# Anomaly Detection on the Edge using Smart Cameras under Low-Light Conditions

**Yaser Abu Awwad**

School of Computer Science and Informatics

Cardiff University

This dissertation is submitted for the degree of

*Master of Philosophy (MPhil)*

November 2023

# Abstract

**The vast number of cameras utilised in smart city domains is becoming increasingly prominent and notable for monitoring indoor and outdoor areas such as buildings and road traffic as well as in rural areas (e.g. farms) in order to deter thefts of farming machinery and livestock, besides monitoring workers to guarantee their safety. In addition, to detect anomalies meant as Identifying an unusual occurrence that does not adhere to the nature and regulations. However, detecting anomalies becomes much more challenging in environments with low lighting and poor visibility conditions, such as at night (when a scene is entirely dark) and partially dark during dusk and dawn, resulting in obtaining inefficient outcomes to recognise events leading to an increase in false positives detection. Thus, this research aimed to identify objects (referred to as Anomalies) in low-light settings with the assistance of pre-existing methodologies in image enhancement and object recognition on resource-constrained devices (referred to as Nodes). Rather than focusing on enhancement methods for image quality comparison purposes or developing a novel approach, the main goal is to exploit existing methods to boost the detection stage's accuracy. Further, a lightweight classification algorithm is proposed to differentiate (1) Bright scenes captured in the daytime from low-light ones and (2) To distinguish between low-light scenes that might differ in their darkness levels and incorporate additional factors such as noise. Therefore, images with insufficient light are enhanced by multi-enhancement networks, where the optimal one is chosen based on the input image features and characteristics. The results demonstrated an increase of *25% & 3%* in the detection accuracy on the ExDark database. Moreover, the classifier could discern between bright and dark scenarios achieving an accuracy**

**of *85.24%*. Finally, the proposed classification, enhancement, and detection stages were implemented on the resource-constrained devices, demonstrating efficiency and resilience, retaining high performance and time-response roughly *(1 second)* across all phases.**

# Contents

# List of Figures

# List of Tables

# Dedication

My thesis is dedicated to my parents for the unconditional love and support they have shown me throughout my life, including the time I spent working on this thesis. Their direction, support, and faith in my abilities have been of immeasurable help. Thank you for being there for me every step of the way.

# Acknowledgements

I want to thank my supervisors, Dr Charith Perera & Prof. Omer Rana, for their excellent advice, support, and encouragement throughout my research. Since my study and overall comprehension of the topic significantly benefited from their extensive knowledge and insightful observations. Also, thanks again to them since they made me a part of this research based on a funded project by the Department for Digital, Culture, Media, and Sport (DCMS), named 5G Wales Unlocked, which allowed me to gain both academic and industry knowledge—in addition, improving my communication skills and my ability to work in a team by interacting with people from a variety of fields.

# Chapter 1

# Introduction

This chapter first discusses an overview of the 5G Wales Unlocked project, on which the current research is based, by describing assets, use cases and other related information. Then, the motivation for conducting this research due to the limitations explored and encountered during the project is explained. Finally, the aim, objectives and contributions are presented in separate sections at the end of this chapter.

## *5G Wales Unlocked Project (Overview)*

The 5G technology is the latest generation for broadband and cellular networks, established in 2019 by the 3rd Generation Partnership Project (3GPP). Many worldwide telecommunication industries have started deploying 5G on most cellular and broadband devices since it is up to 100 times faster than the current 4G standard technology, allowing the creation of never-before-seen opportunities for people and businesses [102]. This technology is promising by providing faster connectivity, low latency and the potential to connect multiple heterogeneous IoT devices. Moreover, the 5G revolution will impact the physical security industry, especially in Video Surveillance System (VSS) domains which play a vital role since they span large sites and city locations in outdoor environments, requiring faster and more reliable connectivity. This thesis is conducted based on the 5G Wales Unlocked project funded by the Government of Welsh Department for Digital, Culture, Media & Sport (DCMS). The 5G technology was invested in and utilized in the project to enhance different aspects of Wales's rural and semi-rural areas. Several cameras and sensors were incorporated and deployed in different use cases for data collection and scene understanding. For instance, a multi-camera video surveillance system was deployed in four different areas to monitor and safeguard lone workers on a farm in Monmouthshire, North Wales. The objective is to detect two types of objects (People and Vehicles) in order to prevent theft of machinery and livestock and to monitor farmers to ensure their safety

Moreover, the same camera was used to detect vandalism at Raglan Castle in Monmouthshire, prevent children from climbing walls, detect people in forbidden areas, and maintain safety, allowing immediate human intervention for making decisions. The above use cases focused on detecting abnormal events through video surveillance cameras, whether in behaviours or appearances. However, far from abnormality, the project also studied people's behaviours in Transport services by counting the number of people on bus transportation. In addition, Blaenau Gwent in South-East Wales utilized the same camera system of Farm and Castle scenarios for parking lot detection to determine free and occupied parking spaces and people at the bus stop.

The main idea is to measure and evaluate the video analytic approaches for accomplishing several or similar tasks instead of relying on high-cost installed sensors, which require installing a single sensor per spot—for instance, bus seats or park space for transportation and car parking scenarios. Moreover, modern cameras come in various functionalities, features and prices, a comprehensive range of views and zones *"field-of-view"*. Consequently, one or more cameras may cover a specific area for controlling and managing similar tasks as pricey non-vision sensors can do with more efficiency and reliability. Indeed, depending only on non-vision sensors will require a human examination and confirmation of a suspected fire if a temperature sensor returns a high or suspicious measurement, for instance. In addition, extra sensors may be required to replace damaged or failed ones to maintain system redundancy, hence increasing the cost of maintenance and replacement. Alternatively, systems that depend on both kinds of sensors (vision and non-vision) have a greater possibility of achieving both redundancy and precision in identifying abnormalities and dangerous activities.

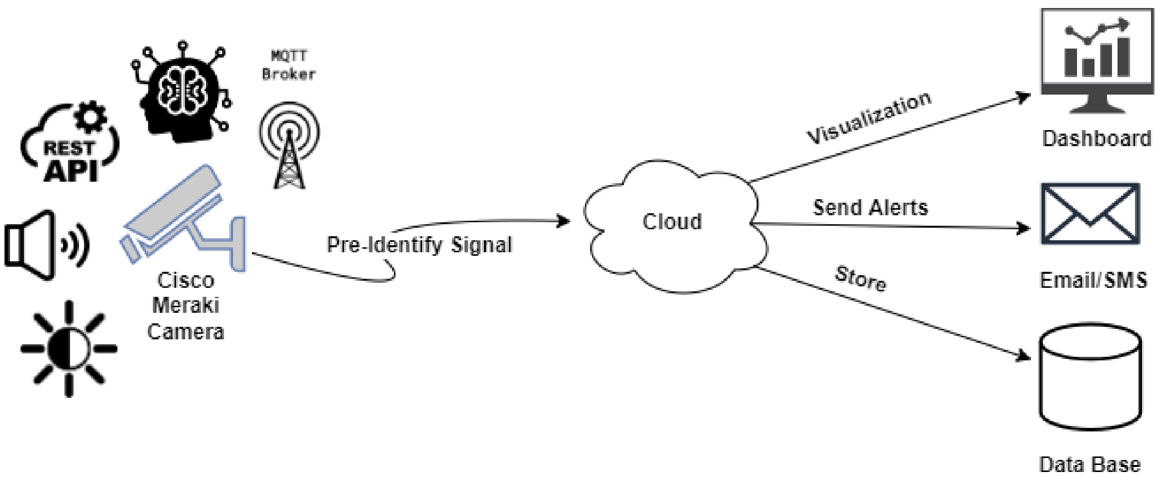


**Figure 1.1** The video analytic flow of the standard Meraki MV Camera.

Figure 1.1 shows a basic flow for the video analytic system exploiting only the capability of the Meraki MV Smart Camera *(MV72X)*. Indeed, many functions are embedded within the camera for extracting valuable information for a specific scene; where these functions are:

1. A tiny machine learning algorithm to detect only two types of instances (people and vehicles), producing a message consisting of:
   (a) Timestamp.
   (b) Object ID.
   (c) Bonding Box (rectangle that surrounds an object) coordinates; [x_center, y_center, width, height], where x & y are the centre of the bounding box, whereas width and height are image dimensions.
   (d) Confidence (%).
   (e) Class name.
2. An audio sensor for providing audio levels in the (dB) unit.
3. A light sensor for an overall pixel-value intensity (Brightness).
4. A REST API to request screenshots and meta-data.

5. An MQTT-broker for publishing the data produced by the previously mentioned functions, where the results might be visualized on dashboards or sent to users as alerts (Email/SMS) or stored in databases for historical data and further analysis.

However, relying solely on a pre-identify signal composed of previously mentioned functions to detect anomalies in outdoor environments, when cameras are exposed to noise such as poor illumination with no further investigation, showed an increase in false detections. The reason behind that regards the tiny algorithm integrated with the edge-camera for the detection phase. Edge-cameras, in general, are limited in resources and require lightweight strategies during model creation in order to fit edge devices and make a trade-off between performance and speed. Thus, relying only on this intuitive algorithm without further investigation yields to obtain wrong or missed detection and increases the false positive and negative detections, which is undesired in many real-world applications, especially when detecting anomalies.

Consequently, a new video analytic pipeline was proposed to overcome the system limitations and mitigate the high false positives by introducing pre-existing techniques and algorithms in Computer Vision (CV) and Deep learning (DL). Figure 1.2 depicts the multi-stages designed for the object-detection task. As mentioned, the project studied four critical development rural areas in North Wales. Therefore, depending on each use case's requirements, some developments and adjustments were applied to stages and functions—for instance, only person and vehicle instances, certain zones and motion detection. In general, the intelligent camera publishes MQTT messages to the chosen platform, *"Node-Red"*, which was tested on both; Single-Board Computer (SBC) Raspberry Pi Model 4B (RPi) and Laptop with Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz2.30 GHz and *"Windows"* as the operating system. The broker systematically streams messages, whether an object exists in the scene. Therefore, (1) messages are checked through the property *"raw_detections"* if it includes *"Person"* & *"Car"* classes allowing to reduce the burden on network communication, avoid duplicate messages and mitigate false alert notifications. (2) In case of accurate detection, a screenshot is requested using the build-in *"API"* for further processing. Otherwise, messages are neglected or dropped, and no image is requested. The availability of many prior studies for state-of-art object detection algorithms facilitates and speeds up the deployment stage, saving time going through complex tasks like the tedious training process of creating new approaches from scratch, for instance. Therefore, (3) during the project, different open-sourced pre-existing models in deep learning for the object detection task were tested and considered, such as TensorFlow, Detectron2, and PyTorch, to name a few. Afterwards, results are structured and aggregated, including brightness value and audio level, besides other information to be stored (4) in databases for message filtering and additional analysis, while only relevant and suspicious events that might be potentially considered unusual are sent to the dashboard for illustration, send notifications and alerts.

On the front-end side, scene information is received by a third party under the name *"UtterBerry"* for processing data from the camera. In addition, they provide various types of sensors installed in all use cases, such as GPS, Temperature, Motion, etc. Indeed, to reduce the inconvenient high volume of notifications delivered to the end-user of a potential anomaly, their sensor data are compared or integrated with the one generated by the camera. For example, motion sensors have the ability to acquire information about an object's movements and appearance in the scene; hence, comparing the timestamp of the API-requested image with the time of motion occurred might assist in guaranteeing and confirming anomalies (e.g. unknown vehicle entering the farm).

Thus, the collaboration of vision and non-vision sensors is beneficial for accurately identifying objects and reducing the number of unpleasant messages resulting from false warnings. For example, motion sensors have the ability to acquire information about an object's movements and appearance in the scene; hence, comparing the timestamp of the API-requested image with the time of motion occurred might assist in guaranteeing and confirming anomalies (e.g. unknown vehicle entering the farm). Thus, the collaboration of vision and non-vision sensors is beneficial for accurately identifying objects and reducing the number of unpleasant messages resulting from false warnings.

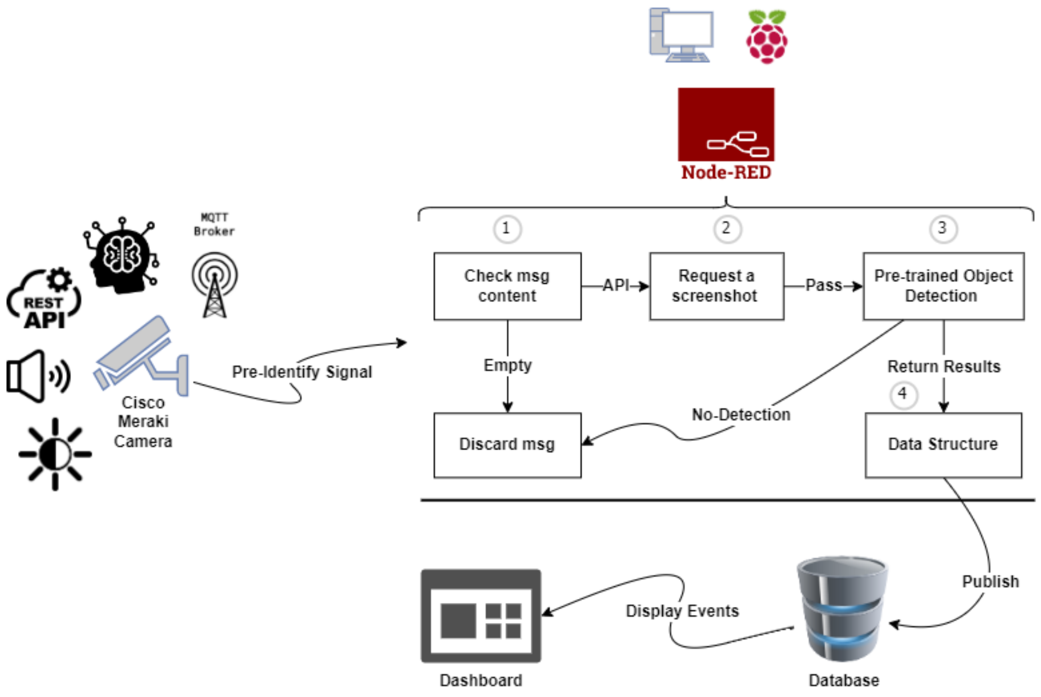


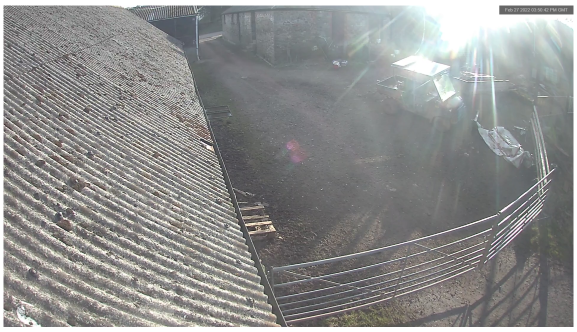
**Figure 1.2** The developed video analytic pipeline for the 5G Wales Unlocked project.

