number, and D_n represents the default box number.

The fusion weights w_c and w_l computed by FWN, described in Section 6.2.3, are applied to calculate the weighted sum of class scores and bounding box regression. w_c is the weight to



(a) Early Fusion

(b) Middle Fusion

(c) Late Fusion

Figure 6.2: Frameworks of Fusion Strategies

fuse feature maps used for classification, and w_l is the weight to merge feature maps used for bounding box regression. The fusion operation is a pointwise addition. A 1×1 convolution operation follows the weighted sum for better fusion performance. Add an extra 1×1 convolution operation is a general operation in deep learning after merging or concatenating feature maps. As a result, the final fused results are computed through $y_0 = f_0(w_l \cdot x_v + (1 - w_l) \cdot x_t)$ and $y_1 = f_1(w_c \cdot x_v + (1 - w_c) \cdot x_t)$, where f_0 and f_1 represent operations of $conv0$ and $conv1$ respectively. "Loc Prediction" is responsible for predicting the coordinates of bounding boxes.

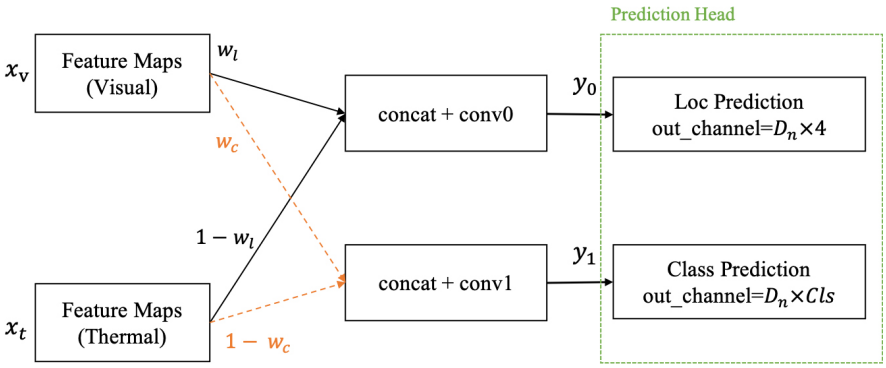


Figure 6.3: Framework of Localization and Classification Fusion

The values are relative coordinates to anchors instead of absolute coordinates. "Class Prediction" is responsible for predicting classification scores for each bounding box. The classification prediction output will pass through a softmax operation to ensure all values are between zero and one.

6.2.3 Fusion Weight-Computation Network

Previous works have considered and studied the impacts of illumination on detection performance and thus designed IA network[174,100]. In most scenarios, computer vision models receiving RGB images perform better in a bright environment, while models receiving thermal images work better in a dark background. Thus, researchers have designed the IA network to estimate the illumination impacts using RGB images. In addition to illumination impacts, the temperature will also affect the detection performance because the infrared cameras are more sensitive to object radiation and temperature differences. When the environment temperature is high, it becomes hard for infrared cameras to distinguish the background and pedestrians based on texture because heat radiation from the background and target is similar. The RGB input should occupy a more significant weight in the fusion process in this

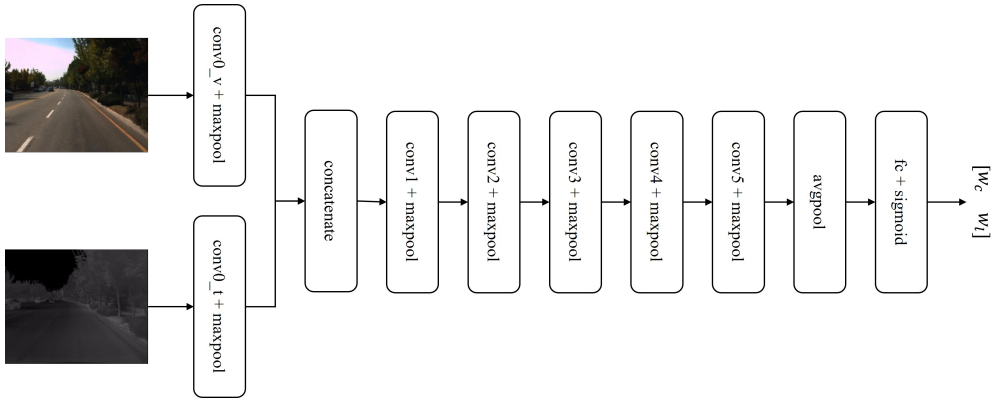


Figure 6.4: FWN Framework

situation. However, the IA network cannot analyze temperature impacts due to the input type. It is necessary to introduce temperature awareness into the network. Therefore, this chapter proposes the FWN to compute the fusion weights by considering the impacts of both factors. According to Figure 6.1, FWN is a separate network, and its framework of FWN is shown in Figure 6.4.

FWN is a simple CNN with only six convolutional layers. In Figure 6.4, concatenate operation is working on channel dimension. *fc* indicates the fully connected layer. Following the same plotting rule in Figure 6.1, this figure also omits all ReLU and batch normalization layers. The parameters of convolutional and fully connected layers in FWN, including input channel number, output channel number, and kernel size, are shown in Table 6.1. The pooling operation is max pooling with a downsampling factor of 2. A sigmoid function is added to the end of the *fc* layer. The inputs of FWN are RGB and thermal images. The output of FWN is synthesized illumination-temperature parameters w_c and w_l, which are fusion weights for classification and localization, respectively. Considering FWN is also a trainable network, this module does not need separate training, which indicates that the whole model

Table 6.1: Network Parameters of FWN

Layers	In Channel	Out Channel	Kernel Size
conv0	3	16	3
conv1	32	64	3
conv2	64	128	3
conv3	128	256	3
conv4	256	128	3
conv5	128	64	3
fc	64	2	-

can be trained end-to-end.

6.2.4 Default Box Generation

IT-MN is specifically designed for efficient multispectral pedestrian detection on edge devices. This chapter reduces the number of default boxes to shorten the inference time because the default box number is directly related to the computing complexity of the last prediction layers. Another method is reducing parameter precision using model quantization following the process described in Section 3.2.2. However, reducing the default box number will decrease the detection accuracy since proposing ROI relies on dense candidates. More default boxes indicate a higher probability of covering target objects. Each grid on the feature map will generate multiple default boxes with different aspect ratios and sizes. A default box setting for IT-MN is shown in Figure 6.5, where L and W represent one default box's length and width respectively. The box aspect ratio a_r is defined as W/L, including four different values of 1, 2, 3, 1/2, 1/3. On a 4×4 example feature map as Figure 6.5, the aspect ratios applied are 1, 2, and 1/2 and the total number is 4. The different box colors represent different aspect ratios.

This chapter mainly focuses on pedestrian detection using onboard cameras. Based on

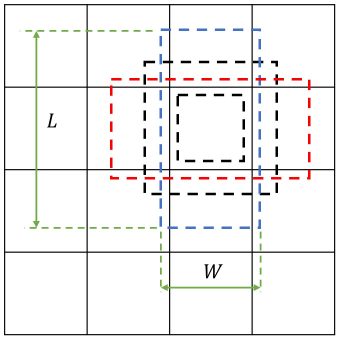
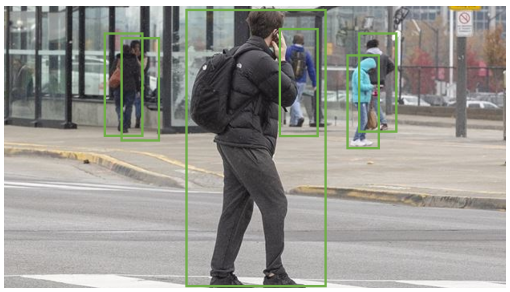
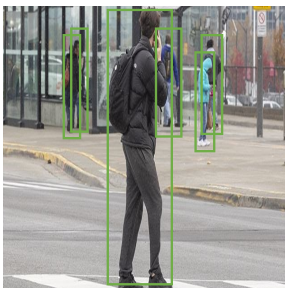


Figure 6.5: Default Boxes for IT-MN

statistics of pedestrian detection datasets, the mean aspect ratio of a person is less than one. In addition, the aspect ratio of the original video frame is usually 16:9, while the input dimension of SSD requires a square size. After resizing the input shape to a square, the aspect ratio of a person in the image will further decrease and becomes close to or less than one. Figure 6.6 shows two sample images of the same scenario before and after the resizing, where Figure 6.6a shows the image in the original size and Figure 6.6b shows the resized image.



(a) Sample of Original Dimension



(b) Sample of Resized Dimension

Figure 6.6: Sample Pedestrian Images

Thus, those default boxes, whose aspect ratio is greater than one, will not significantly contribute to final detection results because they will not be selected to predict targets. This chapter does not consider those relatively useless boxes and eases the computing work in pre-

diction. Based on this fact, the aspect ratios in IT-MN are 1, 1/2, 1/3. After setting aspect ratios, the actual size of bounding boxes follows the original SSD. Suppose there are m feature maps used for prediction, the box scale s_k for k^{th} feature map is computed using Equation 6.1, where s_{max} and s_{min} are set as 0.9 and 0.2 respectively. Thus, the length L_k and width W_k of default boxes on the k^{th} feature map are defined $s_k\sqrt{a_r}$ and $s_k/\sqrt{a_r}$ respectively. Since the number of default boxes for each grid reduces to 4, the final prediction will only be conducted on the remaining 66% candidate boxes, and the corresponding computing complexity reduces by 33%.

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1) \tag{6.1}$$

6.3 Experiments

6.3.1 Experiment Setting

Experiments are conducted on the KAIST dataset[126], published in 2015. The training and testing images came from a pair of onboard RGB and thermal cameras. Compared to the datasets collected by the camera at a fixed location, this dataset is more challenging since the background and environment illumination change frequently. As a result, some effective methods, such as background subtraction, are not suitable in this scenario. The images in the KAIST dataset were captured at 20Hz with a resolution of 640×512. Dataset creators have completed image registration and alignment of RGB and thermal cameras. Thus, each pair of RGB and thermal images have no perspective difference. All images contain over 100,000 dense annotations in total. There are six scenarios collected in this dataset: campus, road, and downtown during daytime and nighttime. Each scenario has both training and testing

sub-datasets, but the sample number of each scenario is insufficient. This chapter mixes all scenarios for model training and testing. Considering that images are collected continuously using an onboard camera system, the adjacent frames are similar. Furthermore, partial images do not contain objects or have uncleared labels such as "person?". As suggested in previous work[173,351], the data sampling and cleaning are required to filter out most images not containing any target object, leading to unbalanced data distribution and increasing prediction bias. According to Chapter 4, data augmentation is necessary and applied to train the model more efficiently, including random-sized cropping, horizontal flipping, and saturation adjustment. All input images are resized to 300×300.

In this chapter, experiments were conducted on a PC equipped with the GPU of Nvidia Titan XP. A Raspberry Pi 4 is used to evaluate model efficiency on the edge side. The IT-MN's backbone MobileNet v2 is pre-trained on ImageNet. The weights of additional layers are randomly initialized with a uniform distribution. The batch size in the training stage is 8. Due to the limited GPU memory, the model weights will update every four batches using gradient accumulation with a step length of 4. As a result, the equivalent batch size becomes 32. In the testing stage, the batch size becomes 1 to mimic the image sequence in practice. The optimization method is Adam instead of stochastic gradient descent to speed up convergence. The initial learning rate is 0.001 with a learning rate adjustment of a cosine annealing schedule. The total training epoch number is 200. The focal loss is applied to improve the training process, as the improvement of general cross-entropy loss. Its definition is shown in Equation 5.3. The log-average MR is adopted as the accuracy evaluation metric to evaluate the detection results. According to Section 1.2.3, MR reflects the missing object number among all positive objects, which is more meaningful than the average precision in the trans-

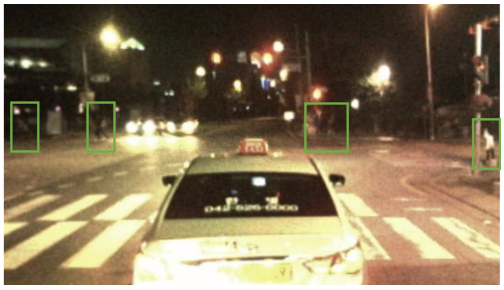
portation field since the missing detection may lead to serious transportation safety issues. A predicted bounding box is a true positive when it matches a ground truth box with IoU greater than 0.5. Since multiple bounding boxes will match a single ground truth, only the box with the highest IoU will be the true positive. Unmatched predicting and ground truth boxes are false positives and false negatives. The log-average MR is computed by averaging MR at nine False Positives Per Image (FPPI) rates evenly spaced in a log space from 0.01 to 1 [65]. In addition, the inference speed is also a critical evaluation metric. Efficiency is vital when applying these algorithms on the edge side.