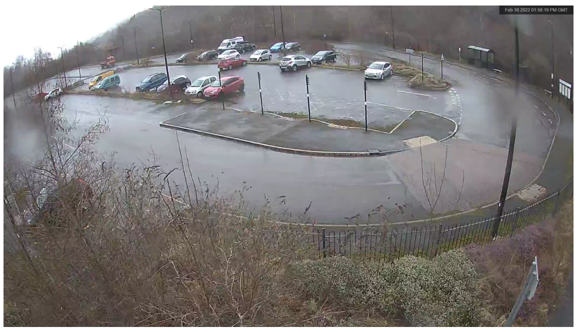
## 1.1 Motivation

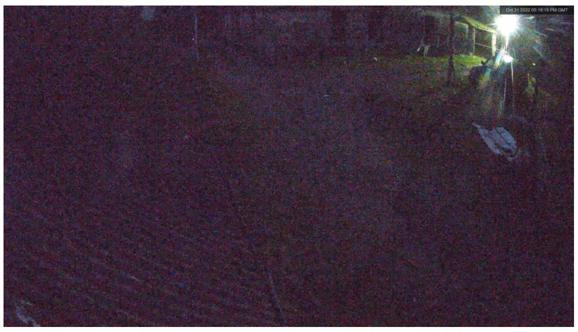
The previous section introduced a brief overview of the 5G Wales Unlocked project, showing the developed video analytics pipeline using the existing pre-trained model in Computer Vision (CV) and Deep Learning (ML) for the state-of-art object detection algorithm. Many limitations and obstacles were faced while implementing and evaluating the system. Recently, several studies relied on processing images and videos through different paradigms. For instance, some take advantage of high-computational resources on Cloud-computing when massive data is offloaded from the edge. On the other hand, who relied on Edge-computing for time-sensitivity and accelerated processing stages. Lastly, further studies focused on combining both paradigms for partial computation and reduction of network occupancy, known as *'Joint Modeling'*. Since smart cameras at the edge generate pre-identify data that might be normal or abnormal, simultaneously, the cloud environment holds the large models supported with GPUs or Multi-Core CPUs to assure the probability of an existing object in the scene. Nevertheless, the joint modelling showed scarce performance in some aspects for many reasons. For example, the MV Camera sends MQTT messages every (msec) regardless of whether a human object exists in a particular area due to model sensitivity to motion and the effect of external conditions, such as low or high lighting in specific regions or the appearance of other objects that might indicate a potential anomaly. Therefore, requesting images continuously with no actual threats puts a burden on network communication and energy consumption of computational resources. Moreover, data transmission requires a high-large bandwidth and less network usage. Due to the above limitations, this research implements the whole pipeline on resource-constrained devices at the edge for performing high-level tasks in order to overcome project limitations.

Furthermore, image quality is essential for providing trustworthy inputs for object detection tasks to produce better results. The communication channel, *the medium used to transport information from one network device to another*, is one example affecting the data source, causing a deterioration in image quality since it loses more-or-less features during the transmission phase, so images are contaminated with noise. However, the most considerable factors confronted during the system evaluation were external low-light conditions resulting from different weather conditions and images captured at night, sunset or sunrise. Figures 1.3 show a few samples of low-quality images captured during the 5G Wales Unlocked project. For example, 1.3a represents an image in midday affected by sunlighting obscuring or covering the vehicle (Mule) with bright pixels. Similarly, Figure 1.3b when rain leads to a loss of information and disability in reflecting authentic images of the situation. In addition, figures 1.3c and 1.3d for images obtained in poor illumination during the night with additional noise (e.g. salt-and-pepper) added due to transmission channel or weak connection. Thus, captured images with various circumstances (e.g. poor illuminations and weather conditions) suffer from low visibility and smoother, losing local characteristics such as edges, sharpness, texture, etc., resulting in a significant decrease in detecting instances as well as difficulty in identifying anomalies. Indeed, several factors can affect and distort obtained inputs, such as weather conditions (e.g. rain, snow, haze, etc.), blur, and low resolution. Nevertheless, various removal methods were studied and adopted to overcome these types of noises.

However, due to the vast number of studies that contributed to solving each issue by providing various strategies and concepts, it is challenging to consider and investigate all problems in the current research. As a result, the images considered for this research will only be those taken in low-light situations, such as at night and twilight, and images with insufficient illumination captured during the daytime.


**(a)** High-Light, *"Farm"*.


**(b)** Rainy Weather, *"Car Parking"*.


**(c)** Low-Light and Noise, *"Farm"*.

**(d)** Low-Light and Noise, *"Castle"*.

**Figure 1.3** Examples of the effects of diverse natural conditions on the 5G Wales Unlocked use cases.

## 1.2 Background

Several fields in Artificial Intelligence (AI) have been involved as a key-points in many applications to replace human efforts in many video surveillance systems. For example, computer Vision (CV) is one of the dominion fields to extract local and global features through processing digital images and live-streaming videos [10]. Traditionally, authorities relied on implementing Closed-Circuit Television (CCTV) to record human-objects videos in private and public areas. However, implementing these systems is highly complex due to the installation procedure, the number of cameras needed (*Depends on a particular use-case*), and cloud-based services for recording and storing captured data for more investigations.

Currently, Cameras are defined as low-cost devices and so-called vision sensors. Therefore, many sectors are inclined to replace traditional non-vision sensors with vision ones to build intelligent surveillance systems for security and other purposes. The reasons behind that camera-sensors are becoming affordable (low-cost), reliable and high-resolution, low power consumption, etc. Moreover, they can detect, track, and identify object behaviours and send alerts in an automated manner without human interaction. However, one of the ongoing and disputable challenges studied in Video Surveillance System (VSS) domains in the last decades regards the location of processing and analysing the obtained data. Indeed, many prior studies relied on processing the data on Cloud-only or Edge-only or Joint Modeling; *Partially, between Cloud and Edge paradigms* [49]. Strategies that rely on the computational resources provided by cloud computing, whether on their own or in conjunction with edge-based implementations, continue to be a cause for worry for several applications, particularly those involving the detection of abnormalities. However, streaming or transmitting acquired data to the cloud showed a heavy workload on the communication network in 2017, which accounts for 74% of the total network [7] [8]. In addition, factors like delay-sensitivity, bandwidth limitations, privacy and storage have crucial effects on increasing the computational cost of affording hardware infrastructure to maintain the number of assets recorded *images and videos* [89].

The Internet of Things (IoT) technology relies on performing all or most stages, including data acquisition, pre and post-processing, results and visualisation on resource-constrained devices on edge environments without relying on systems with powerful computational like Cloud or Fog. The implementation of data collection and processing stages at the edge illustrates effective solutions to overcome cloud-paradigm challenges (e.g. latency, privacy, etc.) by allowing multi-devices connected on the same network to perform multiple tasks in a small amount of time [85]. Thus, adapting anomaly detection systems at the edge without relying on external computational assistance (e.g. cloud, fog, etc.) plays a critical role in mitigating network burden, delay sensitivity, and faster response in actions by authorities and stakeholders in terms of making decisions. An *anomaly* is an object or data (e.g. images, videos, text or voice) that stands out and is distinct from the group.

Indeed, anomaly detection became a significant problem to consider for maintaining security and guaranteeing safety (e.g. ensuring that only people with permission enter a specific place), encouraging societies and researchers to investigate for a faster and more reliable solution. Moreover, since the data is heterogeneous and come from different sources (e.g. cameras and traditional sensors) in different formats (e.g. digital, numeric, etc.), scientists are motivated to explore and propose advanced methodologies to extract useful information and insights and detect anomalies. For instance, traditional sensors (e.g. temperature, carbon monoxide, etc.) and digital images working together may aid and verify the detection of smoke caused by a fire. Moreover, anomalies are rare and unpredictable events which require immediate intervention to prevent or control them.

In video surveillance systems, anomaly detection is the problem of detecting any abnormality in images/videos in space, time or both which deviates from normal behaviours, such as appearing in a forbidden area, children climbing walls and others. In smart cities, it plays a vital role in detecting anomalies through single or multiple cameras in real-time or by inspecting massive recorded videos maintained at high computational resource storages like databases. Since screenshots are collected or videos are recorded, a whole video inspection is possible or required occasionally for any video anomaly detection pipeline, in contrast to other systems that depend only on sensors by gathering signal values without being aware of the actual and precise condition in the field. [69].

Nowadays, cameras are dramatically increasing in dense and crowded areas to monitor human and object behaviours. In addition, security in urban and rural environments is becoming a crucial requirement, such as in the transportation, tourism and farming sectors, to name a few. Prior methods relied on human resources to continuously monitor live-streaming videos or stored videos to detect instances and make decisions about their behaviours in terms of normal and abnormal. However, it requires a massive human effort when it comes to managing and monitoring numerous screens at the same time. Therefore, developing an intelligent and automated system that can understand and identify anomalies in videos nearly to human brains is necessary for saving time and human resource efforts.

## 1.3 Aim.

This research aims to create a technique that can be employed in low-light conditions for object detection tasks on various resource-constrained devices. The proposed technique is designed to select the most effective enhancement method among multi-enhancement networks by considering the characteristics and features of the input images.

In the following section, an outline of the steps needed to achieve the research aim is delineated and explained.

## 1.4 Objectives.

1. Test existing low-light image enhancement techniques on the ExDark datasets containing images captured in indoor and outdoor environments with various light intensities.

2. Evaluate the performance and trade-offs among existing low-light image enhancement approaches with the object detection task by comparing the detection accuracy before and after enhancement.

3. Build a lightweight dynamic classifier algorithm to categorise images based on low-light image enhancement techniques capabilities for distinguishing different light intensity levels (e.g. totally dark, partially dark, foggy dark, etc.). In addition, to differentiate from the previously implemented pipeline in the 5G Wales Unlocked project, which only worked on images captured in clear or bright conditions (e.g. sunny, daytime, etc.).

4. Implement and compare the entire pipeline on the Cloud and Edge paradigms, where the edge environment consists of multiple resource-constrained *"Nodes"*, each responsible for a specific task.

5. Measure metrics related to edge devices to ensure sustainability and scalability while implementing the proposed methodologies. In addition, calculate the time needed to process a single image from source to destination and other metrics.

## 1.5 Contributions.

In summary, the main contributions of this research are as follows:

- Propose a dynamic lightweight classifier that can determine the optimal enhancement technique from a range of techniques based on the input's features and illumination level. Since low-light environments may accumulate additional noise and enhancement techniques, performance and capabilities vary depending on many factors (e.g. data used, method structure, etc.). Therefore, the classifier determines the most effective approach after the feature extraction and selection phases instead of forwarding inputs to static or specific techniques based on assumptions or criteria.

- Evaluate the efficacy of image enhancement techniques in object detection tasks involving the identification of instances such as people, cars, buses, motorcycles, etc. Specifically, consider the most effective pre-processing stage for the proposed design by analyzing the performance of various image enhancement techniques.

- To demonstrate the feasibility of our proposed system, we created a proof-of-concept by implementing it on a range of resource-constrained devices, including the Raspberry Pi and Jetson Nano Developer Kit. We then measured edge devices' processing speed and other relevant computational resource metrics.

## 1.6 Publication List

**Note** that the paper associated with this thesis is under revision. As the author, I am diligently working to refine and improve the content based on valuable feedback and insights. After completing the required viva examination, I will submit the revised paper for publication *"IEEE Internet of Things Journal"*. In the meantime, I encourage you to read the paper, which comprehensively summarises my research work, findings, and contributions. In addition, the paper offers a condensed and up-to-date representation of the project, capturing the essence of my work more effectively than the thesis. Please find the link to download the paper in the following reference.

- [111] Yaser, A., Omer, R., and Charith, P. (2023). Anomaly detection on the edge using smart cameras under low-light conditions. *Sensors*. https://drive.google.com/file/d/1VuwB-A6gmwN9AGHTZme2b1Ls4Yf1gekp/view?usp=drive_link

## 1.7 Chapter Summary.

This chapter commenced with an introduction to the 5G Wales Unlocked project, during which the purpose, objectives, results, and assessment were discussed. The primary objective was identifying anomalies and detecting instances (people and vehicles) across various use cases in urban and rural areas, such as farms, tourism (e.g. Castle), parking lots, and bus service. Nevertheless, a single scenario might call for various requirements and goals. For example, identify unknown vehicles as well as strangers in time out of work for the farm use case, whereas for tourism, individuals walk or enter unwanted areas. On the other hand, tasks focused on customer behaviours by detecting people and cars getting in/out for the bus and parking lost scenarios, respectively. Furthermore, pre-existing object detection and classification methods assisted in accomplishing the aims outlined above and have been shown encouraging results. Moreover, certain difficulties and challenges encountered during the project inspired the conduct of this research to address the deficiencies in prior studies, which are discussed in the following chapter. Also, a background on anomaly detection in video surveillance domains and the processing location is provided. In addition, the current thesis's primary aim, objectives, and contributions are described and summarized at the end of the chapter.

# Chapter 2

# Literature Review

## 2.1 Introduction

This chapter describes previous studies of anomaly detection using the edge computing paradigm, including techniques, datasets, and evaluations. Afterwards, studies related to modern deep learning and traditional machine learning approaches for scene classification to categorise scenes according to noise environment. Finally, a broad review of enhancement and removal methods for input images obtained under various degradation conditions (e.g. blur, noise, low-resolution, poor illumination, etc.) will be presented, focusing on enhancement techniques for low-light environments.

## 2.2 Anomaly Detection at the Edge Paradigm

Recently, a few studies focused on detecting anomalies at the edge paradigm [74]. In [121], the authors proposed a real-time video streaming for human detection and tracking. They focused on extracting low-level features on edge and fog environments since high-level features like human and action recognition are more suitable for performing on supported layers like Cloud-computing. A Histogram of Orientation (HOG) [9] with a Support Vector Machine (SVM) classifier [35] used to extract object representations and classify them as human and non-human on a single-board computer (*Raspberry Pi 3*). Then, features extracted by the detector are passed on to a Kernelized Correlation Filter (KCF) [30] tracker, placed at the fog stratum (*Laptop*) to estimate pedestrian's future positions and construct their trajectories. The system was evaluated with video streams from real-world surveillance, achieving a throughput of 12.2 frames per second. However, the algorithm is costly since a single input requires multi-stages through the pipeline for completion. In addition, the tracker is unable to re-identify the objects. In other words, if an object disappears from the scene and shows again, the system provides a new label for the same object.

In [1], a real-time face emotion detection was implemented on a Field-Programmable Gate Array (FPGA) device (*PYNQ-Z1 board*). A Harr-Cascade algorithm and a Local Binary Pattern (LBP) were introduced to extract face features from human images and construct a feature map. Then, features were fed to a Binary Neural Network (BNN) to train a face emotions classifier. The authors believe that face emotions classification help to detect anomalies in public transportation, such as shared cabs and taxis, to maintain passengers' safety. The classifier outputs six categories: Angry, Disgust, Fear, Sad, Happy and Surprise, where destructive emotions are labelled as abnormal and good ones as normal. The system was evaluated on the benchmark dataset JAFFE (*A dataset consists of 213 images of different facial expressions from 10 different Japanese female subjects*) [64], and on images captured by a webcam attached to the FPGA embedded-device. The overall model achieved an 81% and 75% accuracy on the public dataset and the images collected from real-world scenarios.

In [122], an Intelligent Surveillance System at the edge called (iSENSE) was proposed to illustrate the possibility of implementing machine learning algorithms at the edge. The authors designed a Lightweight Convolution Neural Network (L-CNN) based on Depth-Wise Separable Convolution [31] technique to extract human features. An SSD head was added to provide bonding boxes, probabilities and classes. Then, a multi-algorithm; Karman Filter (KF) [100], KCF (Kernelized Correlation Filter) [30] and Background Subtraction [77] were tested for tracking performance along with the proposed L-CNN. The system was tested on different Single-Board Computers (SBC), including (*A Raspberry Pi 3 Model B and Tinker board*) to improve the feature extraction stage at the edge. The evaluation was done on real-world campus video surveillance and the open image dataset; Pascal Visual

Object Classes (VOC) [20] both VOC07 and VOC12. A comparison with recent methods for the object-detection stage showed that algorithms based on Haar features are faster than the proposed L-CNN. However, L-CNN showed a low false-positive rate (FPR) and false-negative rate (FNR) with a percentage of 6.6% and 26.3%, whereas an 18.1% and 34.9% for the Haar-Cascade, respectively. Moreover, regarding accuracy and speed, L-CNN performed better than SSD-GoogleNet and MobileNet. In addition, the designed model occupied less memory on constrained devices, making it more suitable for edge environments than the standard and complex CNN approaches.

In [123], a combination of edge-cloud computing was proposed using a lightweight deep learning model at the edge. Once again, Depth-Wise Separable Convolution technique [31] was applied on the CNNs part of the Tiny-Yolo and MobileNetV2-SSD to reduce the model complexity at the constrained device (*NVIDIA Jetson TX*). At the same time, the centralized cloud is supported with a *NVIDIA Jetson TX* graphics card hosting a large model (YoloV3) to validate the detection phase. The research focused on utilizing fewer network resources and providing a faster response to the edge by proposing the joint model. The evaluation was tested on a self-deployed dataset, collected and labelled for detecting people wearing/non-wearing helmets at a construction site. The video data collected by a camera attached to the edge device are sent to the edge note for the pre-detection stage using a lightweight model. Only clips with people not wearing helmets are extracted, encoded, and transmitted through the User Datagram Protocol (UDP) to the cloud to validate edge detection performance. The results are then sent back to the edge for other purposes. The system showed a speed of 16 FPS using the MobieNetV2-SSD with an accuracy of 0.68 mAP. In contrast, Tiny-Yolo achieved a better accuracy of 0.73 mAP with a speed of 12 FPS.

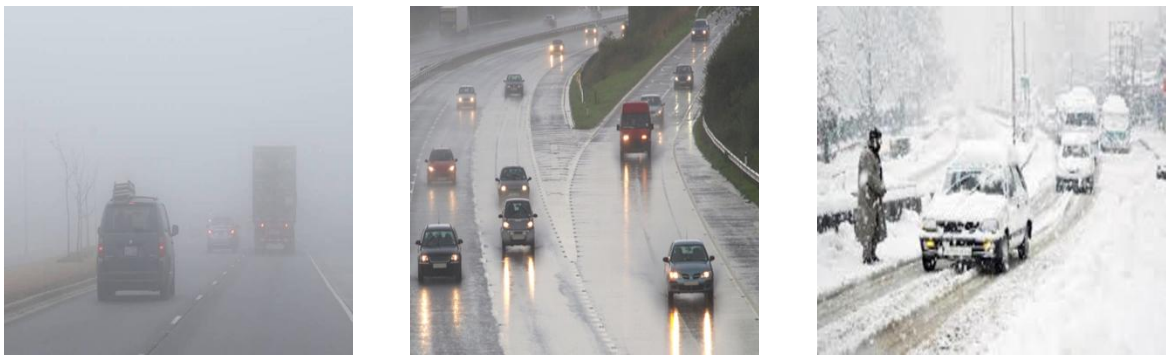
The released YOLOv5 model was exploited in recent studies since it is suitable for real-time video feeds, providing high accuracy and speed, outperforming recent one-stages and two-stages models [125]. The authors in [66] proposed real-time video analytics on the edge device supported with a GPU (*NVIDIA Jeston TX2*) for detecting people in forbidden areas by combining YOLOv5s (*s: for small version; suitable for constrain-devices*) [37] with the DeepSORT tracker [103]. The detection phase is done by YOLO producing critical information (*Bounding Box, Class Name and Probability*). The tracker uses these detections to track and assign a unique ID to each object. The system was tested and evaluated on the edge device using a GPS receiver, different camera sensors (RGB and Thermal), different positions (Outdoor Building fixed-position and Mounted on Vehicles) and different light conditions (Daytime and Night). Moreover, they experimented with the system several times with different factors to measure the accuracy performance. For instance, only with the detector (YOLOv5) or aggregated with the tracker (DeepSORT) day, night mounted on cars or fixed-camera outside a building. In their research, an open-source edge platform called DECIoT was introduced to collect data; (Objects information and Metadata, like Timestamp, GPS longitude and latitude and File Path of relevant images stored at the FTP server) through an MQTT-Broker (Mosquito). In addition, the platform can send alerts and notifications to the middle-ware *(Apache-Kafka)* for sharing information with other platforms and services. The system achieved an overall accuracy with *(F1_Score = 86.4)* for the object detection phase. However, the tracking phase performs poorly when an object is occluded, yielding to assigning a new ID to the same object, which is not desirable in real-world scenarios.

**Table 2.1** Summary of related works in anomaly detection on the resource-constrained devices at the edge.

| Paper Research | Task | Method | Location | Class | Device | Evaluation | Environment /Weather Condition |
|---|---|---|---|---|---|---|---|
| [121] | Detection &Tracking | **Edge:** HOG+SVM **Fog:** KCF | Edge-Fog | Pedestrian | Raspberry Pi3 and Laptop | Video-streaming from real-world surveillance | Outdoor /Clear |
| [1] | Face detection | Haar-Cascade &LBP+BNN | Edge | Pedestrian | PYNQ-Z1 board | JAFFE Dataset and images from real-world | Still images /Clear |
| [122] | Detection & Tracking | L-CNN based on Depth-Wise Separable Convolution + SSD with tracking algorithm (KF, KCF and BS) | Edge | Pedestrian | Raspberry Pi3 | Video-streaming form real world and VOC datasets | Outdoor and Indoor /Clear |
| [123] | Detection | **Edge:** L-CNN based on Depth-Wise Separable Convolution + Tiny-Yolo and MobileNetV2-SDD **Cloud:** Yolov3 | Edge-Cloud | Pedestrian and Helmet | NVIDIA Jetson TX | Self-Images and frames extracted from video | Outdoor and Indoor /Clear |
| [66] | Detection & Tracking | YoloV5+DeepSORT | Edge | Pedestrian | NVIDIA Jeston TX2 | Video-streaming form real-world | Outdoor /Clear |

## 2.3 Weather Classification

Globally, accidents occur daily owing to inattentive drivers, inadequate infrastructure and weather conditions in many terrains and sectors, such as train collisions, ships crash and even aviation occurrences. Therefore, weather conditions are one of the crucial aspects examined recently in the computer vision era to solve problems which vary in context. For instance, whether recognition might help prevent and mitigate vehicle mishaps on the road, sends alerts to drivers for slowing down speed, and is essential for approaches in driver assistance [43], self-driving vehicles [32] and detecting vehicles and pedestrians in-the-wild environments [94]. In place of the vast demand for human resources and costly sensors to acquire meteorological information, researchers applied visual computing approaches to predict and classify categories from images Figure 2.1 using a single camera. Weather recognition has been investigated massively, utilising multiple computer vision and deep learning techniques in previous years. Therefore, the following sections briefly review related studies conducted in classifying weather conditions.



**(a)** Foggy/Hazy. **(b)** Rainy. **(c)** Snowy.

**Figure 2.1** Samples of weather conditions categories.

### 2.3.1 Traditional Machine Learning.

When mentioning traditional machine learning, "feature engineering" or "hand-crafted features" often comes up. These terms refer to the process in which the features that will ultimately be used in the algorithm are studied and picked by user choice. In contrast, modern techniques, such as convolutional neural networks, carry out the feature selection step in an automated fashion [33].

In [50], a combination of Support Vector Machine (SVM) and Decision Tree (DT) was proposed by [91] by extracting certain features to recognise weather phenomena. The system relied on traditional feature engineering to extract image representations that might belong to particular weather. These features are; power spectrum slop, contrast, noise and saturation. The trained models were tested on Wild Image Dataset [71], and self-images were captured. Because multiple forms of weather might occur concurrently, the system exhibited a significant false-positive rate when detecting the rain class. As a result, images need to be correctly labelled with their actual label. Similarly, [90] relied on the same factors besides the inflection point fed to a K-NN classifier to recognise weather categories such as sunny, fog, snowy and rain. However, the classifier works only for fixed scenes; traffic scenarios and was tested on only two images captured from a real-world scenario.

The authors of [61] exploited weather signals (e.g. sky, shadow, reflection, contrast and haze) as a feature vector to categorise only two categories (Sunny and Cloudy). By assuming that each cluster has distinct characteristics and corresponds to one of the classes, a learning-based approach was developed to cluster the extracted image features into distinct groups, then be classified into a specific weather class. A dataset of 10,000 was collected and divided for training and evaluation purposes. Unfortunately, some images lacked one of the identified cues (e.g. sky), which hindered the system's ability to classify the correct label.

Moreover, [119] relies on combining numerous local and global, such as Sky, Shadow, Reflection, Contrast and Haze and Contrast and Saturation, respectively. Finally, the authors employed Multiple Kernel Learning [4] to train a robust classifier for predicting the weather. The multi-class Weather Image (MWI) dataset contains sunny, hazy, snowy and rainy images used for training and testing. As a result of the sparse features extracted, the model attained an overall accuracy of 0.59%.

### 2.3.2 Deep Learning

Recent studies used GoogleNet through pre-training and fine-tuning, RestNet-15 based on RestNet-50, MeteCNN, RestNet-18, for weather classification [129][105][106][2] respectively. In [51], images were categorised as sunny or cloudy using a CNN-based Multi-task classification. The CNN extracted features to predict the type of weather and shared low-layer features for semantic segmentation to understand weather cues such as sky (Blue or Dark grey) and objects shadows, which were drawn manually with a bounding box. The method was evaluated with 10K images (5K per class) with an overall classification accuracy of 60.8%.

Furthermore, in [120], a CNN and LSTM-based Recurrent Neural Network to recognise weather was proposed. The reason behind combining two neural networks in case more than one weather condition, such as rain and fog, might coincide. The authors of [34] suggested the Weather-Net architecture for weather recognition. Instead of depending on a single class to represent the type of weather, they trained a 4-Convolution Neural Network with RestNet-50, where each node has a distinct function. A CNN for category classification into Dawn/Dusk or Day/Night referred to as *NightNet*. Then, under *GlareNet*, a binary classifier for Glare (Natural-Light, such as sunlight) or Non-Glare (Artificial, such as vehicle lights). In addition, *FogNet* is a binary classifier for fog or no fog, and *PreciptationNet* is a multi-class CNN for clear, rainy, and snowy conditions. They believe that weather conditions alone cannot provide a clearer scenario picture. Therefore, adding more labels relating to visual information could result in a more robust and accurate system in real-world circumstances. The testing was conducted using a self-collected dataset to utilise different metrics throughout the evaluation phase. The accuracy of the framework ranged from 91.6% to 95.6% among the four models.

A fusion approach was presented in [52] for integrating specific features extracted manually, such as Brightness, Contrast and etcetera, with features extracted by CNN to recognise whether classes (e.g. sunny, cloudy, foggy, rainy and snowy). The model was assessed on dataset inspired by [26] using different CNNs {*VGGNet-16, VGGNet-19, RestNet-50 and RestNet-101*}. Many strategies were tested with (1) CNN as a standalone, (2) CNN and one specific feature, and (3) CNN and all specific features. The experiments showed that CNN features paired with feature engineering offered better results than depending solely on CNN, which is insufficient for categorising weather images adequately. Moreover, they increased the recognition accuracy by applying methods for weighting features, which allow retrieving relevant features from a trained neural network based on the strength of the weights [113].