Some positive samples of multispectral pedestrian detection using IT-MN are shown in Figure 6.7, where green bounding boxes mark the detected pedestrians. The sample scenarios include both day and night in road scenarios. Figure 6.7a and Figure 6.7b show the detection results using RGB and thermal images at day road scenario respectively. The detection results at night road scenario are presented in Figure 6.7c and Figure 6.7d. Comparably, pedestrians in RGB images have more texture information during the daytime. During the nighttime, thermal images can recognize pedestrians more easily.

What's more, some samples of false detection are shown in Figure 6.8 and Figure 6.9. Figure 6.8a and Figure 6.8b show false negative detections in campus scenario at both day and nighttime, which are marked with yellow bounding boxes. It can be observed that the false negative objects are almost at a small scale, where the texture information is scarce on all feature maps. One primary reason is the low input resolution. The downsampling factor of the first layer for prediction is 16, which may lead to poor detection for small objects. The pedestrian in Figure 6.9 projected to that feature map occupies 3 pixels. Figure 6.9a and Figure 6.9b show false positive detections in campus scenario at both day and nighttime, which

(a) Day Road (RGB)

(b) Day Road (Thermal)

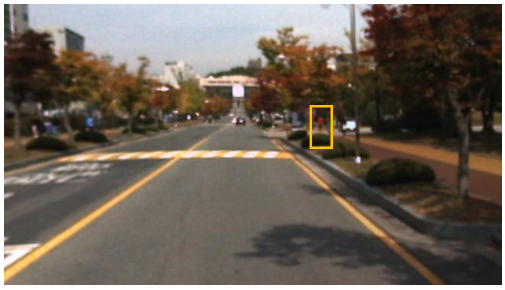


(c) Night Road (RGB)

(d) Night Road (Thermal)

Figure 6.7: True Positive Samples in Road Scenario

are marked with red bounding boxes. Objects with a similar texture to pedestrians may be recognized as pedestrians in some cases.

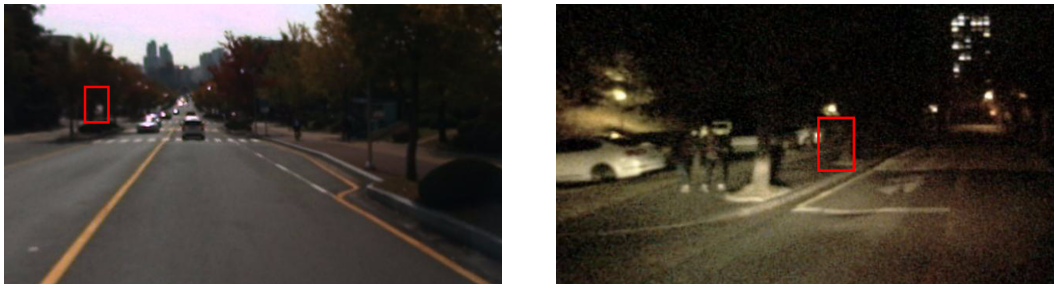


(a) Day Campus

(b) Night Campus

Figure 6.8: False Negative Samples in Road Scenario

The experiments are divided into the following sections. Section 6.3.2 compares the accuracy of the proposed algorithm with the state-of-the-art models in terms of both MR and



(a) Day Campus **(b)** Night Campus

Figure 6.9: False Positive Samples in Campus Scenario

inference time. The log-log plots of MR against FPPI present the comparison results. To show the improvement by fusing the RGB and thermal images with FWN, Section 6.3.3 and Section 6.3.4 present the ablation tests using different inputs, fusion strategies, and awareness networks. Section 6.3.5 shows the improvements by optimizing the default box generation by comparing the optimized and original settings. Compared to previous research, this chapter also concentrates on the performance of the edge device. This chapter conducts all speed-related tests on PC and Raspberry Pi 4 to evaluate its efficiency on the server and edge platforms.

6.3.2 Evaluating Detection Accuracy of the Proposed IT-MN

In this section, the proposed algorithm is evaluated based on comparing state-of-the-art multispectral pedestrian detection methods in terms of MR and inference time. The comparison models are Late-Fusion SSD (L-SSD)[117], CWF-CNN[224], IATDNN[100] and MLF-CNN[49], CS-RCNN[351]. Six scenarios in KAIST dataset are merged into three groups based on the collection time – day, night, and all time. The comparison results of MR are shown in Table 6.2. Table 6.2 presents the mean MR in terms of all-day, daytime, and nighttime. Figure 6.10

to Figure 6.12 plot MR versus FPPI.

In Table 6.2, the mean MR is computed. Comparing two SSD-based algorithms (L-SSD and IT-MN), the proposed model reduces MR significantly from 43% to 16%, which proves the necessity of applying the extra network to compute the fusion weights instead of fusing feature maps with equal contribution. IT-MN has a much lower MR than other state-of-the-art algorithms including CWF-CNN, IATDNN, and MLF-CNN. The mean MR of IT-MN is around 14%, and other algorithms have at least 10% higher than the proposed IT-MN. Meanwhile, the quantized IT-MN, which is designed for the edge device, is also evaluated. The only difference between the quantized and original IT-MN is the parameter precision from 32-bit floating point to 8-bit integer precision. The quantized IT-MN is called "Q. IT-MN" in the following part. Contributed by the post-training process in Chapter 3, the MR of Q. IT-MN is only around 0.3% higher than IT-MN. In Figure 6.10 and following two figures, the x-axis is the FPPI and the y-axis is MR. The range of FPPI is between 0.001 and 1.0 using a log scale and the mean value of MR is computed in the range between 0.01 and 1.0. According to the plots, the curve of IT-MN is lower than the curves of all other algorithms, which indicates that IT-MN has higher accuracy when using different thresholds of detection confidence under all light conditions.

Table 6.2: Comparison of Mean MR in terms of All Day, Daytime and Nighttime

Time	L-SSD	CWF-CNN	IATDNN	MLF-CNN	IT-MN	Q. IT-MN
All Day	43.06%	31.36%	29.62%	25.65%	**14.19%**	14.55%
Day	50.73%	31.79%	30.30%	25.22%	**14.30%**	14.67%
Night	35.38%	30.82%	26.88%	26.60%	**13.98%**	14.29%

Besides the detection accuracy, the inference time is another critical evaluation metric. This chapter compares the inference time of the proposed algorithm and other state-of-the-

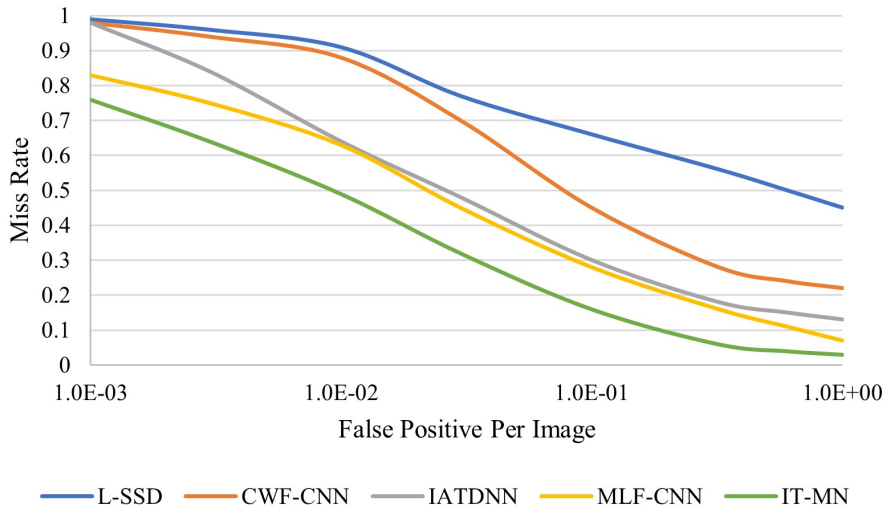


Figure 6.10: Comparison of Detection Accuracy (MR versus FPPI) in All Day

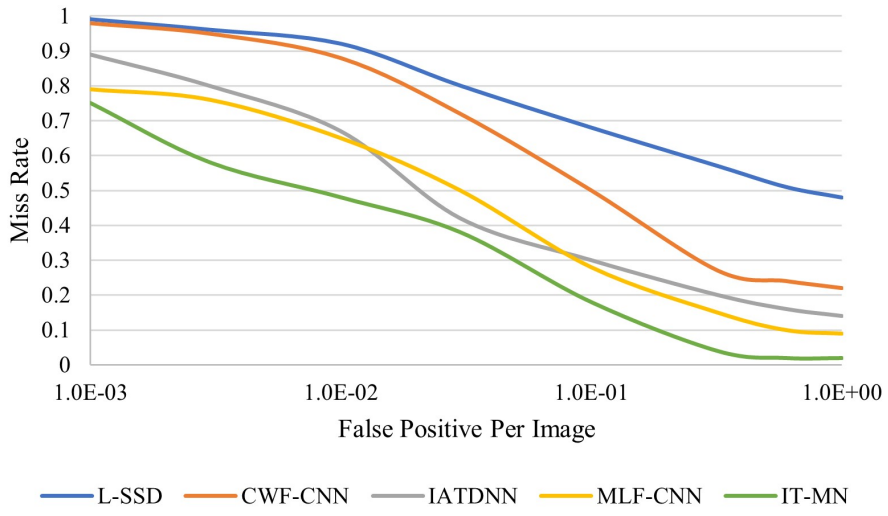


Figure 6.11: Comparison of Detection Accuracy (MR versus FPPI) in Daytime

art algorithms. To demonstrate the algorithm's potential on an edge device, this chapter conducts separate evaluations on PC with GPU and Raspberry Pi 4, popular open-source hardware for edge computing. According to the comparison results presented in Table 6.3, it

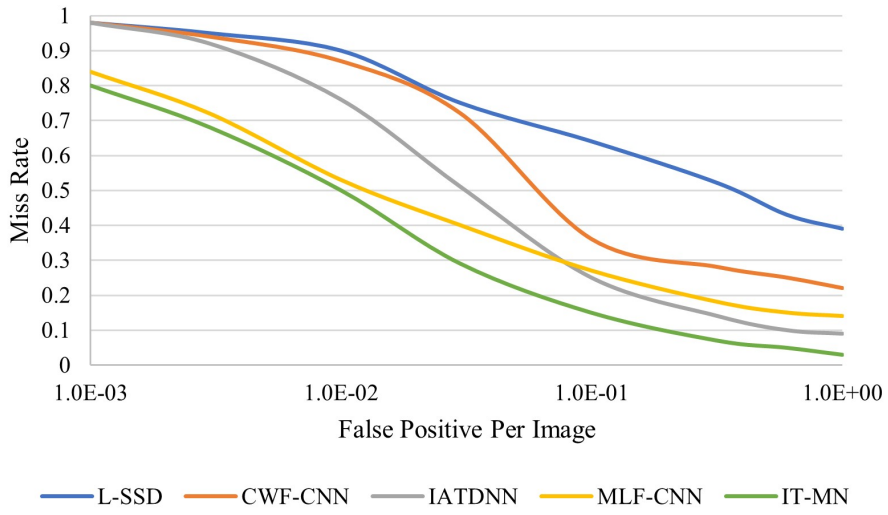


Figure 6.12: Comparison of Detection Accuracy (MR versus FPPI) in Nighttime

can be found that the inference time of IT-MN is impressive and at 33 FPS when using GPU. However, it is notable that L-SDD achieves the same inference time as the proposed IT-MN, whose accuracy is much worse than IT-MN. Compared to other algorithms achieving similar MR, IT-MN is much faster than others. To be more specific, the proposed IT-MN is five times faster than MLF-CNN when using GPU. Compared to IATDNN, the efficiency difference becomes incredibly eight times. When implementing these models on Raspberry Pi 4, their comparison results are similar to the GPU situation. The proposed IT-MN outperforms almost all other algorithms except for L-SDD. However, L-SSD has a much higher MR than IT-MN. In addition, deep compression on the proposed model is one highlight for this chapter. After quantizing the proposed IT-MN, the inference time increased significantly, about two times faster than the non-quantized IT-MN. Furthermore, Q. IT-MN is around seven-time faster than MLF-CNN and over ten times faster than IATDNN. The effectiveness of deep learning compression is demonstrated again as stated in Chapter 3. In conclu-

sion, the proposed IT-MN reaches 0.03 and 0.21 seconds per frame, respectively, when using GPU and the Raspberry Pi 4. At the same time, other state-of-the-art algorithms take several seconds to process one image pair. According to the inference time, experiments results in Table 6.3 implies IT-MN can process dozens of image pairs using GPU. Even if running on the Raspberry Pi 4, whose computing power is way more limited than GPU, IT-MN also can process about five image pairs per second. The results indicate that the proposed IT-MN inference time is short enough to achieve excellent real-time performance for detecting pedestrians through a live video.

Table 6.3: Comparison of Inference Time on GPU and Raspberry Pi (Unit: FPS)

Devices	L-SSD	CWF-CNN	IATDNN	MLF-CNN	IT-MN	Q. IT-MN
GPU	33	2	4	7	**33**	-
Edge	2.6	0.2	0.4	0.6	2.5	**5.0**

6.3.3 Evaluating Fusion Strategies

Conventional algorithms always rely on RGB images and suffer from poor data quality in low-light conditions. Thus, the multispectral fusion of RGB and thermal images can compensate for this drawback. The proposed IT-MN relies on both RGB and thermal inputs. It is essential to conduct ablation tests to evaluate the performance when applying multispectral data. Hence, this part will explore the improvements using different data sources and fusion strategies. The first experiment focuses on assessing the improvement after applying the thermal images. Then, the second one compares the accuracy when adopting varying fusion strategies.

Thermal images can provide more pedestrian feature information from persons' radiation when light conditions are poor without sufficient visible-light information. The machine sys-

tem can easily recognize pedestrians using thermal images in this situation. However, thermal images are sensitive to both environment and pedestrian temperature. When the environment temperature is relatively, or pedestrians are distant from the camera, it becomes hard to recognize only using thermal images. Compared with the night scenario, the temperature in the daytime is higher, which indicates that other objects besides pedestrians are more noticeable in thermal images. It thus increases the difficulty of distinguishing pedestrians from the background. Therefore, the thermal image still has limitations and can improve detection results under specific scenarios.

The first experiment evaluates how different input sources affect performance. Table 6.4 shows MR comparison results among three networks – SSD only using RGB images, SSD only using thermal images, and IT-MN fusing RGB and thermal images. According to Table 6.4, SSD using RGB images has lower MR during daytime and higher MR during nighttime. Considering that the pixel value of thermal images represents the heat received by the sensor, a higher pixel value means higher temperature. Thus, comparing the average pixel value of images from day and night can obtain the mean heat or temperature at different times. The average pixel value of day thermal images is 73% higher than night images, indicating that the daytime temperature is higher than at night. After synthesizing this information, thermal images are more suitable for pedestrian detection in the condition of low temperature and illumination. The RGB image can provide more detailed and reliable information when the illumination is bright enough. The thermal image can improve MR performance at night.

Table 6.4 shows multispectral fusion is more effective than single data source because multispectral fusion utilize more information. However, it may negatively impact MR during daytime if thermal images are directly concatenated with RGB images as the input, equal to

Table 6.4: Comparison of Different Data Sources

Data Sources	MR (All Day)	MR (Day)	MR (Night)
RGB	41.27%	36.79%	46.22%
Thermal	40.73%	47.77%	32.29%
Fusion	**14.19%**	**14.30%**	**13.98%**

Table 6.5: Comparison of Different Fusion Strategies

Fusion Strategies	MR (All Day)	MR (Day)	MR (Night)
Early Fusion	29.51%	29.82%	28.60%
Middle Fusion	24.77%	26.60%	22.33%
Late Fusion	**14.19%**	**14.30%**	**13.98%**

applying a fixed fusion weight. Therefore, the next experiment compares different fusion strategies described in Section 6.2.2, including early fusion, middle fusion, and late fusion. Table 6.5 shows the experiment results at different time in a day. The early fusion has the highest MR, which indicates that directly stacking RGB and thermal images as one input branch could not fully use thermal information. The late fusion strategy has much lower MR than middle fusion and only has 14.19% MR on average, almost half of middle fusion. It can be found that early fusion and middle fusion do not have much difference in accuracy partially because they fuse two branches in shallow feature levels. Fusion at deeper feature levels is more likely to generate better results. When comparing Table 6.4 and Table 6.5, even early and middle fusion can exceed the model using single data source, which further demonstrate the effectiveness of multispectral fusion.

6.3.4 Evaluating Awareness Network

One main contribution of this chapter is proposing a novel FWN considering both illumination and temperature factors. This chapter compares four kinds of awareness networks

– Network without Awareness Network (N-MN), Network with Illumination-Aware Network (I-MN), Network with Temperature-Aware Network (T-MN), and the proposed IT-MN. The fusion weight in N-MN is 0.5, which indicates that the final detection results are calculated by averaging the outputs of RGB and thermal image data streams. The comparison results of the four strategies show in Table 6.6. According to the results, both IA and Temperature-Aware (TA) can effectively reduce MR. However, the decreasing scale of these networks is different under different conditions. I-MN has a lower MR in the daytime, while T-MN has a lower MR at nighttime. In general, the IA network has a lower MR on all scenarios than TA network, which indicates that the illumination factor is more apparent than the temperature factor. After integrating IA and TA, IT-MN reduces MR by 20% compared to N-MN.

Table 6.6: Comparison of Different Awareness Networks

Algorithms	MR (All Day)	MR (Day)	MR (Night)
N-MN	35.28%	35.66%	34.30%
I-MN	27.93%	27.45%	28.70%
T-MN	29.65%	30.80%	27.61%
IT-MN	**14.19%**	**14.30%**	**13.98%**

To further demonstrate the contribution TA network to the detection accuracy, a set of images are selected to test the performance of the algorithms in an environment with low illumination and high temperature. The image pairs are collected, which meet the requirement that the illumination is lower than the threshold T_{ill} and the temperature is higher than the threshold T_{tem}. The illumination measurement is the sum of lightness channel pixel value after transforming the RGB image into LAB color space. The temperature is measured by computing the sum of pixel values from the thermal image. Two thresholds are set based on

Table 6.7: Comparison of Different Awareness Networks under Specific Temperature

Algorithms	N-MN	I-MN	T-MN	IT-MN
MR (Night)	39.18%	33.37%	31.09%	**20.29%**

the average illuminance and temperature of all training and testing datasets. The experiment results are shown in Table 6.7. According to the results, the network with TA performs better in the selected image pairs than in the full KAIST dataset, where the MR drops by 5.63% by integrating TA with N-MN in the complete KAIST data. In contrast, the MR drops by 8.09% in the selected image pairs. Furthermore, IT-MN has an outstanding performance in both sets, whose MR is at least 10% lower than other networks. Thus, this comparison result verifies that the temperature factor can contribute more when the environment temperature is high. As a result, FWN is more beneficial for pedestrian detection in various environments than single awareness networks, e.g., IA and TA networks.

6.3.5 Evaluating Performance of Default Box Optimization

Besides proposing FWN to compute the fusion weights, another framework improvement is optimizing the default box generation by reducing the number of boxes and modifying their aspect ratios. This improvement focuses on the model efficiency for edge computing applications. According to Section 6.2.4, the total number of default box decreases by 33% from 8,732 to 5,820 compared to original SSD. Only boxes with an aspect ratio of less than one remain for prediction. In general, denser default boxes indicate better coverage and higher detection accuracy. However, more boxes also imply higher computing complexity and lower inference speed. This chapter considers the specific detection target and pedestrian aspect ratio. The simplification in default box generation is expected to have almost no negative

impact on detection accuracy while reducing inference time. The comparison results in Table 6.8 match this expectation. Table 6.8 has two blocks. The upper block compares MR in different light conditions. The lower block compares inference speed on the server and edge platforms. In general, this optimization operation achieves a trade-off between accuracy and speed. It can keep the accuracy at a high level while reducing inference time by over 25%.

Table 6.8: Comparison of Different Default Box Settings

Time	Original	Improved	Q. Improved
MR (All Day)	**14.01%**	14.19%	14.55%
MR (Day)	**14.10%**	14.30%	14.67%
MR (Night)	**13.87%**	13.98%	14.29%
Devices			
Speed (GPU)	25 FPS	**33 FPS**	-
Speed (Edge)	1.7 FPS	2.5 FPS	**5.0 FPS**

This experiment is conducted in three scenarios: all-day, nighttime, and daytime. In the upper block of Table 6.8, experiment results show that the network with the original default box has a lower MR since more boxes can increase the probability of covering objects to be detected. However, the original and improved settings have little MR difference, and the gap is around 0.2%. Comparing the third row with others, the quantized model has a slight MR increase of less than 0.5%. The practical applications may even ignore this accuracy difference. There is a more significant gap between these two settings when comparing the inference speed. What is more, IT-MN with an optimized setting is 25% faster than the original one and reaches 33 FPS on 33 FPS on the GPU device and 2.5 FPS on Raspberry Pi 4. The quantized one can be as fast as 5.0 FPS on the same edge device. Considering that the computing power on the edge side is always limited, every effort to promote inference time is meaningful. In conclusion, optimizing the default box generation improves the efficiency significantly while

having slightly negative impacts on accuracy. The compression and post-training strategies proposed in Chapter 3 have demonstrated to be effective.

6.4 Chapter Conclusion

Vision-based traffic sensing is required to work in different weather and light conditions, including a dim environment. Thus, accurate and robust algorithms in such situations are significant to traffic safety and efficiency. Conventional RGB cameras suffer from high noise when the environment illumination is low. This chapter focuses on improving the system performance when the data quality is poor in low-light conditions. Thus, an efficient and accurate multispectral pedestrian detection algorithm, IT-MN, is proposed by considering illumination and temperature factors. The base model of this algorithm is SSD with a late-fusion strategy and additional FWN to compute the fusion weights. The default box generation reduces the proposed number to shorten the inference time. This chapter also discusses applications of model compression and post-training targeting for edge computing applications. The optimization process follows steps in Chapter 3.

The experiment results show that the proposed IT-MN outperforms most state-of-the-art models in accuracy and achieves a low MR at 14.19%. More importantly, the proposed IT-MN can process an image pair every 0.03 seconds using GPU and 0.4 seconds on an edge device without engineering optimization. Furthermore, its quantized version can reduce this time by almost 50%. In conclusion, IT-MN has achieved excellent detection accuracy and efficiency performance. Besides, the proposed algorithm has a great potential for real-time pedestrian detection on edge devices. Future work will explore the domain adaption, which can help bridge the gap between different domains[176]. Different weather will cause the do-

main shift. Considering that it is impossible to collect and label training data under adverse weather conditions, e.g., foggy and rainy days, domain adaptive object detection can be an excellent option to improve the detection performance.

7

Final Remarks

7.1 Summary and Contributions

Urbanization has become a global trend and brought great convenience to citizens by offering better overall facilities, more education opportunities, and better public transport. From the transportation perspective, urbanization stimulates the upgrading and construction of transportation facilities, including roadways and transit centers. Meanwhile, urbanization

indirectly increases the daily travel distance because most office workers live far from their workplaces to avoid high residence costs. As a result, growing travel demand for private and public transportation tools leads to high traffic volume and severe congestion, especially in busy metropolitan regions. Nowadays, traffic congestion is not limited to peak hours but almost the whole day in some cities. According to statistic data in Chapter 1, traffic congestion has wasted a surprising amount of cost and time for commuters and cities. Besides roadway congestion, urbanization has created more crowd gathering scenarios, e.g., transit centers and public activities. It is necessary to monitor crowd safety effectively and avoid crowd disasters. Conventional transportation systems heavily rely on manual operation, leading to slow response to emergent events and suboptimal management plans. Benefitting from advanced AI and IoT technologies, ITS has gradually replaced the conventional system and has become an inseparable part of modern cities. Generally, ITS has four hierarchical components – urban sensing, data analysis, data management, and service providing. This dissertation focuses on urban sensing and data analysis as the knowledge basis of ITS. Various sensing technologies have been applied for accurate and efficient traffic sensing. Compared with other sensing technologies, vision-based traffic sensing has attracted the most attention and achieved promising success in multiple applications, e.g., traffic flow counting and collision avoidance. The advancement of computer vision and machine learning algorithms has further boosted vision-based traffic sensing. As more sensors are deployed for different tasks in ITS, machine learning algorithms have suffered new challenges. This dissertation works on three representative challenges – high data volume for analysis, insufficient training data, and poor data quality in adverse conditions. Therefore, the main research objective of this dissertation is to develop efficient and robust machine learning methods for challenging traffic

video sensing applications.

Chapter 3 concentrates on improving algorithm efficiency and distributing computing tasks to edge devices to cope with the increased data volume. Urbanization increases the monitoring scope of traffic sensors and brings a massive amount of data for analysis. Existing traffic sensing and data analysis frameworks adopt cloud computing structures and are constrained by the central computing power and data transmission bandwidth. Thus, edge computing and super-efficient algorithms are necessary to improve system latency, scalability, and privacy. This chapter firstly designs an optimization pipeline to compress deep learning models. The pipeline has two primary steps – model compression and post-training. This chapter uses model quantization to reduce the model size and computing complexity. In the post-training stage, this chapter applies knowledge distillation to compensate for the accuracy drop in the compression stage. This optimization pipeline is also a fundamental tool in the remaining dissertation works. This chapter studies automated parking monitoring as an application by deploying deep learning classification algorithms on an edge device to monitor real-time parking availability. A real-world dataset is collected and contains continuous video clips instead of image sequences. What is more, this chapter innovatively implements a simulation to demonstrate that low algorithm efficiency will lead to high duration calculation errors in a large-scale parking garage.