**Table 2.2** Summary of related works for weather classification using deep learning and traditional machine learning approaches.

| Paper Research | Feature Extraction | Classifier | Weather Classes | Evaluation |
|---|---|---|---|---|
| [50] | **Global:** Power Spectrum Slop, Contrast, Noise and Saturation | SVM + Decision Tree | Clear, Rain, Fog& Overcast | Wild Image Dataset |
| [90] | **Global:** Inflection Point, Edge, Contrast, Saturation and Noise | K-NN Classifier | Sunny, Fog, Snowy& Rain | Images from real-world scenes |
| [61] | **Weather Cues:** Sky, Shadow, Reflection, Contrast and Haze | Learning-based approach | Sunny & Cloudy | Weather Image Dataset: 10K images self-collected |
| [119] | **Local:** Sky, Shadow, Reflection, Contrast and Haze **Global:** Contrast and Saturation | Multiple Kernel Learning | Sunny, Rainy, Snowy& Hazy | Multi-class Weather Image (MWI) dataset |
| [129][105] [106] [2] | **CNNs:** GoogleNet, RestNet-15, MeteCNN & RestNet-18 | Fully-Connected Convolution Network | Sunny & Cloudy | N/A |
| [51] | **Multi-CNN:** for weather cues segmentation and weather recognition. | Fully-Connected Convolution Network | Sunny & Cloudy | 10k-Images dataset |
| [120] | **4-CNNs:** RestNet-50 & LSTM-based (Recurrent Neural Network) | Fully-Connected Convolution Network | Day, Night, Dusk/Dawn, Glare, No-Glare,Fog, No-Fog, Clear, Rainy & Snowy | Self-images |
| [52] | **Global:** Brightness & Contrast **CNNs:** VGGNet-16, VGGNet-19,. RestNet-50 & RestNet-101 | Fully-Connected Convolution Network | Sunny,Cloudy,Foggy, Rainy & Snowy | Dataset from [26] |

## 2.4 Image Restoration

In applications for smart cities and wild environments, the efficiency of recognising people and vehicles is becoming indispensable. The evolution of sensors and high-computing resources such as GPU enables researchers to devote considerable time and energy to understanding and to investigate new techniques such as video surveillance systems, traffic monitoring, and autonomous driving, which has become a trend. In video surveillance systems, for instance, precise detection of objects (e.g., people, vehicles, animals, etc.) is required in real-time for decision-making and environmental safety. However, the efficiency of sensor heterogeneity (e.g., cameras, light and audio sensors) is crucial for achieving the desired results when detecting various objects. For instance, a camera sensor's light absorption, reflection, calibration, and field of vision can vary. In addition to its inherent qualities, the taken image is also affected by external circumstances, such as weather, motion blur, background change, camera shake, and low-light situations, which result in a low-quality image during the acquisition stage [109].

In contrast, indoor environments are less affected by noise because the parameters mentioned above may be adjusted and configured by users [28]. Poor visibility resulting in incorrect detection causes accidents and traffic, for instance, in the autonomous vehicles domains. To gain clear visibility and extract valuable feature representations, developing or utilising existing image enhancement, image restoration, and noise reduction approaches is necessary. Therefore, delivering a clean image is necessary for precise detection in many systems. Since this research focuses on detecting instances in noisy environments, pre-processing steps are required before passing the input source to the detection system. Therefore, this section gives basic information about image restoration methods, including denoising, dehazing, super-resolution and low-light enhancement methods for a single image. Images captured by sensors devices such as IoT cameras and smart mobiles typically are sensitive to different internal and external factors resulting in many kinds of degradation. For example, motion blurring occurs due to camera shaking, moving objects, haze, rain and depth variations in video surveillance. Generally, image restoration is restoring the image's features, clear and sharp, from the degraded inputs.

The captured image during the accusation stage might expose to different types of noise, such as weather conditions, blur and low resolution, to name a few. Image restoration is divided into many restoration methods for solving several degraded challenges. Indeed, this section focuses on restoring images captured with insufficient illumination due to many conditions, such as nighttime, rainy or cloudy weather, to improve the detection stage. Moreover, additional techniques belonging to restoration approaches for better visual representation and feature restoration are presented.

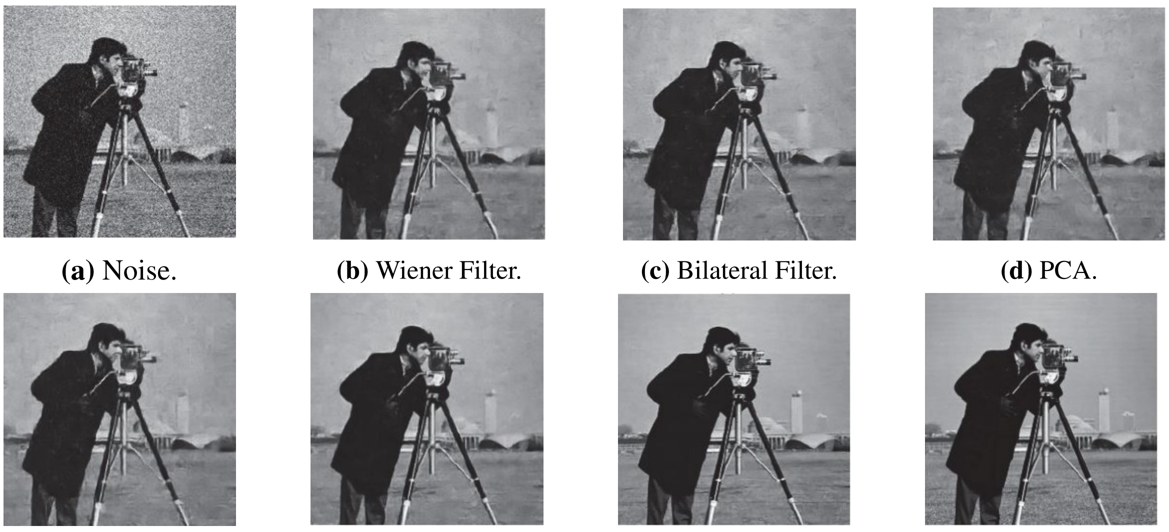
### 2.4.1 Image Denoising

In recent decades, removing image noise has become necessary for obtaining a high-quality output to optimise other processes such as object segmentation, detection and tracking in many image processing pipelines [107]. As the name suggests, image denoising is the process in which the degraded image is translated into a clean one to be close to the ground truth or original image. The idea is to obtain the clear image "$I$" from the noisy image "$O$" by suppressing the noise "$n$", called Additive White Gaussian Noise (AWGN) with variance $\hat{\sigma}^2$ [92], as is described in the Equation 2.1. Compared to Multiplicative Noise, additive occurs during transmission, compression and acquisition stages by adding undesired pixels to the original ones.

In contrast, Multiplicative manipulates the original pixels in such a way making it more challenging to be removed. Traditional approaches [101][82][40] focus on feature extraction and selection methods to overcome noise in image processing. However, these methods have fixed parameters which require a tweak in their hyper-parameters to work for a specific task. Moreover, they are unsuitable for blind image denoising, requiring a prior/reference image to capture low-level image statistics. In addition, optimisation is needed for high performance and satisfactory results leading to a dramatic increase in computational cost [114].

$$O = I + n \tag{2.1}$$

Indeed, to overcome the traditional challenges, many deep learning networks were studied based on CNNs and generative adversarial networks GANs [93]. One of the recent achievements was DnCNN by [114] using a Residual Network known as the solver of gradient vanishing problem for deeper neural networks [29]. The authors mentioned in their paper that DnCNN could handle different tasks for image processing besides denoising, like super-resolution and JPEG image blocking. In Figure 2.2, experimentation done by [24] on MATLAB using Deep Learning Toolbox version R2919a shows the results of different denoising techniques. The authors compared some traditional techniques of denoising based on spatial or transform methods versus the pre-trained model DnCNN. As is depicted, the CNN output in 2.2g is superior to conventional methods in preserving features such as texture and edges besides noise suppression, obtaining an output closer to the original image in Figure 2.2h.



**(a)** Noise. **(b)** Wiener Filter. **(c)** Bilateral Filter. **(d)** PCA.

**(e)** Wavelet. **(f)** Total Variation. **(g)** CNN. **(h)** Original.

**Figure 2.2** Denoised Image methods.

## 2.4.2 Image Dehazing

The goal of image dehazing is to sharpen and restore a haze image. Haze or fog mitigates the quality of the image in outdoor scenes by manipulating image colour and reducing the contrast between the foreground and background, smoothing the region of interests and objects in the captured image. In image processing, obtaining a free-haze image becomes a vital and inevitable area [84]. Because the light reflected from the surface comes from different sources, genuine or artificial, with hazy weather, the light is scattered and distributed in the atmosphere before reaching the camera. Therefore, foggy conditions affect the image in which the contrast is reduced, and surface colours become dim, making instances hard to be detected.

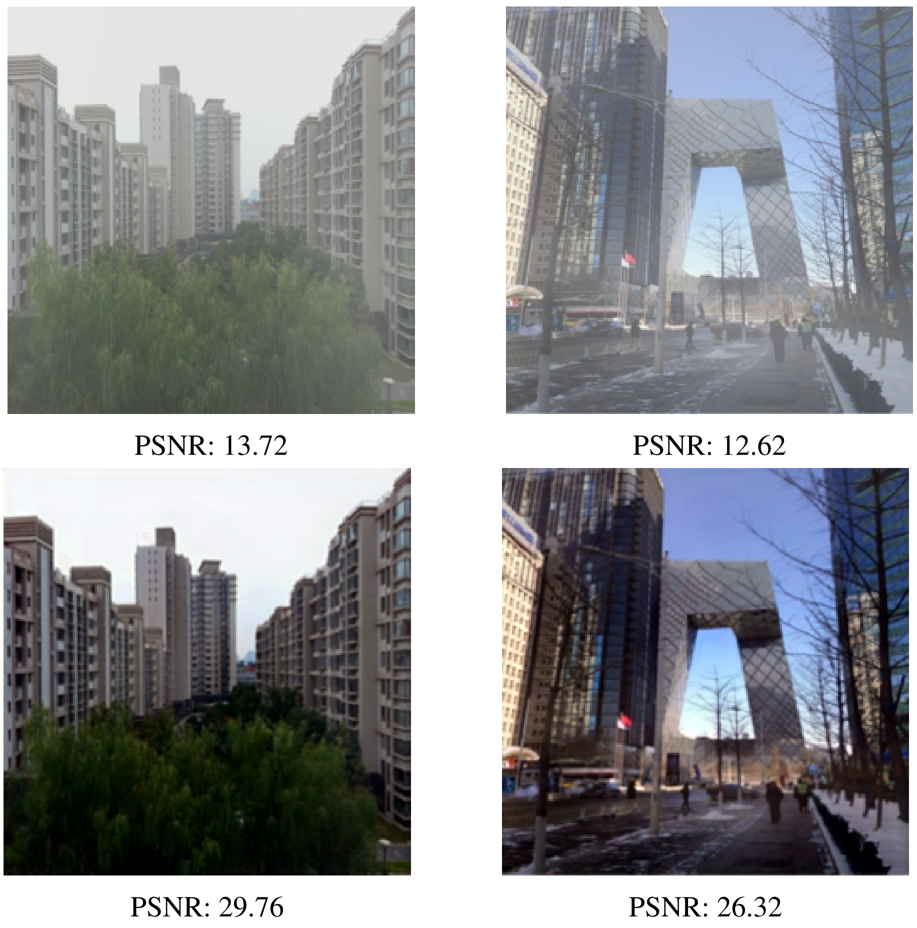


**Figure 2.3** First raw haze inputs, second raw de-hazed outputs [6].

Figure 2.3 illustrates an experiment implementing the haze-free approach by [6], where a smoothed dilation technique was used besides a multi-sub network to fuse different features from different levels. Peak-to-Signal-Noise Ratio (PSNR) is calculated for haze and de-haze images, which shows an improvement in visual, machine perceptual and image quality. Dark Channel Prior (DCP) was implemented by [67] for dehazing under the assumption that the most local patches in haze-free images consist of pixels with low intensity in at least one colour channel, which is a critical benchmark among dehazing approaches. A physics model was presented in [72] to describe the appearances of scenes in uniform haze images. Polarization was used in multiple haze images in different angles [87]. With the prosperity of data-driven approaches, many deep learning dehazing techniques were studied and proposed. In [45], the authors proposed a reformulated atmospheric scattering model, so-called all-in-one, using a lightweight CNN. In [78], an Enhanced Pix2pix Dehazing Network (EPDN) was proposed using a generative adversarial network (GAN) for dehazing using a discriminator to guide the generator to create a pseudo-realistic image on a coarse scale, whereas the enhancer following the generator was required to generate a realistic haze-free image on the fine scale.

### 2.4.3 Image Super-Resolution

Super-Resolution (SR) is another type of image restoration and inverse problem for recovering high-resolution (HR) inputs from low-resolution (LR) ones. LR occurs due to down-sampling *(e.g. bicubic, average down-sampling...etc.)* or applying down-sampling kernels on images besides to other reasons such as camera modality, calibration and lenses leading to capture a degraded image. Figure 2.4 illustrates a super-resolution approach based on a single image's simple convolution neural network (CNN) architecture.

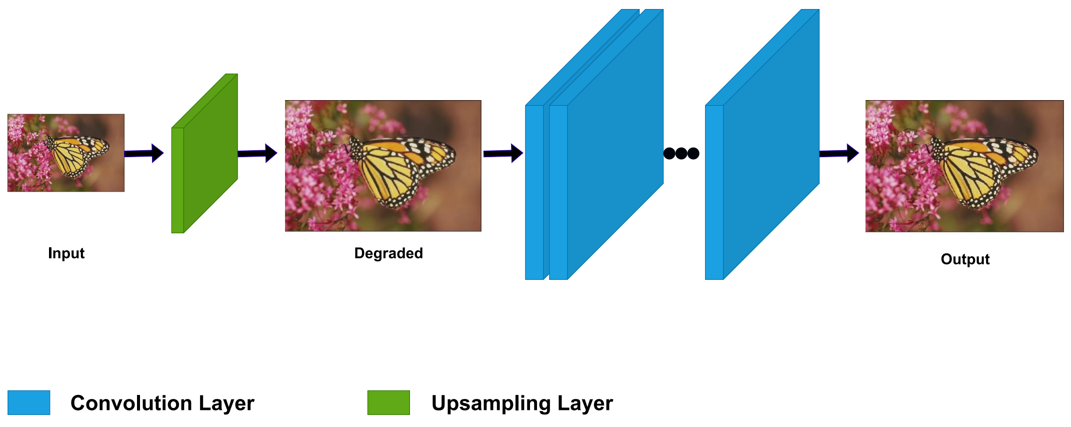


**Figure 2.4** Super-resolution for single image [19].

In [19], a simple CNN was proposed to solve image super-resolution. Whereas [53] improved the performance of previous models by removing unnecessary modules or layers in conventional residual networks with EDSR. Moreover, [117] proceed with residual networks by adopting residual in residual (RIR) and residual channel attention networks (RCAN), solving the problem that deeper networks for image SR are difficult to train.