Although there is massive traffic data for analysis, the machine learning training data is contrastively lacking. Supervised learning is the primary method for model training, requiring data annotation as supervision. In practical applications, some categories always have fewer training samples than others. Training models on imbalanced and insufficient datasets may lead to high prediction bias and overfitting problems. The following two chapters, Chapter

4 and Chapter 5, pay attention to the challenge of insufficient annotations in training data. Though self-supervised learning can utilize unlabeled data, it requests much more computing power than supervised learning and carefully designed training strategies. These drawbacks make it not suitable for transportation applications. This dissertation still works on using existing labeled data for model training. **Chapter 4** proposes a simple but effective data augmentation strategy, Zoom-Stitcher, to increase data variance, especially for tiny object detection. This method randomly resizes and composes images from the existing dataset to assemble new input images. This chapter applies this data augmentation strategy in the crowd monitoring task to improve the accuracy of crowd counting, group clustering, and motion detection. In addition, this chapter also designs a multi-task crowd monitoring framework for dense and sparse crowd conditions.

Traffic sensing may face some novel or rare objects with a super low appearance frequency in the training dataset. Though data labeling may fundamentally solve this problem, it requires extra time and budget, which may not work for short-term projects. Different from the previous chapter that applies data augmentation to increase the sample number, **Chapter 5** tries to improve the detection accuracy of rare or novel traffic objects using few-shot samples. Since few-shot object detection suffers the situation that some categories only have minimal samples, the variance increased by data augmentation is still insufficient. In addition, small-scale and large-scale objects in Chapter 4 have similar textures, and data augmentation can rescale objects to increase datasets. In ITS applications, there are always newly appearing or rare objects. In the self-driving system, failing to detect these objects may cause severe traffic crashes. This chapter applies a two-step few-shot training strategy and designs a one-stage detection model. One contribution is introducing the few-shot object detection to

ITS to improve traffic safety. The few-shot training strategy implements the cosine similarity to learn features from limited samples and utilize prior knowledge from base categories. This chapter also explores how intra-class variance affects detection accuracy, providing guidelines for data annotation works.

Besides high data volume for analysis and insufficient training data, traffic sensing applications' data quality in adverse conditions is also a concern. Traffic sensing must operate under different weather and light conditions, e.g., the nighttime or in the tunnel, where traffic crashes are more likely to occur. Detecting objects in such conditions becomes significant for traffic safety. Conventional traffic video sensing algorithms using RGB images face problems of lacking texture information and much noise. Sensor fusion is a quick and effective solution to this challenge by combining RGB cameras and other sensors. **Chapter 6** adopts the infrared camera as the assistant sensor instead of LiDAR because of mature 2D computer vision algorithms and high hardware cost for the LiDAR system. This chapter thus proposes a multispectral object detection network for onboard pedestrian detection. This network introduces the dynamic fusion weight to adapt to different light and temperature conditions. Additionally, model efficiency is considered at the beginning of this model design because it targets to operate on edge platforms, e.g., autonomous vehicles.

In conclusion, this dissertation concentrates on proposing efficient and robust machine learning methods and works on several challenging traffic sensing applications. Inspired by the growing traffic analysis demand, this dissertation develops algorithms on edge devices and simultaneously optimizes the accuracy and efficiency of the traffic video sensing system. This dissertation selects four representative application scenarios, e.g., parking monitoring and collision avoidance, to evaluate the proposed algorithms. The experiment results are promis-

ing and display the highest accuracy in multiple downstream tasks compared to state-of-the-art algorithms. Besides, after passing through the proposed optimization pipeline, these proposed computer vision algorithms achieve a much higher inference speed while maintaining accuracy.

7.2 Future Works

This dissertation provides several generalized pipelines and methods which can transfer to other transportation applications beyond vision-based traffic sensing. Meanwhile, this dissertation has made remarkable progress in improving the efficiency and robustness of vision-based traffic sensing algorithms. The achievements build a solid foundation for future works. This section summarizes the next-step works mentioned in the previous chapters from the methodology and application perspectives. Edge computing is a promising research field for ITS applications. As more powerful edge devices, e.g., Nvidia Jetson Nano and Nvidia AGX Orin, are released, these devices can accommodate larger models for multi-tasking jobs. Deep learning compression strategies, e.g., model pruning, low-rank approximation, and neural architecture search, can be explored and optimized to improve system performance under high traffic data volume conditions. Besides general compression technologies, research on hardware-based operator optimization is necessary for more profound acceleration. Data insufficiency is an unavoidable challenge, so FSL or even zero-shot learning has excellent potential when lacking annotated training samples. These methods can utilize prior knowledge to improve the training process. However, it is still essential to annotate more uncommon or novel categories for the long-term goal. Besides, FSL and zero-shot learning can also enhance data annotation services, which traditionally depend on purely human labeling and

are relatively inefficient.

There are also more additional works from the application perspective. Chapter 4 designs a multi-tasking crowd monitoring framework, including crowd counting, group clustering, and motion detection. After obtaining the current crowd motion status, the next step is motion prediction, inspired by research works in multi-agent motion prediction [31] to assess and predict the crowd safety level. Considering traffic sensing in adverse conditions is complex, sensor fusion can make up for drawbacks when deploying a single kind of sensor. This dissertation studies the fusion of RGB and infrared cameras and achieves an impressively low MR. As the price of LiDAR keeps dropping, the large-scale deployment of LiDAR becomes possible. LiDAR can provide depth information compared to cameras and contribute to 3D scene reconstruction.

References

[1] Ackermann, S., Schawinski, K., Zhang, C., Weigel, A. K., & Turp, M. D. (2018). Using transfer learning to detect galaxy mergers. *Monthly Notices of the Royal Astronomical Society*, 479(1), 415–425.

[2] Afrin, T. & Yodo, N. (2020). A survey of road traffic congestion measures towards a sustainable and resilient transportation system. *Sustainability*, 12(11), 4660.

[3] Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Parra, H., & Ruiz, H. (2017a). Pedestrian detection for uavs using cascade classifiers with meanshift. In *2017 IEEE 11th international conference on semantic computing (ICSC)* (pp. 509–514).: IEEE.

[4] Aguilar, W. G., Luna, M. A., Moya, J. F., Luna, M. P., Abad, V., Ruiz, H., & Parra, H. (2017b). Real-time detection and simulation of abnormal crowd behavior. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (pp. 420–428).: Springer.

[5] Alahi, A., Bierlaire, M., & Vandergheynst, P. (2014). Robust real-time pedestrians detection in urban environments with low-resolution cameras. *Transportation research part C: emerging technologies*, 39, 113–128.

[6] Algan, G. & Ulusoy, I. (2020). Label noise types and their effects on deep learning. *arXiv preprint arXiv:2003.10471*.

[7] Ali, S. & Shah, M. (2008). Floor fields for tracking in high density crowd scenes. In *European conference on computer vision* (pp. 1–14).: Springer.

[8] Alwani, M., Chen, H., Ferdman, M., & Milder, P. (2016). Fused-layer cnn accelerators. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 1–12).: IEEE.

[9] Amato, G., Carrara, F., Falchi, F., Gennaro, C., & Vairo, C. (2016). Car parking occupancy detection using smart camera networks and deep learning. In *2016 IEEE Symposium on Computers and Communication (ISCC)* (pp. 1212–1217).: IEEE.

[10] Arunmozhi, A. & Park, J. (2018). Comparison of hog, lbp and haar-like features for on-road vehicle detection. In *2018 IEEE International Conference on Electro/Information Technology (EIT)* (pp. 0362–0367).: IEEE.

[11] Asiyanbola, R. & Akinpelu, A. (2012). The challenges of on-street parking in nigerian cities' transportation routes. *International journal of development and sustainability*, 1(2), 476–489.

[12] Aygün, S., Güneş, E. O., Subaşı, M. A., & Alkan, S. (2019). Sensor fusion for iot-based intelligent agriculture system. In *2019 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics)* (pp. 1–5).: IEEE.

[13] Bachman, P., Hjelm, R. D., & Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*.

[14] Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12), 2481–2495.

[15] Barata, E., Cruz, L., & Ferreira, J.-P. (2011). Parking at the uc campus: Problems and solutions. *Cities*, 28(5), 406–413.

[16] Barua, L., Zou, B., & Zhou, Y. (2020). Machine learning for international freight transportation management: a comprehensive review. *Research in Transportation Business & Management*, 34, 100453.

[17] Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., Garcia, F., & De La Escalera, A. (2018). Birdnet: a 3d object detection framework from lidar information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (pp. 3517–3523).: IEEE.

[18] Benevolo, C., Dameri, R. P., & D'auria, B. (2016). Smart mobility in smart city. In *Empowering organizations* (pp. 13–28). Springer.

[19] Bera, A. & Manocha, D. (2015). Reach-realtime crowd tracking using a hybrid motion model. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 740–747).: IEEE.

[20] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3464–3468).: IEEE.

[21] Bibi, N., Majid, M. N., Dawood, H., & Guo, P. (2017). Automatic parking space detection system. In *2017 2nd International Conference on Multimedia and Image Processing (ICMIP)* (pp. 11–15).: IEEE.

[22] Bitam, S. & Mellouk, A. (2012). Its-cloud: Cloud computing for intelligent transportation system. In *2012 IEEE global communications conference (GLOBECOM)* (pp. 2054–2059).: IEEE.

[23] Blue, V. J. & Adler, J. L. (1999). Cellular automata microsimulation of bidirectional pedestrian flows. *Transportation Research Record*, 1678(1), 135–141.

[24] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

[25] Boukerche, A. & Wang, J. (2020). Machine learning-based traffic prediction models for intelligent transportation systems. *Computer Networks*, 181, 107530.

[26] Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300, 17–33.

[27] Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, 249–259.

[28] Bura, H., Lin, N., Kumar, N., Malekar, S., Nagaraj, S., & Liu, K. (2018). An edge based smart parking solution using camera networks and deep learning. In *2018 IEEE International Conference on Cognitive Computing (ICCC)* (pp. 17–24).: IEEE.

[29] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., & Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11621–11631).

[30] Cao, Y., Song, H., Kaiwartya, O., Zhou, B., Zhuang, Y., Cao, Y., & Zhang, X. (2018). Mobile edge computing for big-data-enabled electric vehicle charging. *IEEE Communications Magazine*, 56(3), 150–156.

[31] Chai, Y., Sapp, B., Bansal, M., & Anguelov, D. (2019). Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*.

[32] Chan, A. B. & Vasconcelos, N. (2009). Bayesian poisson regression for crowd counting. In *2009 IEEE 12th international conference on computer vision* (pp. 545–551).: IEEE.

[33] Chang, B.-J. & Chiou, J.-M. (2019). Cloud computing-based analyses to predict vehicle driving shockwave for active safe driving in intelligent transportation system. *IEEE transactions on intelligent transportation systems*, 21(2), 852–866.

[34] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8748–8757).

[35] Chen, C., Liu, B., Wan, S., Qiao, P., & Pei, Q. (2021a). An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 22(3), 1840–1852.

[36] Chen, G., Choi, W., Yu, X., Han, T., & Chandraker, M. (2017a). Learning efficient object detection models with knowledge distillation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 742–751).

[37] Chen, H., Wang, Y., Wang, G., & Qiao, Y. (2018a). Lstd: A low-shot transfer detector for object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[38] Chen, J. & Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8), 1655–1674.

[39] Chen, M., Wang, Q., & Li, X. (2017b). Anchor-based group detection in crowd scenes. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1378–1382).: IEEE.

[40] Chen, Q., Wang, Y., Yang, T., Zhang, X., Cheng, J., & Sun, J. (2021b). You only look one-level feature. *arXiv preprint arXiv:2103.09460*.

[41] Chen, R.-C. et al. (2019a). Automatic license plate recognition via sliding-window darknet-yolo deep learning. *Image and Vision Computing*, 87, 47–56.

[42] Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607).: PMLR.

[43] Chen, T., Kornblith, S., Swersky, K., Norouzi, M., & Hinton, G. (2020b). Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*.

[44] Chen, T., Zhai, X., Ritter, M., Lucic, M., & Houlsby, N. (2019b). Self-supervised gans via auxiliary rotation loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12154–12163).

[45] Chen, X., Fan, H., Girshick, R., & He, K. (2020c). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

[46] Chen, X. & He, K. (2020). Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*.