### 2.4.4 Low-Light Image Enhancement

The objective of low-light image enhancement is to augment the recognition and visibility of an image that is captured in low-light surroundings. This particular task falls within the restoration category, wherein a deteriorated input is handled to obtain an improved output similar to the original, accurate image [60].

Nowadays, the prevalence of sensors and mobile devices has led to a significant rise in the number of images acquired in various settings. Furthermore, the dimensions of the camera aperture, resolution and other features have a crucial impact on the resulting image. Previous research has employed Learning-based techniques in this field, particularly Deep Learning (DL), which has surpassed traditional methods in several respects. These include superior precision, resilience and a reduced number of optimization procedures, leading to considerable speed and model convergence acceleration. In 2017, the initial implementation of Auto-Encoders (AE) was introduced by [60] for managing images in low-light surroundings. This technique addressed several difficulties and surpassed traditional methods. Subsequently, various learning-based approaches, network architectures, loss functions and training datasets were proposed, resulting in models with diverse performances concerning factors like image quality, speed, model size, and others. On the other hand, conventional methods are mainly founded on Histogram Equalization-based (HE) and Retinex-based methods. Retinex techniques aim to split an input image into illumination and reflection components, with the reflection component assumed to be the enhanced outcome or used for further examination based on priors and hand-crafted regularizations, which are difficult to adjust.

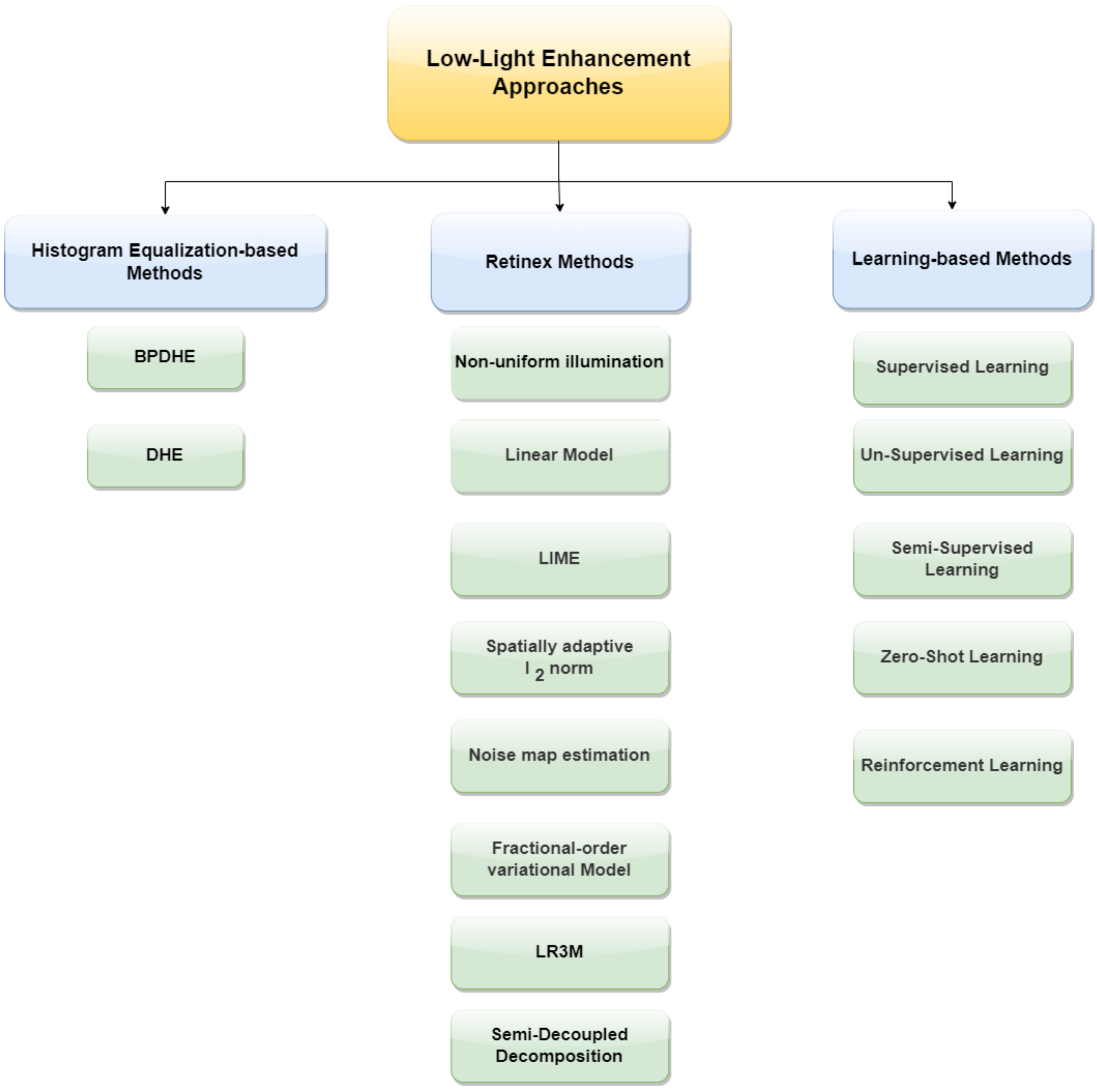


**Figure 2.5** Low-light image and video enhancement methods [47].

The diagram 2.5 illustrates the most popular approaches and their methods studied recently in the area of low-light enhancement for a single image and video [47]. Histogram Equalizer (HE) is a technique to adjust image intensity, improving image contrast to distinguish foregrounds from the background by stretching the image's dynamic range. Retinex methods are based on illumination and reflection components; each is manoeuvred separately to obtain the enhanced image. Deep learning methods are known for learning from a large volume of data through tedious training on high computational devices like *GPUs*. Generally, restoration methods rely on paired data (Degrade vs Ground Truth) for training a particular model. Deep learning models vary in architectures and concepts (e.g. Auto-Encoder (AE), Recurrent Neural Network (RNN), etc.). However, they all aim to map features from inputs to obtain results close to the ground truth image, known as *"Supervised-Learning"* task. On the other hand, the availability of paired data is cumbersome in some circumstances since obtaining images with poor and normal illumination of the same visual scene is daunting.

Moreover, training a deep learning model on paired data might yield over-fitting and produce a non-generic model, which cannot deal with various illumination levels as well as inputs that might be accumulated with additional types of noise. Therefore, few studies introduced *"Unsupervised-Learning"* to enhance low-light images without the need for ground truth data for both mapping features and evaluation [36]. In addition, *"Zero-Reference Learning"* is one of the most needed and efficient approaches nowadays because it directly examines and comprehends degraded image features to improve dark pixels into bright ones without needing ground truth (as paired data) for mapping and comparison. It is worth mentioning that this thesis exploited pre-existing deep learning and computer vision models to restore degraded images in low-light environments without considering the learning type, focusing more on efficiency and produced results.

Indeed, different factors can cause poor illumination, including low brightness, contrast, dynamic range, colour distortion, and significant noise. Such factors can have a negative impact on both human vision and various machine vision systems. In the deep learning era (2017-2021), various learning-based models were developed and analysed, and Table 2.3 presents some of these models. Learning-based techniques have significantly improved compared to traditional methods like Retinex versions, Histogram Equalizer, frequency-domain methods, and defogging models. In particular, they have overcome traditional methods of processing speed, model generalisation, accuracy, and memory usage, all of which are essential in domains like video surveillance and self-driving systems [97].

**Table 2.3** **(SL)** Supervised Learning, **(USL)** Un-Supervised Learning, **(SSL)** Semi-Supervised Learning, **(ZSL)** Zero-Shot Learning.

| **Method / Learning** | **SL** | **USL** | **SSL** | **ZSL** |
|---|---|---|---|---|
| LLNet [60] | ✓ | | | |
| LightenNet [48] | ✓ | | | |
| RetinexNet [99] | ✓ | | | |
| MBLLEN [63] | ✓ | | | |
| Chen et al [5] | ✓ | | | |
| DeepUPE [96] | ✓ | | | |
| KinD [118] | ✓ | | | |
| KinD++ [116] | ✓ | | | |
| EnlightenGAN* [36] | | ✓ | | |
| ExCNet [115] | | | | ✓ |
| Zero-DCE [27] | | | | ✓ |
| DRBN [110] | | | ✓ | |
| Xu et al. [108] | ✓ | | | |
| TBEFN [62] | ✓ | | | |
| RRDNet [127] | | | | ✓ |
| DSLR [54] | ✓ | | | |
| Zero-DCE++* [46] | | | | ✓ |
| RUAS* [58] | ✓ | | | |
| Retinex-DIP [124] | | | | ✓ |
| UTVNet [126] | ✓ | | | |
| CSDNet [65] | ✓ | | | |
| CSDGAN [65] | | ✓ | | |
| LiteCSDNet-LOL* [65] | ✓ | | | |
| LiteCSDNet-UPE* [65] | ✓ | | | |
| SLiteCSDNet-LOL* [65] | ✓ | | | |
| SLiteCSDNet-UPE* [65] | ✓ | | | |
| RED-RT* [44] | | ✓ | | |

The pre-existing models analyzed for low-light image enhancement through learning-based tasks are presented in Table 2.3. However, it is noteworthy that the prevalent techniques depend on supervised learning, which involves feature mapping and metrics evaluation using paired images of degraded and ground truth. In addition, the availability of various theories, such as the retinex theory, enables users to produce synthetic images that resemble those captured in low-light settings. This data can be used to devise new methodologies for addressing the challenges posed by different low-light levels. Nonetheless, approaches based on synthetic data exhibit poor performance in real-world situations, leading to an increase in false positives during detecting anomalies (e.g. people, vehicles, etc.), particularly at night.

Consequently, researchers are exploring a new direction for addressing the low-light issue that does not rely on prior knowledge derived from reference images with normal illumination, such as zero-reference and unsupervised learning. In the study by [63], a sub-networks enhancement was suggested by extracting feature representation from low-light and enhanced images for subsequent feature fusion. Conversely, [36] proposed an unsupervised learning model that employed generative adversarial learning (GAN) techniques. The UNet [79] architecture has been chosen as the *generator* part of the network. In general, *generator* and *discriminator* in GAN networks fight each other until the *discriminator* gives up on recognizing images created by the *generator* as fool images, producing a realistic output similar to the original ones. In contrast to most known learning approaches, [46] proposed a model without using paired images based on curve methods to create a lightweight network. In addition, [110] proposed a model for recovering linear band representation of an enhanced image under supervised learning, then obtaining an improved one by recomposing the given bands via a learnable linear transformation based on unsupervised adversarial learning. The attached star *(*)* to some model names in 2.3 stands for *"Lightweight"* models. Alternatively, models highlighted with *(Blue)* colour refer to additional pre-existing and recent models that have not been explored in [47] and recent studies.

Consequently, in this thesis, these models are studied in terms of their performance as a pre-processing stage for improving poor illumination before applying high-level tasks and further analysis, such as object recognition and segmentation. In most cases, lightweight models outperform their normal/non-lightweight counterparts in some aspects, including processing speed, model size, and others. Furthermore, lightweight models will receive the most attention because this thesis aims to implement the whole pipeline on constrained devices. In the next and last section, a brief discussion is conducted to wrap up all critical points of this research concerning the aim, objectives and focus. In addition, a graphical representation of the proposed system architecture is presented and detailed at the begging of the following Chapter (3).

## 2.5 Discussion

This thesis aims to identify the instances (e.g. people & vehicles, referred to as *"Anomalies"*) in rural environments on resource-constrained devices when these images are captured under low light and poor visibility. As mentioned in the previous sections, noise is challenging and inevitable due to darkness, weather conditions, camera calibration, motion and other factors in many image processing domains. Therefore, noise accumulates and distributes arbitrarily across the captured data source by adding low/high frequencies, which come in different shapes and intensities, making it a challenging problem to solve. As previously stated at the beginning of Chapter (1), this study is based on the 5G Wales Unlocked aiming to exploit computer vision and machine learning existing approaches to overcome several gaps or limitations in diverse smart cities sectors (e.g. farms, tourism, transportation services, etc.) from images captured by video surveillance systems. It's worth noting that the scenarios of the 5G project primarily contributed to identifying the problem state in this research, rather than serving as a source of data. This was due to privacy concerns, inactivity in most of the field, and other factors.

The following key points provide a reminder and a brief description of each use case studied during the project for a better understanding of each scenario's requirements:

- **Farm.**
  - **Location:** Monmouthshire, North of Wales.
  - **Num. of Camera(s):** 4.
  - **Camera(s) Brand:** Meraki Camera MV72X (by Cisco)
  - **Environment Type:** Outdoor/Private.

  - **Task:** Maintaining safety and preventing theft of animals and equipment are the goals of this use case. The primary objective is identifying suspicious individuals and vehicles except for farm vehicles (e.g., a tractor, mule, etc.), owners and co-workers.

- **Raglan Castle.**
  - **Location:** Monmouthshire, North of Wales.
  - **Num. of Camera(s):** 3.
  - **Camera(s) Brand:** Meraki Camera MV72X (by Cisco).
  - **Environment Type:** Outdoor/Public.
  - **Task:** To detect individuals in forbidden areas and prevent children from climbing walls.

- **Transportation Services.**
  - **Location:** Bus (Camera mounted inside a public bus).
  - **Num. of Camera(s):** 3.
  - **Camera(s) Brand:** EdgeVis Cam (by Digital Barriers).
  - **Environment Type:** Indoor/Public (But exposed to external conditions).
  - **Task:** To count the number of individuals getting in and out from a public bus and determining available space and occupied seats.

- **Parking Lot.**
  - **Location:** Blaenau Gwent in South-East of Wales.
  - **Num. of Camera(s):** 3.
  - **Camera(s) Brand:** Meraki Camera MV72X (by Cisco).
  - **Environment Type:** Outdoor/Public.

- **Task:** To determine free and occupied parking spaces by vehicles. In addition, to detect individuals at the Bus stop.