[47] Chen, X., Xie, S., & He, K. (2021c). An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9640–9649).

[48] Chen, Y. & Shin, H. (2020). Pedestrian detection at night in infrared images using an attention-guided encoder-decoder convolutional neural network. *Applied Sciences*, 10(3), 809.

[49] Chen, Y., Xie, H., & Shin, H. (2018b). Multi-layer fusion techniques using a cnn for multispectral pedestrian detection. *IET Computer Vision*, 12(8), 1179–1187.

[50] Chen, Y., Zhang, P., Li, Z., Li, Y., Zhang, X., Meng, G., Xiang, S., Sun, J., & Jia, J. (2020d). Stitcher: feedback-driven data provider for object detection. *arXiv preprint arXiv:2004.12432*.

[51] Chen, Z., Pears, N., Freeman, M., & Austin, J. (2009). Road vehicle classification using support vector machines. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 4 (pp. 214–218).: IEEE.

[52] Cheng, J., Wu, J., Leng, C., Wang, Y., & Hu, Q. (2017a). Quantized cnn: A unified approach to accelerate and compress convolutional networks. *IEEE transactions on neural networks and learning systems*, 29(10), 4730–4743.

[53] Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017b). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

[54] Choi, H., Kim, S., Park, K., & Sohn, K. (2016). Multi-spectral pedestrian detection based on accumulated object proposal with fully convolutional networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)* (pp. 621–626).: IEEE.

[55] Church, J. S., Hegadoren, P., Paetkau, M., Miller, C., Regev-Shoshani, G., Schaefer, A., & Schwartzkopf-Genswein, K. (2014). Influence of environmental factors on infrared eye temperature measurements in cattle. *Research in veterinary science*, 96(1), 220–226.

[56] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 113–123).

[57] Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1 (pp. 886–893).: Ieee.

[58] Datondji, S. R. E., Dupuis, Y., Subirats, P., & Vasseur, P. (2016). A survey of vision-based traffic monitoring of road intersections. *IEEE transactions on intelligent transportation systems*, 17(10), 2681–2698.

[59] Davis, J. W. & Sharma, V. (2007). Background-subtraction using contour-based fusion of thermal and visible imagery. *Computer vision and image understanding*, 106(2-3), 162–182.

[60] De Almeida, P. R., Oliveira, L. S., Britto Jr, A. S., Silva Jr, E. J., & Koerich, A. L. (2015). Pklot–a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11), 4937–4949.

[61] Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., & Leal-Taixé, L. (2020). Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*.

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

[63] Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision* (pp. 1422–1430).

[64] Dollár, P., Appel, R., Belongie, S., & Perona, P. (2014). Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8), 1532–1545.

[65] Dollar, P., Wojek, C., Schiele, B., & Perona, P. (2011). Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4), 743–761.

[66] Dong, H., Wang, X., Zhang, C., He, R., Jia, L., & Qin, Y. (2018). Improved robust vehicle detection and identification based on single magnetic sensor. *Ieee Access*, 6, 5247–5255.

[67] Dong, L., Parameswaran, V., Ramesh, V., & Zoghlami, I. (2007). Fast crowd segmentation using shape indexing. In *2007 IEEE 11th International Conference on Computer Vision* (pp. 1–8).: IEEE.

[68] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[69] Dow, C.-R., Ngo, H.-H., Lee, L.-H., Lai, P.-Y., Wang, K.-C., & Bui, V.-T. (2020). A crosswalk pedestrian recognition system by using deep learning and zebra-crossing recognition techniques. *Software: Practice and Experience*, 50(5), 630–644.

[70] Du, S., Huang, T., Hou, J., Song, S., & Song, Y. (2019). Fpga based acceleration of game theory algorithm in edge computing for autonomous driving. *Journal of Systems Architecture*, 93, 33–39.

[71] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6569–6578).

[72] Eigen, D. & Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision* (pp. 2650–2658).

[73] El Hamdani, S., Benamar, N., & Younis, M. (2020). Pedestrian support in intelligent transportation systems: challenges, solutions and open issues. *Transportation research part C: emerging technologies*, 121, 102856.

[74] Elbamby, M. S., Perfecto, C., Liu, C.-F., Park, J., Samarakoon, S., Chen, X., & Bennis, M. (2019). Wireless edge computing with latency and reliability guarantees. *Proceedings of the IEEE*, 107(8), 1717–1737.

[75] Falconer, G. & Mitchell, S. (2012). Smart city framework. *Cisco Internet Business Solutions Group (IBSG)*, 12(9), 2–10.

[76] Fan, Q., Brown, L., & Smith, J. (2016). A closer look at faster r-cnn for vehicle detection. In *2016 IEEE intelligent vehicles symposium (IV)* (pp. 124–129).: IEEE.

[77] Fedorov, A., Nikolskaia, K., Ivanov, S., Shepelev, V., & Minbaleev, A. (2019). Traffic flow estimation with data from a video surveillance camera. *Journal of Big Data*, 6(1), 1–15.

[78] Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6202–6211).

[79] Feng, Y., Yuan, Y., & Lu, X. (2017). Learning deep event models for crowd anomaly detection. *Neurocomputing*, 219, 548–556.

[80] Ferdowsi, A., Challita, U., & Saad, W. (2019). Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview. *ieee vehicular technology magazine*, 14(1), 62–70.

[81] Ferrández-Pastor, F.-J., Mora, H., Jimeno-Morenilla, A., & Volckaert, B. (2018). Deployment of iot edge and fog computing technologies to develop smart building services. *Sustainability*, 10(11), 3832.

[82] Fraifer, M. & Fernström, M. (2016). Smart car parking system prototype utilizing cctv nodes: A proof of concept prototype of a novel approach towards iot-concept based smart parking. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)* (pp. 649–654).: IEEE.

[83] Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., & Berg, A. C. (2017). Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*.

[84] Funke, J., Brown, M., Erlien, S. M., & Gerdes, J. C. (2016). Collision avoidance and stabilization for autonomous vehicles in emergency scenarios. *IEEE Transactions on Control Systems Technology*, 25(4), 1204–1216.

[85] Fürst, M., Wasenmüller, O., & Stricker, D. (2020). Lrpd: Long range 3d pedestrian detection leveraging specific strengths of lidar and rgb. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1–7).: IEEE.

[86] Gajjar, A., Zhang, Y., & Yang, X. (2017). A smart building system integrated with an edge computing algorithm and iot mesh networks: demo abstract. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (pp. 1–2).

[87] Gandhi, T. & Trivedi, M. M. (2007). Pedestrian protection systems: Issues, survey, and challenges. *IEEE Transactions on intelligent Transportation systems*, 8(3), 413–430.

[88] Gao, J., Lin, W., Zhao, B., Wang, D., Gao, C., & Wen, J. (2019). C^3 framework: An open-source pytorch code for crowd counting. *arXiv preprint arXiv:1907.02724*.

[89] Gawande, U., Hajari, K., & Golhar, Y. (2020). Pedestrian detection and tracking in video surveillance system: issues, comprehensive review, and challenges. *Recent Trends in Computational Intelligence*, (pp. 1–24).

[90] Gazzaley, A. & Nobre, A. C. (2012). Top-down modulation: bridging selective attention and working memory. *Trends in cognitive sciences*, 16(2), 129–135.

[91] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11), 1231–1237.

[92] Geirhos, R., Janssen, D. H., Schütt, H. H., Rauber, J., Bethge, M., & Wichmann, F. A. (2017). Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*.

[93] Ghose, D., Desai, S. M., Bhattacharya, S., Chakraborty, D., Fiterau, M., & Rahman, T. (2019). Pedestrian detection in thermal images using saliency maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 0–0).

[94] Gidaris, S. & Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4367–4375).

[95] Girshick, R. (2015). Fast r-cnn.

[96] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

[97] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1), 142–158.

[98] Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, (pp. 1–31).

[99] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

[100] Guan, D., Cao, Y., Yang, J., Cao, Y., & Yang, M. Y. (2019a). Fusion of multispectral data through illumination-aware deep neural networks for pedestrian detection. *Information Fusion*, 50, 148–157.

[101] Guan, D., Luo, X., Cao, Y., Yang, J., Cao, Y., Vosselman, G., & Ying Yang, M. (2019b). Unsupervised domain adaptation for multispectral pedestrian detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 0–0).

[102] Ha, K., Pillai, P., Lewis, G., Simanta, S., Clinch, S., Davies, N., & Satyanarayanan, M. (2013). The impact of mobile multimedia applications on data center consolidation. In *2013 IEEE international conference on cloud engineering (IC2E)* (pp. 166–176).: IEEE.

[103] Hamurcu, M. & Eren, T. (2020). Strategic planning based on sustainability for urban transportation: An application to decision-making. *Sustainability*, 12(9), 3589.

[104] Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al. (2020). A survey on visual transformer. *arXiv preprint arXiv:2012.12556*.

[105] Hastie, T., Tibshirani, R., & Friedman, J. (2009). Overview of supervised learning. In *The elements of statistical learning* (pp. 9–41). Springer.

[106] Hbaieb, A., Rezgui, J., & Chaari, L. (2019). Pedestrian detection for autonomous driving within cooperative communication system. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1–6).: IEEE.

[107] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9729–9738).

[108] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961–2969).

[109] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9), 1904–1916.

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

[111] Helbing, D., Johansson, A., & Al-Abideen, H. Z. (2007). Dynamics of crowd disasters: An empirical study. *Physical review E*, 75(4), 046109.

[112] Hendrycks, D., Mazeika, M., Kadavath, S., & Song, D. (2019). Using self-supervised learning can improve model robustness and uncertainty. *Advances in Neural Information Processing Systems*, 32.

[113] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[114] HN, R. & Jyothi, S. (2021). Implementation of covid-19 social distance detection and suspicious human behavior recognition using machine learning. *IJO-International Journal of Electrical And Electronics Engineering*, 4(07), 01–06.

[115] Horn, B. K. & Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, 17(1-3), 185–203.

[116] Hou, X., Wang, Y., & Chau, L.-P. (2019). Vehicle tracking using deep sort with low confidence track filtering. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (pp. 1–6).: IEEE.

[117] Hou, Y.-L., Song, Y., Hao, X., Shen, Y., Qian, M., & Chen, H. (2018). Multispectral pedestrian detection based on deep convolutional neural networks. *Infrared Physics & Technology*, 94, 69–77.

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

[119] Huang, B., Mao, G., Qin, Y., & Wei, Y. (2019). Pedestrian flow estimation through passive wifi sensing. *IEEE Transactions on Mobile Computing*, 20(4), 1529–1542.

[120] Huang, G., Laradji, I., Vazquez, D., Lacoste-Julien, S., & Rodriguez, P. (2021). A survey of self-supervised and few-shot object detection. *arXiv preprint arXiv:2110.14711*.

[121] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7310–7311).

[122] Huang, Q., Zhou, K., You, S., & Neumann, U. (2018a). Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 709–718).: IEEE.

[123] Huang, X., Ge, Z., Jie, Z., & Yoshie, O. (2020). Nms by representative region: Towards crowded pedestrian detection by proposal pairing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10750–10759).

[124] Huang, X., Yu, R., Pan, M., & Shu, L. (2018b). Secure roadside unit hotspot against eavesdropping based traffic analysis in edge computing based internet of vehicles. *IEEE Access*, 6, 62371–62383.

[125] Hussain, M. M., Khan, F., Alam, M. S., & Beg, M. S. (2019). Fog computing for ubiquitous transportation applications—a smart parking case study. In *Engineering Vibration, Communication and Information Processing* (pp. 241–252). Springer.

[126] Hwang, S., Park, J., Kim, N., Choi, Y., & So Kweon, I. (2015). Multispectral pedestrian detection: Benchmark dataset and baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1037–1045).

[127] Ibrahim, M., Ali, N., et al. (2017). Comparison of forward vehicle detection using haar-like features and histograms of oriented gradients (hog) technique for feature extraction in cascade classifier. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2-13), 101–105.

[128] Idrees, H., Saleemi, I., Seibert, C., & Shah, M. (2013). Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2547–2554).

[129] Idrees, H., Tayyab, M., Athrey, K., Zhang, D., Al-Maadeed, S., Rajpoot, N., & Shah, M. (2018). Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 532–546).

[130] Imprialou, M. & Quddus, M. (2019). Crash data quality for road safety research: current state and future directions. *Accident Analysis & Prevention*, 130, 84–90.

[131] Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*.

[132] Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).: PMLR.

[133] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2704–2713).

[134] Jaworski, P., Edwards, T., Moore, J., & Burnham, K. (2011). Cloud computing concept for intelligent transportation systems. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (pp. 391–936).: IEEE.