Several problems were encountered throughout the 5G Wales Unlocked, from the source (camera), through the medium (network transmission channel), ending with processing and complex analysis at the destination (cloud or edge). When receiving inputs from the Meraki camera, some obstacles that might arise include motion blur caused by moving objects, rainfall on the camera lens due to rain, and insufficient light, especially at night. In addition, cloudy and rainy weather can also provide problems in obtaining a blurry and degraded image. As a result, this thesis focuses solely on the most affected and avoidable type of noise; poor illumination or low-light conditions, which were encountered and suffered during the project as well as other smart city scenarios. The following key points justify studying the low-light issue.

- Generally, researchers and prior studies have investigated these challenges intensively and separately by proposing and designing diverse methodologies to tackle each problem. Therefore, it is impossible and challenging to study and cover all in one study.

- Poor illumination was the most notable and affected factor when implementing high-level tasks. Indeed, the detection task struggles to find and localize objects in images with insufficient light or dark ones due to the night or lack of artificial lights (e.g. street lights, lighthouse, etc.) across most use cases. For example, while testing the implemented method described in Figure 1.2 specifically for the Farm use case, many false alerts indicating the detection of a person or car were received during the night, dawn, and dusk of the day. However, the dominant dark pixels in the requested screenshot make it impossible for machines and humans to examine and verify the camera detection output.

- The sensitive and tiny algorithm integrated with the Meraki camera provides a pre-single continuously due to low-driven tasks (e.g. motion, low/high light and audio, etc.), leading to a significant rise in false alarms result in an unscalable system, as well as annoying notifications and false messages delivered to the end-user (e.g. stakeholders, authorities, etc.).

The studies mentioned in section 2.2 vary in the system performance in terms of accuracy, latency, computational resources and location of processing at edge or cloud environments. Indeed, some approaches focused on improving the overall accuracy by manipulating the model architecture [123] and others by leveraging the traditional feature engineering techniques for the object detection phase [121]. However, traditional machine learning algorithms perform weakly compared to deep learning models that extract rich information without human effort. Moreover, generating a new model requires collecting massive data for the tedious training phase. In addition, some techniques are needed to tweak the model hyper-parameters to obtain an optimised model. Studies like [1][125] relied on exploiting optimised edge devices for faster processing in a low-time response by neglecting the high cost of these devices. The proposed approaches studied different aspects to improve the system's functionality at the edge with a trade-off between accuracy, speed of analysis and obtaining higher time response. However, the focus was on proposing a lightweight approach to fit the edge device with a faster inference.

Nevertheless, the approaches in the literature were tested and evaluated on data captured from indoor environments or taken under clear weather and normal light-setting for the outdoor scenarios, without considering noise and poor illumination resulting from weather, depth variation and darkness or inadequate lighting. Noise affects by adding corrupted pixels to the collected videos and images, yielding low-quality input data. Therefore, high-level tasks like object detection, segmentation and classification struggle to extract high-level features, which might increase the system's False Positive (FP) or True Negative (TN) of recognising or classifying objects. The reason behind this, may these models were trained on a large volume of images [56] captured under clear weather with high-resolution cameras, leading to mitigation in the model performance on noisy and low-quality inputs. Thus, to fill the above gap, this research focuses on identifying anomalies on edge devices in degraded images and scenes (shot in low-light environments) for the object detection task rather than solely considering bright scenarios (e.g. daytime, sunny, etc.) investigated by mentioned studies in the literature. Moreover, to accomplish this goal, a number of the existing approaches (see Table 2.3) are used and considered as a pre-processing step. This stage aims to enhance the visibility of low-light images to produce brighter ones for better performance and output during the detection stage. As noted in Section 2.2, numerous research depends on evaluating input characteristics to decide whether to analyse data locally, partially or transmit it to centralised paradigms for exploiting powerful computing capabilities. It is worth mentioning that partial and centralised systems require establishing a connection between both ends, exchanging or returning data for visualisation or publishing alerts for subsequent actions. These stages, however, result in excessive latency, which is undesirable for anomaly detection domains.

Moreover, more specifically, strategies that rely on dividing tasks between the edge environment and the cloud environment require offloading a portion of the data through the transmission channel. This results in a variety of unintended consequences, including, but not limited to, delays (e.g. caused by limited bandwidth provided or channel congestion), cyber-attacks (e.g. man-in-the-middle), and privacy violations (e.g. images with people's faces and car licence plates).

Therefore, since most studies rely on partial processing by introducing both edge and cloud for task accomplishment, this research proposes the full implementation of multiple tasks on limited-resource devices, also known as *"Edge-devices"* or *"Nodes"*, without relying on high-computational resources such as Cloud or Fog to achieve lower latency with satisfactory results and prevent invasion of privacy, DDoS attacks and more critical aspects.

## 2.6 Chapter Summary

This chapter included a thoroughly comprehensive review and discussion. To begin with, the main contribution of this thesis is detecting objects on resource constrained-devices in noisy environments, as opposed to previous studies, which focused on only data-source captured in clear conditions and during the daytime where light is sufficient to recognise and detect instances. In addition, an overview of classification methods for weather conditions, where numerous studies have focused on using a variety of approaches to categorising meteorological situations into different groups. On the other hand, other forms of deterioration, such as low light, blur and noise (e.g. salt-and-pepper), are roughly neglected or studied and measured with less care. Finally, an overview of the most essential and contributing part, the low-light image enhancement, with a brief review of other image restoration techniques, is provided to enhance captured image feature representation for better visualisation and to aid high-level tasks to perform well.

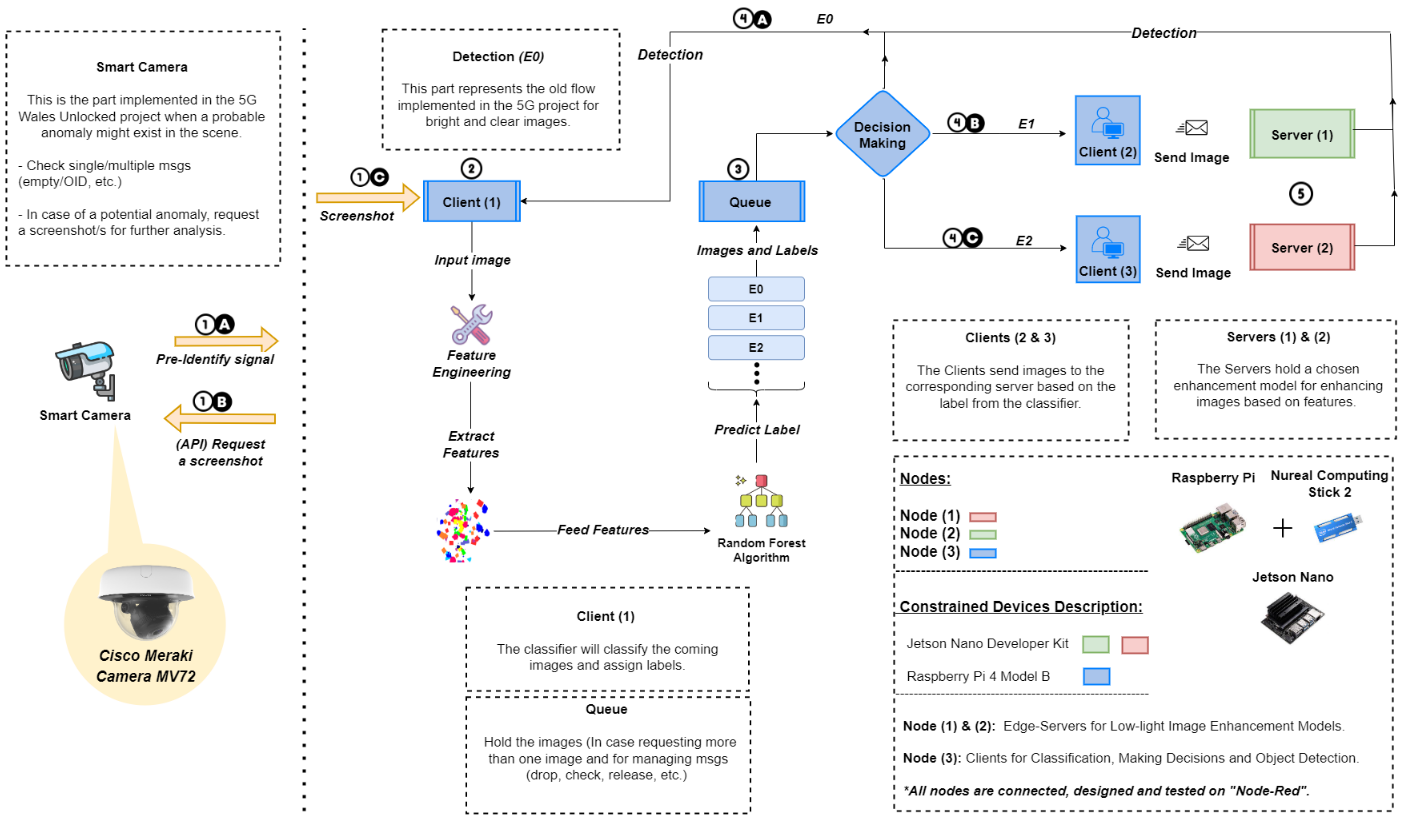
# Chapter 3

# Methodologies

Intelligent video surveillance is one of the most non-trivial topics researched over the last few years [112]. Anomaly Detection (AD) plays a vital role in video surveillance to maintain safety in residence places, whether in indoor or outdoor environments. Indoor areas are easy to control and able to adjust. For instance, install fixed lighting and camera position and cover a small room or a specific region. Moreover, areas are not exposed to external factors such as raindrops, sunlight, etc. On the other hand, In-The-Wild (ITW) environments, it is unfeasible to avoid unexpected, sudden weather changes and insufficient illumination during dawn, dusk and darkness in the nighttime, causing systems and applications to fail when carrying on monitoring and controlling tasks as well as to underperform in detecting anomalies in outdoor scenes.

Thus, to have a system that performs well in such environments, this research studies anomaly detection (e.g. person & vehicles) under low-light conditions on resource-constrained devices at the edge by exploiting the existing techniques in Machine Learning (ML) and Computer Vision (CV) studied before. The proposed architecture introduces low-cost edge devices as the main components for the full implementation. Tasks are distributed to multi-nodes for performing specific tasks such as classification, making decisions, enhancement, detection and sending alerts. Figure 3.1, shows the proposed design from source to destination.

Each node is thoroughly described individually to enhance understanding. The numbers in the diagram represent the sequential flow steps of the input image from source to destination, and they are detailed in the following subsections.



Smart Camera
This is the part implemented in the 5G Wales Unlocked project when a probable anomaly might exist in the scene.
- Check single/multiple msgs (empty/OID, etc.)
- In case of a potential anomaly, request a screenshot/s for further analysis.
Smart Camera
1A Pre-Identify signal
1B (API) Request a screenshot
Cisco Meraki Camera MV72
1C Screenshot
Detection (E0)
This part represents the old flow implemented in the 5G project for bright and clear images.
Detection
2 Client (1)
Input image
Feature Engineering
Extract Features
Feed Features
Random Forest Algorithm
Client (1)
The classifier will classify the coming images and assign labels.
Queue
Hold the images (In case requesting more than one image and for managing msgs (drop, check, release, etc.)
4A E0
Detection
3 Queue
Images and Labels
E0
E1
E2
Predict Label
Decision Making
4B E1
4C E2
Client (2)
Client (3)
Send Image
Send Image
Server (1)
Server (2)
5
Clients (2 & 3)
The Clients send images to the corresponding server based on the label from the classifier.
Servers (1) & (2)
The Servers hold a chosen enhancement model for enhancing images based on features.
Nodes:
Node (1) 🟥
Node (2) 🟩
Node (3) 🟦
Raspberry Pi
Nureal Computing Stick 2
Jetson Nano
Constrained Devices Description:
Jetson Nano Developer Kit 🟩 🟥
Raspberry Pi 4 Model B 🟦
Node (1) & (2): Edge-Servers for Low-light Image Enhancement Models.
Node (3): Clients for Classification, Making Decisions and Object Detection.
*All nodes are connected, designed and tested on "Node-Red".

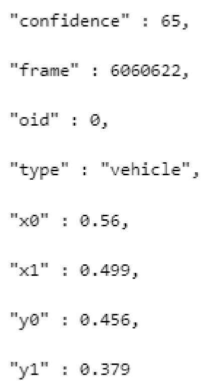
**Figure 3.1** Design proposed for detecting anomalies on the constrained devices for low-light environments for the edge paradigm.

## 3.1 Meraki Camera

The Meraki camera depicted in Figure 3.2a is one of the most recent intelligent cameras introduced by Cisco for Video Surveillance Systems. The camera is more suited to indoor areas to ensure safety, space occupancy, and other functions. Indoor environments provide static lighting and restricted and limited regions, which enables the capture of clear and high-quality images for investigation and further analysis. In addition, the camera offers several capabilities to assist users and authorities in analysing surrounding activities and behaviours, such as *"How many people are entering/exiting a specific room?"* and *"Which* *zone is more crowded?"*, to name a few. As indicated, the camera was utilised in most outdoor 5G Wales Unlocked project use cases. This camera sensor is one of the system's primary components for gathering data, storing video streams, and providing information regarding scene states such as audio level, light intensity, and information about people and cars. To bind the camera functionality in the 5G Wales Unlocked with the proposed method, it is assumed that the camera will provide a potential anomaly if the audio level and lux value are above a certain threshold, besides a possible detection of a person, car or both. Under these conditions, an API request for screenshot capture is triggered, and the captured image is passed to the subsequent node for a separate task. However, since anomalies are rare and activities in the use cases occasionally happen, the user triggers and sends images directly to the adjacent node for test and evaluation purposes.

For example, 3.2b depicts camera outputs. The output includes each of (1) audio level in dB, (2) pixel intensity; high value for a brighter scene and low value for a darker one, (3) timestamp and (4) object recognition results, which provides the following metadata; (a) probability (%) that a given instance is a vehicle or a person (b) frame number within the video stream (c) object Id; a unique number assigned for every new object (d) classes of the detected object (e.g. person or vehicle) and finally, (e) bounding box (rectangle) coordinates surrounding the object detected, respectively.