[135] Jelić, B., Grbić, R., Vranješ, M., & Bjelica, M. (2020). Urtra2d–urban traffic 2d object detection dataset. In *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)* (pp. 1–6).: IEEE.

[136] Jeong, J., Park, H., & Kwak, N. (2017). Enhancement of ssd by concatenating feature maps for object detection. *arXiv preprint arXiv:1705.09587*.

[137] Jin, Z. & Bhanu, B. (2012). Single camera multi-person tracking based on crowd simulation. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)* (pp. 3660–3663).: IEEE.

[138] Junior, J. C. S. J., Musse, S. R., & Jung, C. R. (2010). Crowd analysis using computer vision techniques. *IEEE Signal Processing Magazine*, 27(5), 66–77.

[139] Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., & Darrell, T. (2019a). Few-shot object detection via feature reweighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 8420–8429).

[140] Kang, D. & Chan, A. (2018). Crowd counting by adaptively fusing predictions from an image pyramid. *arXiv preprint arXiv:1805.06115*.

[141] Kang, D., Ma, Z., & Chan, A. B. (2018). Beyond counting: Comparisons of density maps for crowd analysis tasks—counting, detection, and tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(5), 1408–1422.

[142] Kang, G., Dong, X., Zheng, L., & Yang, Y. (2017). Patchshuffle regularization. *arXiv preprint arXiv:1707.07103*.

[143] Kang, Z., Zhang, L., & Li, K. (2019b). An improved social force model for pedestrian dynamics in shipwrecks. *Applied Mathematics and Computation*, 348, 355–362.

[144] Kannadaguli, P. (2020). Fcos based seatbelt detection system using thermal imaging for monitoring traffic rule violations. In *2020 4th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)* (pp. 1–6).: IEEE.

[145] Karaman, O., Alhudhaif, A., & Polat, K. (2021). Development of smart camera systems based on artificial intelligence network for social distance detection to fight against covid-19. *Applied Soft Computing*, 110, 107610.

[146] Katzmann, A., Taubmann, O., Ahmad, S., Mühlberg, A., Sühling, M., & Groß, H.-M. (2021). Explaining clinical decision support systems in medical imaging using cycle-consistent activation maximization. *Neurocomputing*, 458, 141–156.

[147] Ke, R., Li, Z., Tang, J., Pan, Z., & Wang, Y. (2018). Real-time traffic flow parameter estimation from uav video based on ensemble classifier and optical flow. *IEEE Transactions on Intelligent Transportation Systems*, 20(1), 54–64.

[148] Ke, R., Zhuang, Y., Pu, Z., & Wang, Y. (2020). A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on iot devices. *IEEE Transactions on Intelligent Transportation Systems*.

[149] Khaidem, L., Luca, M., Yang, F., Anand, A., Lepri, B., & Dong, W. (2020). Optimizing transportation dynamics at a city-scale using a reinforcement learning framework. *IEEE Access*, 8, 171528–171541.

[150] Khan, S. I., Khan, A., Sarker, M. N. I., Huda, N., Zaman, M. R., Nurullah, A., & Rahman, M. Z. (2018). Traffic congestion in dhaka city: suffering for city dwellers and challenges for sustainable development. *European Journal of Social Sciences*, 57, 116–127.

[151] Khanna, A. & Anand, R. (2016). Iot based smart parking system. In *2016 International Conference on Internet of Things and Applications (IOTA)* (pp. 266–270).: IEEE.

[152] Kielar, P. M., Handel, O., Biedermann, D. H., & Borrmann, A. (2014). Concurrent hierarchical finite state machines for modeling pedestrian behavioral tendencies. *Transportation Research Procedia*, 2, 576–584.

[153] Kim, J.-a., Sung, J.-Y., & Park, S.-h. (2020). Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition. In *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)* (pp. 1–4).: IEEE.

[154] Kirschner, F. & Lanzendorf, M. (2020). Parking management for promoting sustainable transport in urban neighbourhoods. a review of existing policies and challenges from a german perspective. *Transport Reviews*, 40(1), 54–75.

[155] Kisantal, M., Wojna, Z., Murawski, J., Naruniec, J., & Cho, K. (2019). Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*.

[156] Koehrsen, W. (2018). Overfitting vs. underfitting: A complete example. *Towards Data Science*.

[157] Köhler, M., Eisenbach, M., & Gross, H.-M. (2021). Few-shot object detection: A survey. *arXiv preprint arXiv:2112.11699*.

[158] Konig, D., Adam, M., Jarvers, C., Layher, G., Neumann, H., & Teutsch, M. (2017). Fully convolutional region proposal networks for multispectral person detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 49–56).

[159] Kratz, L. & Nishino, K. (2011). Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, 34(5), 987–1002.

[160] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097–1105.

[161] Ku, Y.-J. & Dey, S. (2019). Sustainable vehicular edge computing using local and solar-powered roadside unit resources. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)* (pp. 1–7).: IEEE.

[162] Kukkala, V. K., Tunnell, J., Pasricha, S., & Bradley, T. (2018). Advanced driver-assistance systems: A path toward autonomous vehicles. *IEEE Consumer Electronics Magazine*, 7(5), 18–25.

[163] Kumar, N., Acharya, D., & Lohani, D. (2020). An iot-based vehicle accident detection and classification system using sensor fusion. *IEEE Internet of Things Journal*, 8(2), 869–880.

[164] Kuo, C.-H., Huang, C., & Nevatia, R. (2010). Multi-target tracking by on-line learned discriminative appearance models. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 685–692).: IEEE.

[165] Lan, W., Dang, J., Wang, Y., & Wang, S. (2018). Pedestrian detection based on yolo network model. In *2018 IEEE international conference on mechatronics and automation (ICMA)* (pp. 1547–1551).: IEEE.

[166] Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R., & Menotti, D. (2018). A robust real-time automatic license plate recognition based on the yolo detector. In *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–10).: IEEE.

[167] Lau, B. P. L., Wijerathne, N., Ng, B. K. K., & Yuen, C. (2017). Sensor fusion for public space utilization monitoring in a smart city. *IEEE Internet of Things Journal*, 5(2), 473–481.

[168] Law, H. & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 734–750).

[169] Leal-Taixé, L., Milan, A., Reid, I., Roth, S., & Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*.

[170] Lee, C., Han, Y., Jeon, S., Seo, D., & Jung, I. (2016). Smart parking system using ultrasonic sensor and bluetooth communication in internet of things. *KIISE transactions on computing practices*, 22(6), 268–277.

[171] Lee, C., Park, S., Yang, T., & Lee, S.-H. (2019). Smart parking with fine-grained localization and user status sensing based on edge computing. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)* (pp. 1–5).: IEEE.

[172] Lee, J., Wang, J., Crandall, D., Šabanović, S., & Fox, G. (2017). Real-time, cloud-based object detection for unmanned aerial vehicles. In *2017 First IEEE International Conference on Robotic Computing (IRC)* (pp. 36–43).: IEEE.

[173] Li, C., Song, D., Tong, R., & Tang, M. (2018a). Multispectral pedestrian detection via simultaneous detection and segmentation. *arXiv preprint arXiv:1808.04818*.

[174] Li, C., Song, D., Tong, R., & Tang, M. (2019a). Illumination-aware faster r-cnn for robust multispectral pedestrian detection. *Pattern Recognition*, 85, 161–171.

[175] Li, Q., Chen, P., & Wang, R. (2019b). Edge computing for intelligent transportation system: A review. *Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health*, (pp. 130–137).

[176] Li, Y., Wang, H., Dang, L. M., Nguyen, T. N., Han, D., Lee, A., Jang, I., & Moon, H. (2020). A deep learning-based hybrid framework for object detection and recognition in autonomous driving. *IEEE Access*, 8, 194228–194239.

[177] Li, Y., Zhang, X., & Chen, D. (2018b). Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1091–1100).

[178] Lin, J., Yu, W., Yang, X., Zhao, P., Zhang, H., & Zhao, W. (2020). An edge computing based public vehicle system for smart transportation. *IEEE Transactions on Vehicular Technology*, 69(11), 12635–12651.

[179] Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

[180] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117–2125).

[181] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

[182] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).: Springer.

[183] Liu, G., Shi, H., Kiani, A., Khreishah, A., Lee, J., Ansari, N., Liu, C., & Yousef, M. M. (2021a). Smart traffic monitoring system using computer vision and edge computing. *IEEE Transactions on Intelligent Transportation Systems*.

[184] Liu, J., Gu, Y., Han, S., Zhang, Z., Guo, J., & Cheng, X. (2021b). Feature rescaling and fusion for tiny object detection. *IEEE Access*, 9, 62946–62955.

[185] Liu, L., Chen, C., Pei, Q., Maharjan, S., & Zhang, Y. (2020a). Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, (pp. 1–24).

[186] Liu, S., Huang, D., et al. (2018a). Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 385–400).

[187] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018b). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8759–8768).

[188] Liu, T., Fu, H. Y., Wen, Q., Zhang, D. K., & Li, L. F. (2018c). Extended faster r-cnn for long distance human detection: Finding pedestrians in uav images. In *2018 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 1–2).: IEEE.

[189] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016a). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21–37).: Springer.

[190] Liu, W., Salzmann, M., & Fua, P. (2019a). Context-aware crowd counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5099–5108).

[191] Liu, W., Salzmann, M., & Fua, P. (2020b). Estimating people flows to better count them in crowded scenes. In *European Conference on Computer Vision* (pp. 723–740).: Springer.

[192] Liu, X., Liu, W., Ma, H., & Fu, H. (2016b). Large-scale vehicle re-identification in urban surveillance videos. In *2016 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 1–6).: IEEE.

[193] Liu, X., Zhang, F., Hou, Z., Wang, Z., Mian, L., Zhang, J., & Tang, J. (2020c). Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218*, 1(2).

[194] Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., & Yu, S. X. (2019b). Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2537–2546).

[195] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

[196] Loong, D. N. C., Isaak, S., & Yusof, Y. (2019). Machine vision based smart parking system using internet of things. *Telkomnika*, 17(4), 2098–2106.

[197] Lu, T., Buehler, R., Mondschein, A., & Hankey, S. (2017). Designing a bicycle and pedestrian traffic monitoring program to estimate annual average daily traffic in a small rural college town. *Transportation research part D: transport and environment*, 53, 193–204.

[198] Lu, Y., Lu, J., Zhang, S., & Hall, P. (2018). Traffic signal detection and classification in street views using an attention model. *Computational Visual Media*, 4(3), 253–266.

[199] Luo, X., Li, D., Yang, Y., & Zhang, S. (2019). Spatiotemporal traffic flow prediction with knn and lstm. *Journal of Advanced Transportation*, 2019.

[200] Lv, W., Song, W.-g., Ma, J., & Fang, Z.-m. (2013). A two-dimensional optimal velocity model for unidirectional pedestrian flow based on pedestrian's visual hindrance field. *IEEE Transactions on Intelligent Transportation Systems*, 14(4), 1753–1763.

[201] Majee, A., Agrawal, K., & Subramanian, A. (2021). Few-shot learning for road object detection. In *AAAI Workshop on Meta-Learning and MetaDL Challenge* (pp. 115–126).: PMLR.

[202] Majid Azimi, S. (2018). Shuffledet: Real-time vehicle detection network in on-board embedded uav imagery. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops* (pp. 0–0).

[203] Major, B., Fontijne, D., Ansari, A., Teja Sukhavasi, R., Gowaikar, R., Hamilton, M., Lee, S., Grzechnik, S., & Subramanian, S. (2019). Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (pp. 0–0).

[204] Martin, J., Mayberry, T., Donahue, C., Foppe, L., Brown, L., Riggins, C., Rye, E. C., & Brown, D. (2017). A study of mac address randomization in mobile devices and when it fails. *arXiv preprint arXiv:1703.02874*.

[205] Mazzon, R., Poiesi, F., & Cavallaro, A. (2013). Detection and tracking of groups in crowd. In *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance* (pp. 202–207).: IEEE.

[206] Meinhold, R. J. & Singpurwalla, N. D. (1983). Understanding the kalman filter. *The American Statistician*, 37(2), 123–127.

[207] Menshov, S., Wang, Y., Zhdanov, A., Varlamov, E., & Zhdanov, D. (2019). Simple online and realtime tracking people with new "soft-iou" metric. In *AOPC 2019: AI in*

Optics and Photonics, volume 11342 (pp. 113420M).: International Society for Optics and Photonics.

[208] Migacz, S. (2017). 8-bit inference with tensorrt. In *GPU technology conference*, volume 2 (pp.5).

[209] Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

[210] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[211] Monteiro, G., Peixoto, P., & Nunes, U. (2006). Vision-based pedestrian detection using haar-like features. *Robotica*, 24, 46–50.

[212] Moubayed, A., Shami, A., Heidari, P., Larabi, A., & Brunner, R. (2020). Edge-enabled v2x service placement for intelligent transportation systems. *IEEE Transactions on Mobile Computing*, 20(4), 1380–1392.

[213] Muhammad, K., Ullah, A., Lloret, J., Del Ser, J., & de Albuquerque, V. H. C. (2020). Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 22(7), 4316–4336.

[214] Murugan, V., Vijaykumar, V., & Nidhila, A. (2019). A deep learning rcnn approach for vehicle recognition in traffic surveillance system. In *2019 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0157–0160).: IEEE.

[215] Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., & Van Gool, L. (2018). Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)* (pp. 286–291).: IEEE.

[216] Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring generalization in deep learning. *arXiv preprint arXiv:1706.08947*.

[217] Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1520–1528).

[218] O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., & Walsh, J. (2019). Deep learning vs. traditional computer vision. In *Science and Information Conference* (pp. 128–144).: Springer.

[219] Paden, B., Čáp, M., Yong, S. Z., Yershov, D., & Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1), 33–55.

[220] Paidi, V., Fleyeh, H., Håkansson, J., & Nyberg, R. G. (2018). Smart parking sensors, technologies and applications for open parking lots: a review. *IET Intelligent Transport Systems*, 12(8), 735–741.

[221] Palanisamy, G. & Manikandan, T. (2017). Group behaviour profiling for detection of anomaly in crowd. In *2017 International Conference on Technical Advancements in Computers and Communications (ICTACC)* (pp. 11–15).: IEEE.

[222] Pan, C., Yan, Z., Xu, X., Sun, M., Shao, J., & Wu, D. (2013). Vehicle logo recognition based on deep learning architecture in video surveillance for intelligent traffic system. In *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)* (pp. 123–126).: IET.

[223] Pang, Y., Wang, T., Anwer, R. M., Khan, F. S., & Shao, L. (2019). Efficient featurized image pyramid network for single shot detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7336–7344).

[224] Park, K., Kim, S., & Sohn, K. (2018). Unified multi-spectral pedestrian detection based on probabilistic fusion networks. *Pattern Recognition*, 80, 143–155.

[225] Polino, A., Pascanu, R., & Alistarh, D. (2018). Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.

[226] Pu, Z., Cui, Z., Tang, J., Wang, S., & Wang, Y. (2021). Multi-modal traffic speed monitoring: A real-time system based on passive wi-fi and bluetooth sensing technology. *IEEE Internet of Things Journal*.

[227] Pu, Z., Zhang, Q., Zhuang, Y., Lv, Y., & Wang, Y. (2020). A device-free wi-fi sensing method for pedestrian monitoring using channel state information. In *International Conference on Transportation and Development 2020* (pp. 207–220).: American Society of Civil Engineers Reston, VA.

[228] Qian, Y., Dolan, J. M., & Yang, M. (2019). Dlt-net: Joint detection of drivable areas, lane lines, and traffic objects. *IEEE Transactions on Intelligent Transportation Systems*, 21(11), 4670–4679.

[229] Qureshi, K. N. & Abdullah, A. H. (2013). A survey on intelligent transportation systems. *Middle-East Journal of Scientific Research*, 15(5), 629–642.

[230] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

[231] Ranaweera, P., Jurcut, A. D., & Liyanage, M. (2021). Survey on multi-access edge computing security and privacy. *IEEE Communications Surveys & Tutorials*.

[232] Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision* (pp. 525–542).: Springer.

[233] Ravanbakhsh, M., Nabi, M., Mousavi, H., Sangineto, E., & Sebe, N. (2018). Plug-and-play cnn for crowd motion analysis: An application in abnormal event detection. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1689–1698).: IEEE.

[234] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

[235] Redmon, J. & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[236] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks.

[237] Ren, W., Wang, X., Tian, J., Tang, Y., & Chan, A. B. (2020). Tracking-by-counting: Using network flows on crowd density maps for tracking multiple targets. *IEEE Transactions on Image Processing*, 30, 1439–1452.

[238] Retting, R. A., Williams, A., Farmer, C. M., & Feldman, A. F. (1999). Evaluation of red light camera enforcement in fairfax, va., usa. *ITE journal*, 69, 30–35.

[239] Revathi, G. & Dhulipala, V. S. (2012). Smart parking systems and sensors: A survey. In *2012 International Conference on Computing, Communication and Applications* (pp. 1–5).: IEEE.

[240] Rodríguez, A., Valverde, J., Portilla, J., Otero, A., Riesgo, T., & De la Torre, E. (2018). Fpga-based high-performance embedded systems for adaptive edge computing in cyber-physical systems: The artico3 framework. *Sensors*, 18(6), 1877.

[241] Rodriguez, M., Ali, S., & Kanade, T. (2009). Tracking in unstructured crowded scenes. In *2009 IEEE 12th International Conference on Computer Vision* (pp. 1389–1396).: IEEE.

[242] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241).: Springer.

[243] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation*. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

[244] Saarika, P., Sandhya, K., & Sudha, T. (2017). Smart transportation system using iot. In *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)* (pp. 1104–1107).: IEEE.

[245] Sajid, U., Sajid, H., Wang, H., & Wang, G. (2020). Zoomcount: A zooming mechanism for crowd counting in static images. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(10), 3499–3512.

[246] Sam, D. B., Surya, S., & Babu, R. V. (2017). Switching convolutional neural network for crowd counting. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4031–4039).: IEEE.

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

[248] Sarker, V. K., Gia, T. N., Ben Dhaou, I., & Westerlund, T. (2020). Smart parking system with dynamic pricing, edge-cloud computing and lora. *Sensors*, 20(17), 4669.

[249] Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1), 30–39.

[250] Setjo, C. H., Achmad, B., et al. (2017). Thermal image human detection using haar-cascade classifier. In *2017 7th International Annual Engineering Seminar (InAES)* (pp. 1–6).: IEEE.

[251] Shahid, M. R., Blanc, G., Zhang, Z., & Debar, H. (2018). Iot devices recognition through network traffic analysis. In *2018 IEEE international conference on big data (big data)* (pp. 5187–5192).: IEEE.

[252] Shao, Z., Cheng, G., Ma, J., Wang, Z., Wang, J., & Li, D. (2021). Real-time and accurate uav pedestrian detection for social distancing monitoring in covid-19 pandemic. *IEEE Transactions on Multimedia*.

[253] Sharma, S. K., Modanval, R. K., Tayal, P., & Gayathri, N. (2022). Social distancing and crowd density distribution system for public places and public transports using computer vision and nlp. In *Advanced Computing and Intelligent Technologies* (pp. 461–479). Springer.

[254] Shen, T., Hong, Y., Thompson, M. M., Liu, J., Huo, X., & Wu, L. (2020). How does parking availability interplay with the land use and affect traffic congestion in urban areas? the case study of xi'an, china. *Sustainable Cities and Society*, 57, 102126.

[255] Shi, H. & Liu, C. (2018). A new foreground segmentation method for video analysis in different color spaces. In *2018 24th International Conference on Pattern Recognition (ICPR)* (pp. 2899–2904).: IEEE.

[256] Shi, J. et al. (1994). Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 593–600).: IEEE.

[257] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5), 637–646.

[258] Shirazi, M. S. & Morris, B. T. (2016). Looking at intersections: a survey of intersection monitoring, behavior and safety analysis of recent studies. *IEEE Transactions on Intelligent Transportation Systems*, 18(1), 4–24.

[259] Shorten, C. & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1–48.

[260] Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 761–769).

[261] Sifuentes, E., Casas, O., & Pallas-Areny, R. (2011). Wireless magnetic sensor node for vehicle detection with optical wake-up. *IEEE Sensors journal*, 11(8), 1669–1676.

[262] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[263] Sindagi, V. A. & Patel, V. M. (2017). Generating high-quality crowd density maps using contextual pyramid cnns. In *Proceedings of the IEEE international conference on computer vision* (pp. 1861–1870).

[264] Sindagi, V. A. & Patel, V. M. (2018). A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107, 3–16.

[265] Siva, P., Javad Shafiee, M., Jamieson, M., & Wong, A. (2016). Real-time, embedded scene invariant crowd counting using scale-normalized histogram of moving gradients (homg). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 67–74).

[266] Sligar, A. P. (2020). Machine learning-based radar perception for autonomous vehicles using full physics simulation. *IEEE Access*, 8, 51470–51476.

[267] Solmaz, G., Berz, E. L., Dolatabadi, M. F., Aytaç, S., Fürst, J., Cheng, B., & Ouden, J. d. (2019). Learn from iot: pedestrian detection and intention prediction for autonomous driving. In *Proceedings of the 1st ACM Workshop on Emerging Smart Technologies and Infrastructures for Smart Mobility and Sustainability* (pp. 27–32).

[268] Song, L., Pan, P., Zhao, K., Yang, H., Chen, Y., Zhang, Y., Xu, Y., & Jin, R. (2020). Large-scale training system for 100-million classification at alibaba. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2909–2930).

[269] Sreenu, G. & Durai, M. S. (2019). Intelligent video surveillance: a review through deep learning techniques for crowd analysis. *Journal of Big Data*, 6(1), 1–27.

[270] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929–1958.

[271] Steinhauser, D., Held, P., Thöresz, B., & Brandmeier, T. (2021). Towards safe autonomous driving: Challenges of pedestrian detection in rain with automotive radar. In *2020 17th European Radar Conference (EuRAD)* (pp. 409–412).: IEEE.

[272] Subburaman, V. B., Descamps, A., & Carincotte, C. (2012). Counting people in the crowd using a generic head detector. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance* (pp. 470–475).: IEEE.

[273] Summers, C. & Dinneen, M. J. (2019). Improved mixed-example data augmentation. In *2019 IEEE winter conference on applications of computer vision (WACV)* (pp. 1262–1270).: IEEE.

[274] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2446–2454).

[275] Sun, Z., Bebis, G., & Miller, R. (2006). On-road vehicle detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 28(5), 694–711.

[276] Suryakala, S., Muthumeenakshi, K., & Gladwin, S. J. (2019). Vision based vehicle/pedestrian detection in traffic surveillance system. In *2019 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0506–0510).: IEEE.

[277] Sutar, S. H., Koul, R., & Suryavanshi, R. (2016). Integration of smart phone and iot for development of smart public transportation system. In *2016 international conference on internet of things and applications (IOTA)* (pp. 73–78).: IEEE.

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

[279] Tabernik, D. & Skočaj, D. (2019). Deep learning for large-scale traffic-sign detection and recognition. *IEEE transactions on intelligent transportation systems*, 21(4), 1427–1440.

[280] Talpes, E., Sarma, D. D., Venkataramanan, G., Bannon, P., McGee, B., Floering, B., Jalote, A., Hsiong, C., Arora, S., Gorti, A., et al. (2020). Compute solution for tesla's full self-driving computer. *IEEE Micro*, 40(2), 25–35.

[281] Tan, J., Wang, C., Li, B., Li, Q., Ouyang, W., Yin, C., & Yan, J. (2020). Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11662–11671).

[282] Tan, M. & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (pp. 6105–6114).: PMLR.

[283] Tang, T., Deng, Z., Zhou, S., Lei, L., & Zou, H. (2017). Fast vehicle detection in uav images. In *2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP)* (pp. 1–5).: IEEE.

[284] Tang, Y., Ji, J., Gao, S., Dai, H., Yu, Y., & Todo, Y. (2018). A pruning neural network model in credit classification analysis. *Computational intelligence and neuroscience*, 2018.

[285] Taylor, L. & Nitschke, G. (2018). Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1542–1547).: IEEE.

[286] Thanasutives, P., Fukui, K.-i., Numao, M., & Kijsirikul, B. (2021). Encoder-decoder based convolutional neural networks with multi-scale-aware modules for crowd counting. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 2382–2389).: IEEE.

[287] Thoma, M. (2017). Analysis and optimization of convolutional neural network architectures. *arXiv preprint arXiv:1707.09725*.

[288] Tian, Z., Shen, C., Chen, H., & He, T. (2020). Fcos: A simple and strong anchor-free object detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[289] Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*.

[290] Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., & Jégou, H. (2021). Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*.

[291] True, N. (2007). Vacant parking space detection in static images. *University of California, San Diego*, 17, 659–662.

[292] Van der Maaten, L. & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

[293] Viola, P., Jones, M. J., & Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2), 153–161.

[294] Wagner, J., Fischer, V., Herman, M., & Behnke, S. (2016). Multispectral pedestrian detection using deep fusion convolutional neural networks. In *ESANN*, volume 587 (pp. 509–514).

[295] Wali, S. B., Abdullah, M. A., Hannan, M. A., Hussain, A., Samad, S. A., Ker, P. J., & Mansor, M. B. (2019). Vision-based traffic sign detection and recognition systems: Current trends and challenges. *Sensors*, 19(9), 2093.