**(a)** MV72 Meraki Camera.


```
"confidence" : 65,

"frame" : 6060622,

"oid" : 0,

"type" : "vehicle",

"x0" : 0.56,

"x1" : 0.499,

"y0" : 0.456,

"y1" : 0.379
```
**(b)** Example of a *"pre-identify signal"* by the camera.

**Figure 3.2** (a) Camera used in the project and it is (b) sample of Meraki Camera output.

## 3.2 Low-Light Enhancement Models

Low-light image enhancement methods can enhance the visual perception of images that have deteriorated due to being acquired in low-light conditions. Deep Learning, or DL for short, has lately emerged as the conventional method that is most effective in solving problems and making contributions in this area. However, poor illumination may manifest in various forms and characteristics. Non-uniform, dim, back-lit, and extremely low light are just a few examples of sub-optimal lighting circumstances that might impact the underlying data source. In addition, most of the degradation occurs to images during acquisition, compression processes and transmission channels before the data is provided for further study.

Additionally, these characteristics undoubtedly affect high-driven tasks such as tracking, classification, and detection in autonomous driving and surveillance systems, to name a few. As Chapter (2) mentioned, methods may be further broken down into data-driven and traditional approaches. For instance, the Retinex approaches concentrate on separating images into their illumination and reflection components, with the reflection component only serving as the stage for the enhancing process. Therefore, methods based on Retinex leave out several degradation types, which results in an accumulation of colour artefacts and noise leading to a quality diminishing on the final output. This section presents an overview of low-light enhancement models, which are compared based on the network structure, data training, and other criteria. In addition, specific criteria were considered to pick and avoid particular approaches depending on the essential needs detailed later.

Moreover, model results and performance are presented and described in Chapter (4), which deals with assessing model compression and how it relates to the detection phase and other critical metrics. The overall aim of applying existing approaches for low-light image enhancement is to precisely recognize objects in dark scenes rather than enhance or evaluate the image enhancement in terms of image quality. Consequently, the emphasis will be placed on the findings produced after the detection stage. To do so, the state-of-the-art object detection algorithm chosen for this research will be evaluated using a wide range of metrics.

**Table 3.1** Summary of models proprieties based on deep learning methods. **N/A**: For Unpaired data or Data is not required for training, **TF**: For TensorFlow.

| Model | Network Structure | Trained On | Tested On | Color Space | Framework |
|---|---|---|---|---|---|
| **RetinexNet** | Multi-scale network | LOL | Self-collected | RGB | TF |
| **MBLLEN** | Multi-branch fusion | Synthetic Data | Self-collected | RGB | TF |
| **LTSITD** | U-Net | SID | SID | RAW | TF |
| **KinD** | U-Net | LOL | LOL and Other public datasets | RGB | TF |
| **KinD++** | U-Net | LOL | LOL and Other public datasets | RGB | TF |
| **TBEFN** | U-Net | SCIE LOL | SCIE, LOL and other public datasets | RGB | TF |
| **EnlightenGAN** | U-Net | N/A | **ExDark** and Other datasets | RGB | PyTorch |
| **RRDNet** | Three-branch CNN | N/A | LIME and Other datasets | RGB | PyTorch |
| **DRBN** | Recursive Neural Network | LOL | LOL | RGB | PyTorch |
| **UTVNet** | Unfolding Total Variation Network | sRGBSID generated from SID | sRGBSID generated from SID | sRGB | PyTorch |
| **RUAS** | Prior Architecture Search Network | LOL MIT-5K | LOL MIT-5K | RGB | PyTorch |
| **DSLR** | U-Net | MIT-5K | MIT-5K and Self-collected | RGB | PyTorch |
| **CSDNet** | Context-sensitive decomposition network | MIT-Adobe 5K and LOL | **ExDark** and Other public dataset | RGB | PyTorch |
| **CSDGAN** | Context-sensitive decomposition Generative Adversarial Network | Unpaired data of EnlightenGAN | NPE, NASA, MEF, and LIME | RGB | PyTorch |
| **Zero-DCE** | U-Net | SICE | SICE | RGB | PyTorch |
| **Zero-DCE++** | U-Net | SICE | SICE | RGB | PyTorch |
| **Retinex-DIP** | Encoder-Decoder Network | N/A | **ExDark** and Public data | RGB | PyTorch |
| **ExCNet** | Fully connected layer | Self-collected | Exposure Database (IEpsD) | RGB | PyTorch |
| **REDIIRT** | Amplification Network | SID | SID | RAW | PyTorch |

## *Overview*

As a reminder, this research's foundation uses previously developed models in DL and CV to enhance low-light images in real-world scenarios. As a result, a comprehensive overview of models is offered, including a concise discussion of the structure, framework, training, testing, and framework employed. A comparison of the tested models in the current study can be seen in Table 3.1. It is essential to consider that several other models for low-light images have been researched over the last few years. Most of them emphasise gleaning characteristics from paired data, which occurs when an image is taken with inadequate lighting (low light), and its equivalent image of the same scene is taken under normal conditions (referred to as bright). Obtaining the same scenes, bright and low light scenarios, is laborious and calls for specific camera settings and calibrations. In some circumstances, models that depend on paired data perform poorly in real-world settings because the same type of "kernel" is applied to all data, which enables models to learn certain traits but limits the models' ability to figure out different lighting conditions.

Regarding the "Network Structure" column in Table 3.1, it is interesting to note that the U-Net design is the most often utilised across all models. The reason is that U-Net was developed to address one of the most critical high-driven tasks, Semantic Segmentation, as stated in the study published in [79], instead of encoding the whole image with a label like standard methods of machine learning (e.g. Random Forest (RF) and Support Vector Machine (SVM)). Semantic Segmentation can categorise individual pixels, or "pixel-wise," into the appropriate category. In addition, the bio-medical image classification problem was the motivation for the proposal of U-Net. This task often classifies different cells or tissue textures, including a few examples. The straightforward construction of this sort of network, which consists of an encoder followed by a decoder, is one of the most critical factors that encourage researchers to adopt this type of network in the low-light field. The role of the encoder, also known as the "Contraction" component, is for Down-sampling features in multi-scales.

In contrast, the purpose of the decoder, known as the "Expansion" part, is to Up-sampling those features in a high-resolution representation. Regarding the dataset used for training or testing, only a few strategies were examined using the Exclusively-Dark dataset as a test bed. The ExDark dataset will be discussed in further depth in the following subsection, explaining why this dataset was selected among the others. On the other hand, the remaining studies concentrated on dividing the same dataset into training and testing batches, which, in most instances, produce over-fitting and allow learning-restricted features. It is noticed from Table 3.1 that non of the models utilised the ExDark dataset for training and model creation; only the EnlightenGAN, Retinex-DIP and CSDNet employed the dataset for testing, which encourages evaluating these models on unseen data for the detection phase.

### *Model Selection*

In recent years, several pre-existing methods for image enhancement have been investigated to find solutions to problems in computer vision, particularly concerning the issue of low-light conditions. However, as indicated in [47], between 2017 and 2021, more than a hundred strategies have been suggested for enlightening images by preserving image features and quality. Indeed, it is not easy and requires exertion to compile all current models in this area for training or testing purposes. In addition, each model needs its independent environment settings (for example, the type of framework used, refer to Table 3.1). Also, the data source determines whether or not architecture modifications are required. therefore, there are no criteria for collecting models for this research. However, the most recent techniques in this subject have been explored and open-sourced, with public implementation accessible. Table 3.2 briefly explains rejecting a few models during the testing phase. When these parameters are employed for the enhancement step, models like the Retinex, ExCNet, and RRDNet execute a search procedure to discover an effective regularisation or optimum parameters during inference. However, approaches based on this concept may include artefact colours and be time-consuming for real-world applications. The UTVNet and DRBN, on the other hand, need ground-truth images, which implies that every low-light image must have an analogous normal or bright image for inference. Otherwise, specific network topology changes are necessary to operate directly on the test dataset.

Furthermore, models trained on a particular colour model must be tested on the same type, such as RAW format. Otherwise, the training step for the subsequent methods, Chen et al. (Learning to see in the dark) and REDIRT, is necessary. Finally, models like the KinD-family and EnlightenGAN are included in the testing and evaluation process, but only after being resized to a smaller size for speedier processing and, importantly, to avoid failure during inference (e.g. Process Killed). Furthermore, this research presented new models known as the CSD-Family, a combination of numerous models trained and evaluated on varied data and with diff trainable parameters producing heavy and lightweight networks, to mention a few. Other research, however, has yet to previously study CSD-Family for the object recognition task and image quality assessment. Ultimately, given that the primary goal is to assess the high-driven activity, Object Detection (OD), any enhancement model that outperforms the existing ones may be substituted. In other words, a demonstration that these models might be suitable and effective as a pre-processing step for enhancing subsequent tasks *(e.g. Classification and Detection)* while preserving output, quality and latency and fitting into resource-constrained devices. Thus, providing efficient results and better representation for instances in-the-wild environments with poor illumination aid in detecting anomalies for maintaining safety in the surrounding ambient. In addition, issue notices or alerts to notify the relevant authorities and users so they may take necessary action.

**Table 3.2** Models were discarded from experimentation and evaluation.

| **Models** | **Why?** | **Tested?** |
|---|---|---|
| ***Retinex, ExCNet & RRDNet*** | Training to find optimal value during inference. | ✗ |
| ***KinD, KinD++ & EnlightenGAN*** | Image size must be small. | ✓ |
| ***UTVNet & DRBN*** | Paired data needed. | ✗ |
| ***Chen et al. & REDIIRT*** | Only RAW format. | ✗ |

## 3.3 Object Detection

Video Surveillance Systems (VVS) respond to real-world events by understanding scenes provided by cameras when captured data are passed for further tasks (e.g. detection and tracking). Therefore, VSS must be able to detect distinct objects in its surrounding (e.g. people, cars...etc.) The 2D Object Detection (OD) task is an evolving and emerging field using deep learning neural networks. The generated output is a bounding box wrapped around the detected object and includes numerical coordinates with confidence (%) and a class name. The specifics of this output are determined by the dataset used to train these detectors to identify desired objects. For example, most of the techniques developed in this area have been trained on image datasets such as ImageNet [42] and MS COCO [56], which include hundreds of object names.

Furthermore, specific class recognition requires fine-tuning or transfer learning methods (e.g., one class only). Numerous recent studies contributed novel architectural designs and foundational concepts. However, reviewing the previously developed models in this domain is not the objective of this research.

Further, comparing and analysing various models becomes irrational due to the possibility that specific techniques need to be updated and outmoded. One-stage and two-stage detectors are the two primary categories that make up the many types of detectors. One-stage procedures are much quicker in the training and inference stages. In contrast, two-stage procedures require more time and effort to train but provide more accurate results than those produced by one-stage methods. Thus, when deciding between the two kinds, one common trade-off that focuses on speed against accuracy may be made based on the use case and application. One of the prevalent models for a single stage is known as YOLO, which stands for "You Only Look Once," and its several modifications (from Yolo to Yolov7) until this thesis was written. At the same time, the Region-Based Convolution Neural Networks (RC-NNs) are gaining interest for two-stage models, which produced various versions including RCNN, Fast-RCNN, and Faster-RCNN [88]. In the following subsection, an overview of the detectors used in the current research will be presented along with brief details and additional information regarding the environment setting that was utilised for the successful execution of the proposed detection model.

### 3.3.1 Detectron2 (Overview)

In the era of deep learning, cloud-based is a well-recognized advantage among alternative paradigms when picking a dependable and robust environment to train and test powerful models. Due to the capacity to process and retrieve results in a short amount of time (referred to as low latency). In addition to the vast availability of storage and high-powered computer resources (e.g. GPU, VPU, etc.). In this study, benchmarking and evaluation of the low-light enhancement models are carried out with the assistance of the Faster-RCNN base model with default configuration under the name *"Detectron2"*.

It is essential to point out that Detectron2 was used for all use cases during the 5G Wales Unlocked project for the detection phase. Detectron2; is a Faster-RCNN version established by Facebook AI Group Research's next-generation approach that implements the state-of-art object detection [104]. The model was trained on the large volume and popular dataset ImageNet [42]. Various base models are provided for substitution depending on user requirements which vary in speed for training and predicting, as well as model accuracy (mAP%). The trade-off between deep learning approaches is constantly considered and addressed when choosing a particular model since an optimal and generalized model is hard to obtain. For the 5G Wales Unlocked project and current research, Faster-RCNN with X101-FPN base model is selected, which provides an accuracy of 43.0%. However, it takes a long time to train and predict. Therefore, despite the significant training time and memory for training, which is 0.638 seconds per iteration and 6.7 GB, respectively, the default configuration of the highly accurate base model "X101-FPN" is utilized for inference only by exploiting massive computational resources provided by the cloud-based. The environment used for running both enhancement and detection approaches is described later.

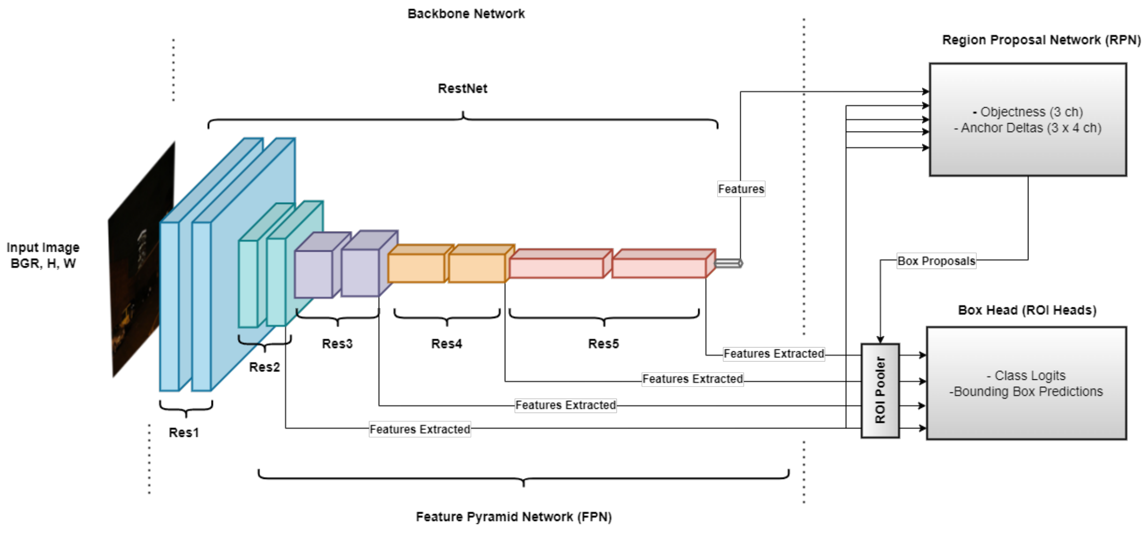


**Figure 3.3** Faster-RCNN architecture with the main components [75].

Figure 3.3 provides a visual representation of the network architecture of the Faster-RCNN base model, which is comprised of the following three primary components:

❊ **Backbone Network:** Within the scope of this component, the Feature Pyramid Network (FPN) [55] composed of RestNet models (Rest1-Rest5), is used to extract features from input images at various scales. Using characteristics with several scales improves the prediction of anchor boxes of varying sizes.