[296] Wang, C., Xu, C., & Fan, P. (2020a). Effects of traffic enforcement cameras on macro-level traffic safety: A spatial modeling analysis considering interactions with roadway and land use characteristics. *Accident Analysis & Prevention*, 144, 105659.

[297] Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., & Yeh, I.-H. (2020b). Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 390–391).

[298] Wang, J., Zheng, H., Huang, Y., & Ding, X. (2017). Vehicle type recognition in surveillance images from labeled web-nature data using deep transfer learning. *IEEE Transactions on Intelligent Transportation Systems*, 19(9), 2913–2922.

[299] Wang, K. & Shen, Z. (2011). Artificial societies and gpu-based cloud computing for intelligent transportation management. *IEEE Intelligent Systems*, 26(4), 22–28.

[300] Wang, Q. & Breckon, T. P. (2019). Crowd counting via segmentation guided attention networks and curriculum loss. *arXiv preprint arXiv:1911.07990*.

[301] Wang, Q., Chen, M., Nie, F., & Li, X. (2018a). Detecting coherent groups in crowd scenes by multiview clustering. *IEEE transactions on pattern analysis and machine intelligence*, 42(1), 46–58.

[302] Wang, T., Qiao, M., Zhu, A., Shan, G., & Snoussi, H. (2020c). Abnormal event detection via the analysis of multi-frame optical flow information. *Frontiers of Computer Science*, 14(2), 304–313.

[303] Wang, W., Hong, W., Wang, F., & Yu, J. (2020d). Gan-knowledge distillation for one-stage object detection. *IEEE Access*, 8, 60719–60727.

[304] Wang, X., Huang, T. E., Darrell, T., Gonzalez, J. E., & Yu, F. (2020e). Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*.

[305] Wang, X. & Li, Z. (2016). Traffic and transportation smart with cloud computing on big data. *International Journal of Computer Science & Applications*, 13(1).

[306] Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., & Weinberger, K. Q. (2019a). Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8445–8453).

[307] Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020f). Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3), 1–34.

[308] Wang, Y., Zhang, W., Liu, Y., & Zhu, J. (2020g). Two-branch fusion network with attention map for crowd counting. *Neurocomputing*, 411, 1–8.

[309] Wang, Y.-X., Ramanan, D., & Hebert, M. (2019b). Meta-learning to detect rare objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9925–9934).

[310] Wang, Z., Ren, W., & Qiu, Q. (2018b). Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint arXiv:1807.01726*.

[311] Wang, Z., Zheng, L., Liu, Y., Li, Y., & Wang, S. (2020h). Towards real-time multi-object tracking. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16* (pp. 107–122).: Springer.

[312] Wang, Z., Zheng, L., Liu, Y., & Wang, S. (2019c). Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2(3), 4.

[313] Wang, Z.-R., Jia, Y.-L., Huang, H., & Tang, S.-M. (2008). Pedestrian detection using boosted hog features. In *2008 11th International IEEE Conference on Intelligent Transportation Systems* (pp. 1155–1160).: IEEE.

[314] Wehbe, R., Massaad, Z., & Otayek, E. (2017). Using intelligent transportation systems to enhance pedestrian safety at beirut signalized intersection. In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)* (pp. 574–579).: IEEE.

[315] Wei, P., Shi, H., Yang, J., Qian, J., Ji, Y., & Jiang, X. (2019). City-scale vehicle tracking and traffic flow estimation using low frame-rate traffic cameras. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers* (pp. 602–610).

[316] Wei, W., Song, H., Li, W., Shen, P., & Vasilakos, A. (2017). Gradient-driven parking navigation using a continuous information potential field based on wireless sensor network. *Information Sciences*, 408, 100–114.

[317] Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, 3(1), 1–40.

[318] Wikipedia contributors (2021). List of human stampedes and crushes — Wikipedia, the free encyclopedia. [Online; accessed 31-July-2021].

[319] Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)* (pp. 3645–3649).: IEEE.

[320] Wu, J., Song, L., Wang, T., Zhang, Q., & Yuan, J. (2020). Forest r-cnn: Large-vocabulary long-tailed object detection and instance segmentation. In *Proceedings of the 28th ACM International Conference on Multimedia* (pp. 1570–1578).

[321] Wu, Q., Huang, C., Wang, S.-y., Chiu, W.-c., & Chen, T. (2007). Robust parking space detection considering inter-space correlation. In *2007 IEEE international conference on multimedia and expo* (pp. 659–662).: IEEE.

[322] Wu, T.-E., Tsai, C.-C., & Guo, J.-I. (2017). Lidar/camera sensor fusion technology for pedestrian detection. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (pp. 1675–1678).: IEEE.

[323] Wu, X., Liang, G., Lee, K. K., & Xu, Y. (2006). Crowd density estimation using texture analysis and learning. In *2006 IEEE international conference on robotics and biomimetics* (pp. 214–219).: IEEE.

[324] Xiao, Y. & Marlet, R. (2020). Few-shot object detection and viewpoint estimation for objects in the wild. In *European conference on computer vision* (pp. 192–210).: Springer.

[325] Xu, B., Ban, X. J., Bian, Y., Li, W., Wang, J., Li, S. E., & Li, K. (2018a). Cooperative method of traffic signal optimization and speed control of connected vehicles at isolated intersections. *IEEE Transactions on Intelligent Transportation Systems*, 20(4), 1390–1403.

[326] Xu, M., Ge, Z., Jiang, X., Cui, G., Lv, P., Zhou, B., & Xu, C. (2019). Depth information guided crowd counting for complex crowd scenes. *Pattern Recognition Letters*, 125, 563–569.

[327] Xu, Y., Wang, Y., Zhou, A., Lin, W., & Xiong, H. (2018b). Deep neural network compression with single and multiple level quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[328] Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., & Lin, L. (2019). Meta r-cnn: Towards general solver for instance-level low-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9577–9586).

[329] Yan, Y., Ren, J., Zhao, H., Sun, G., Wang, Z., Zheng, J., Marshall, S., & Soraghan, J. (2018). Cognitive fusion of thermal and visible imagery for effective detection and tracking of pedestrians in videos. *Cognitive Computation*, 10(1), 94–104.

[330] Yang, H., Liu, C., Zhuang, Y., Sun, W., Murthy, K., Pu, Z., & Wang, Y. (2021). Truck parking pattern aggregation and availability prediction by deep learning. *IEEE Transactions on Intelligent Transportation Systems*.

[331] Yang, H., Liu, W., Wang, X., & Zhang, X. (2013). On the morning commute problem with bottleneck congestion and parking space constraints. *Transportation Research Part B: Methodological*, 58, 106–118.

[332] Yao, Z., Dong, Z., Zheng, Z., Gholami, A., Yu, J., Tan, E., Wang, L., Huang, Q., Wang, Y., Mahoney, M., et al. (2021). Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning* (pp. 11875–11886).: PMLR.

[333] Ye, J., Chen, X., Yang, C., & Wu, J. (2008). Walking behavior and pedestrian flow characteristics for different types of walking facilities. *Transportation Research Record*, 2048(1), 43–51.

[334] Yi, S., Li, C., & Li, Q. (2015). A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data* (pp. 37–42).

[335] Yim, J., Joo, D., Bae, J., & Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4133–4141).

[336] Yin, R. (2019). Multi-resolution generative adversarial networks for tiny-scale pedestrian detection. In *2019 IEEE International Conference on Image Processing (ICIP)* (pp. 1665–1669).: IEEE.

[337] Yogameena, B. & Nagananthini, C. (2017). Computer vision based crowd disaster avoidance system: A survey. *International journal of disaster risk reduction*, 22, 95–129.

[338] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020a). Bdd100k: A diverse driving dataset for heterogeneous multitask learning.

In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2636–2645).

[339] Yu, F. & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

[340] Yu, X., Gong, Y., Jiang, N., Ye, Q., & Han, Z. (2020b). Scale match for tiny person detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1257–1265).

[341] Yu, X., Han, Z., Gong, Y., Jan, N., Zhao, J., Ye, Q., Chen, J., Feng, Y., Zhang, B., Wang, X., et al. (2020c). The 1st tiny object detection challenge: Methods and results. In *European Conference on Computer Vision* (pp. 315–323).: Springer.

[342] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6023–6032).

[343] Zagoruyko, S. & Komodakis, N. (2016). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*.

[344] Zantalis, F., Koulouras, G., Karabetsos, S., & Kandris, D. (2019). A review of machine learning and iot in smart transportation. *Future Internet*, 11(4), 94.

[345] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3), 107–115.

[346] Zhang, H., Wang, K., Tian, Y., Gou, C., & Wang, F.-Y. (2018). Mfr-cnn: Incorporating multi-scale features and global information for traffic object detection. *IEEE Transactions on Vehicular Technology*, 67(9), 8019–8030.

[347] Zhang, H., Zhang, L., & Jiang, Y. (2019a). Overfitting and underfitting analysis for deep learning based end-to-end communication systems. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)* (pp. 1–6).: IEEE.

[348] Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., & Chen, C. (2011). Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4), 1624–1639.

[349] Zhang, J.-S., Cao, J., & Mao, B. (2017). Application of deep learning and unmanned aerial vehicle technology in traffic flow monitoring. In *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 1 (pp. 189–194).: IEEE.

[350] Zhang, L., Liu, Z., Zhang, S., Yang, X., Qiao, H., Huang, K., & Hussain, A. (2019b). Cross-modality interactive attention network for multispectral pedestrian detection. *Information Fusion*, 50, 20–29.

[351] Zhang, Y., Yin, Z., Nie, L., & Huang, S. (2020). Attention based multi-layer fusion of multispectral images for pedestrian detection. *IEEE Access*, 8, 165071–165084.

[352] Zhang, Y., Zhou, C., Chang, F., & Kot, A. C. (2019c). Multi-resolution attention convolutional neural network for crowd counting. *Neurocomputing*, 329, 144–152.

[353] Zhang, Y., Zhou, D., Chen, S., Gao, S., & Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 589–597).

[354] Zhao, H., Gao, J., Lan, T., Sun, C., Sapp, B., Varadarajan, B., Shen, Y., Shen, Y., Chai, Y., Schmid, C., et al. (2020). Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*.

[355] Zhao, Q., Sheng, T., Wang, Y., Ni, F., & Cai, L. (2018). Cfenet: An accurate and efficient single-shot object detector for autonomous driving. *arXiv preprint arXiv:1806.09790*.

[356] Zhao, X., Gong, D., & Medioni, G. (2012). Tracking using motion patterns for very crowded scenes. In *European Conference on Computer Vision* (pp. 315–328).: Springer.

[357] Zhao, X., Zhang, J., & Song, W. (2021). A radar-nearest-neighbor based data-driven approach for crowd simulation. *Transportation Research Part C: Emerging Technologies*, 129, 103260.

[358] Zheng, Y., Capra, L., Wolfson, O., & Yang, H. (2014). Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3), 1–55.

[359] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34 (pp. 12993–13000).

[360] Zheng, Z., Zheng, L., & Yang, Y. (2018). Pedestrian alignment network for large-scale person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(10), 3037–3045.

[361] Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34 (pp. 13001–13008).

[362] Zhou, Y., Lau, B. P. L., Koh, Z., Yuen, C., & Ng, B. K. K. (2020). Understanding crowd behaviors in a social event by passive wifi sensing and data mining. *IEEE Internet of Things Journal*, 7(5), 4442–4454.

[363] Zhou, Z., Liao, H., Gu, B., Huq, K. M. S., Mumtaz, S., & Rodriguez, J. (2018). Robust mobile crowd sensing: When deep learning meets edge computing. *IEEE Network*, 32(4), 54–60.

[364] Zhu, F., Wang, X., & Yu, N. (2014). Crowd tracking with dynamic evolution of group structures. In *European conference on computer vision* (pp. 139–154).: Springer.

[365] Zhu, F., Wang, X., & Yu, N. (2016a). Crowd tracking by group structure evolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(3), 772–786.

[366] Zhu, M. & Gupta, S. (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

[367] Zhu, Y., Zhou, Y., Ye, Q., Qiu, Q., & Jiao, J. (2017). Soft proposal networks for weakly supervised object localization. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1841–1850).

[368] Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., & Hu, S. (2016b). Traffic-sign detection and classification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2110–2118).

[369] Zhuang, Y., Zheng, Y., Zhang, Y., & Xie, X. (2016). A ca model with variable cell size for passengers behavior in subway. In *International Conference on Intelligent Transportation* (pp. 303–318).: Springer.

[370] Ziebinski, A., Cupek, R., Grzechca, D., & Chruszczyk, L. (2017). Review of advanced driver assistance systems (adas). In *AIP Conference Proceedings*, volume 1906 (pp. 120002).: AIP Publishing LLC.