❊ **Region Proposal Network (RPN):** After producing a variety of characteristics based on the prior network, RPN finds object regions in which an object likely exists in that region (1000 box proposals, by default) in addition to the confidence score (%).

❊ **Box Head:** A Fully-Connected Layer predicts bounding boxes and image labels. Non-Maximum Suppression (NMS) is introduced to filter out the bounding box with a maximum confidence score for the final output.

### 3.3.2 Environment Setting

This sub-section provides a short description of the environment exploited for running different deep-learning models; to enhance low-light images and identify objects through the detection phase. As previously discussed, a cloud-based service offers magnificent assistance in rapidly completing and easing the processing of many tasks, such as training and inference, to name a few. Therefore, in order to accomplish the goals outlined above, the present study made use of Super Computing Wales (SCW) under the name *Hawk System* [17]. Hawk is located at Cardiff University, home to several high-powered computer resources supporting various research and development endeavours. It comprises several "nodes" (280 nodes), whereby the combination of two or more nodes produces a strong cluster capable of processing intensive and multiple tasks simultaneously.

For this project, only two isolated nodes based on availability under the name *"gpu or gpu_v100"* were used for testing and evaluating enhancement and detection methods adopted for this research. Two distinct nodes were employed during the project, depending on their respective availability. The capabilities and characteristics of each node are broken down into the following points:

**CPU:** Intel(R) Xeon Gold 6148 CPU @ 2.10GHz (x86_64-bit).

**GPU:**

- **gpu:**
  - NVIDIA Tesla P100, CUDA Version: 11.5.
  - 16 GB of Memory for GPU.
  - 29 GB of Memory (for the whole allocated node).
  - 1792-Cores.
- **gpu_V100:**
  - NVIDIA Tesla V100, CUDA Version: 11.5.
  - 16 GB of Memory for GPU.
  - 29 GB of Memory (for the whole allocated node).
  - 2560-Cores.

**OS:** Red Hat Enterprise Linux Server release 7.9, Core-4.1-amd64.

## 3.4 Lightweight Dynamic Image Classification

Most state-of-the-art (SOTA) image classifiers are built on top of pre-trained Convolutions Neural Networks (CNNs). A classifier will take an image as input and then output a probability, expressed as a percentage, for a label representing the scene's content. Methods of classification can be broken down into feature engineering and data-driven approaches. As mentioned in the literature section, traditional methods extract and select definite features to contribute to the training and learning processes; nonetheless, it is necessary to have a good understanding of those features. Whereas modern methods automatically obtain and learn infinite features from the given image.

Generally, the output produced by both methods comes up in different representations, such as binary, numerical, or categorical labels. In the past, numerous models have been investigated in the classification field. These models were trained on substantial data to categorise images into various categories. AlexNet, for example, is regarded as a lightweight classifier that places a low demand on the available resources. Following the feature extraction phase, it can categorise images into up to one thousand distinct classes [42]. In addition, it can operate on the Central Processing Unit (CPU) of low-cost devices *(e.g. mobiles and single-board computers)* at an approximate speed of fewer than three seconds. Therefore, image classification based on traditional machine learning is the primary goal for a dynamic classifier to distinguish different types of image brightness.

The 5G Wales Unlocked project primarily concentrated on dealing with images of high quality that were taken during the daytime hours. As a result, this algorithm aims to differentiate and discriminate between the standard pipeline and the newly proposed one. Images are separated into three distinct categories—bright or normal and different types of dark images to carry out decisions and subsequent processing. As mentioned in the preceding section 3.1, the camera can measure and report illumination readings regularly, which is the quantity of incoming source light on the scene. However, it only represents total pixels arithmetic mean, which is computing the average of the RGB pixel values in a given image, rather than pixel-wise calculation. Memon et al. [70] proposed a simple method based on pixels manipulation for calculating single-pixel luminosity after separating an input image into three channels R, G & B. The mentioned study investigated one additional class compared to the usual and widely researched light visibility circumstances. Semi-dark images depict partly dark situations, which implies that some areas are dark while others are light in the same input image. However, it is still challenging to differentiate and label images as dark or semi-dark due to different human perspectives. As a result, it is assumed that current low-light image-enhancing algorithms differ in functionality and performance, with some relevant for completely dark instances and others for partly dark ones.

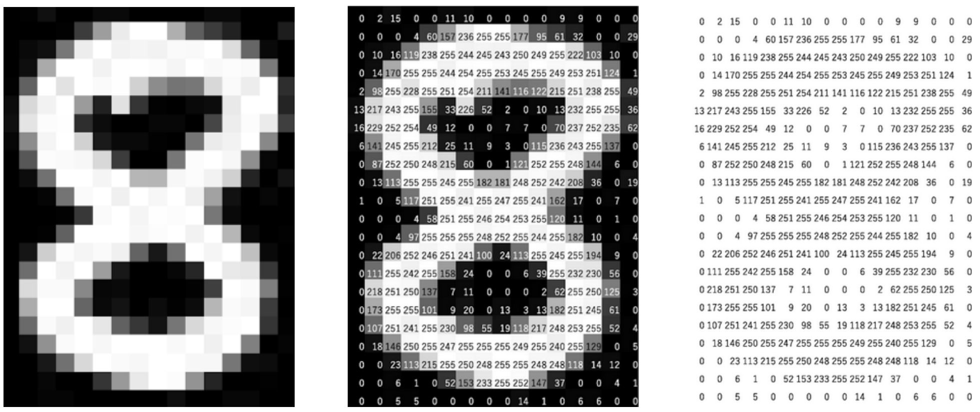
Furthermore, models that perform well for particular images may only enhance dark areas while preserving bright ones by maintaining image quality without adding noise or colour artefacts. Therefore, this research proposes a dynamic low-cost classifier composed of a feature extractor and a Random Forest (RF) algorithm to accomplish the classification task. The following subsection presents an overview of the method components, the feature extraction process and the classical machine learning algorithm used for training and testing.

### 3.4.1 Feature Engineering

The terms *"features"* is often brought up when discussing conventional approaches to machine learning. Most of the time, it is pertinent to the learning style known as *"Supervised Learning"*, where labelled data is available to train the model producing a set of features that can assist in making accurate predictions. The feature extraction process involves applying a selected filter or filters, also referred to as *"kernel"*, directly on the input image to produce a new image with the exact size of the original one. This equation is written as $\acute{f} = I \bigodot K$, where "$i$" stands for the **input image**, "$K$" stands for the **kernel**, and "$\acute{f}$" are the features that define the input pictures and discriminate between them in a variety of ways, including texture, edges, and luminance, to mention a few. As the name indicates, features are characteristics; therefore, depending on the nature of the situation, it may not always be challenging to recognise the appropriate characteristics to extract. If this is not the case, it will be required to investigate a vast number of descriptors in order to locate the ideal filters with their hyper-parameters. In addition, to obtain adequate results when choosing an efficient machine learning algorithm to work with these features. In summary, feature engineering allows the user to choose appropriate filters to be applied to the data for feature extraction, in contrast to contemporary Convolution Neural Networks (CNNs), which accomplish the job intuitively without the need for human interaction in terms of choosing specific kernels and their values. Therefore, the approaches offered by traditional techniques serve to build models suited for implementing constrained devices since they have modest sizes and can be produced and inferred quickly.

### Pixel Value

The pixel values are essential features that must be included in the bag-of-features. Since these pixels provide original characteristics before any application of convolution kernels when extracting features task, these values are represented as actual colour intensity *(RGB-color space or grey scale inputs)*, texture, brightness and more. For example, Figure 3.4 shows the original image of 8-digit in the MNIST dataset *(Left)*. However, by presenting the input as a grey-scale level *(Center)*, pixels can be differentiated by only looking at pixels within the approximate range 180-255 *(Right)*, for representing the actual colour "White" and "Black" for the remaining pixels. In this case, only pixel values might accomplish the job, express the whole image and distinguish it from other digits.



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 15 | 0 | 0 | 11 | 10 | 0 | 0 | 0 | 0 | 9 | 9 | 0 | 0 | 0 |
| 0 | 0 | 0 | 4 | 60 | 157 | 236 | 255 | 255 | 177 | 95 | 61 | 32 | 0 | 0 | 29 |
| 0 | 10 | 16 | 119 | 238 | 255 | 244 | 245 | 243 | 250 | 249 | 255 | 222 | 103 | 10 | 0 |
| 0 | 14 | 170 | 255 | 255 | 244 | 254 | 255 | 253 | 245 | 255 | 249 | 253 | 251 | 124 | 1 |
| 2 | 98 | 255 | 228 | 255 | 251 | 254 | 211 | 141 | 116 | 122 | 215 | 251 | 238 | 255 | 49 |
| 13 | 217 | 243 | 255 | 155 | 33 | 226 | 52 | 2 | 0 | 10 | 13 | 232 | 255 | 255 | 36 |
| 16 | 229 | 252 | 254 | 49 | 12 | 0 | 0 | 7 | 7 | 0 | 70 | 237 | 252 | 235 | 62 |
| 6 | 141 | 245 | 255 | 212 | 25 | 11 | 9 | 3 | 0 | 115 | 236 | 243 | 255 | 137 | 0 |
| 0 | 87 | 252 | 250 | 248 | 215 | 60 | 0 | 1 | 121 | 252 | 255 | 248 | 144 | 6 | 0 |
| 0 | 13 | 113 | 255 | 255 | 245 | 255 | 182 | 181 | 248 | 252 | 242 | 208 | 36 | 0 | 19 |
| 1 | 0 | 5 | 117 | 251 | 255 | 241 | 255 | 247 | 255 | 241 | 162 | 17 | 0 | 7 | 0 |
| 0 | 0 | 0 | 4 | 58 | 251 | 255 | 246 | 254 | 253 | 255 | 120 | 11 | 0 | 1 | 0 |
| 0 | 0 | 4 | 97 | 255 | 255 | 255 | 248 | 252 | 255 | 244 | 255 | 182 | 10 | 0 | 4 |
| 0 | 22 | 206 | 252 | 246 | 251 | 241 | 100 | 24 | 113 | 255 | 245 | 255 | 194 | 9 | 0 |
| 0 | 111 | 255 | 242 | 255 | 158 | 24 | 0 | 0 | 6 | 39 | 255 | 232 | 230 | 56 | 0 |
| 0 | 218 | 251 | 250 | 137 | 7 | 11 | 0 | 0 | 0 | 2 | 62 | 255 | 250 | 125 | 3 |
| 0 | 173 | 255 | 255 | 101 | 9 | 20 | 0 | 13 | 3 | 13 | 182 | 251 | 245 | 61 | 0 |
| 0 | 107 | 251 | 241 | 255 | 230 | 98 | 55 | 19 | 118 | 217 | 248 | 253 | 255 | 52 | 4 |
| 0 | 18 | 146 | 250 | 255 | 247 | 255 | 255 | 255 | 249 | 255 | 240 | 255 | 129 | 0 | 5 |
| 0 | 0 | 23 | 113 | 215 | 255 | 250 | 248 | 255 | 255 | 248 | 248 | 118 | 14 | 12 | 0 |
| 0 | 0 | 6 | 1 | 0 | 52 | 153 | 233 | 255 | 252 | 147 | 37 | 0 | 0 | 4 | 1 |
| 0 | 0 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 14 | 1 | 0 | 6 | 6 | 0 | 0 |

**Figure 3.4** Image sample form MNIST dataset [11].

### Gabor Filters

Gabor is a classic tool used in image processing for texture analysis, edge detection and feature extraction for image classification and segmentation purposes. It is a Band-pass filter, which allows only specific frequencies to pass while rejecting all other frequencies, unlike High-pass and Low-pass filters that pass only high and low frequencies, respectively. In general, it is a product of a *"sinusoidal"* single of a particular frequency and orientation modulated by *"Gaussian"* wave [3]. The representation formula of the filter is shown below:

$$g(\mathbf{x},\mathbf{y},\sigma,\alpha,\theta,\lambda,\gamma,\phi) = exp(-\frac{\acute{x}^2+\acute{y}^2\gamma^2}{2\sigma^2}) * exp[i(2\pi\frac{\acute{x}}{\lambda}+\phi)] \tag{3.1}$$

Where $\acute{x}$ and $\acute{y}$ are expressed as:

$$\acute{x} = x\cos\theta + y\sin\theta$$
$$\acute{y} = -x\sin\theta + y\cos\theta$$

The convolution kernel expressed in [3.1] depends on various parameters, mostly focused on orientation and wavelength, to control the kernel direction and frequency. These parameters are *(x, y)*: refers to the **Kernel Size**, $\sigma$: for **Standard Deviation**, $\theta$: for **Kernel Angle**, $\lambda$: for **Wavelength**, $\gamma$ for **Aspect Ratio** and $\phi$: for **Kernel Offset** from the original center *(0,0)*. The filter has several parameters that allow the production of a disparity of kernel in terms of size, orientation, position and others, resulting in many filters with different values under the name *"Gabor filter bank"*. To be used on the images for local or global features extraction and further tasks [22]. Those parameters were subjected to various value comparisons during the feature extraction phase by generating a filter bank. Therefore, filter banks that were developed are put to use while training a model for the classification task. Indeed, having many features only sometimes leads to better outcomes. On the other hand, features that are more precise and have fewer options tend to perform better. As a result, the next chapter will provide various experiments using various parameter values to demonstrate the best value based on the classifier's overall accuracy on the validation data.

**Sobel (Edge Detection)**

The Sobel operator is a spatial domain filter designed to find the approximate gradient in "*Gx*" and "*Gy*" directions of an image for each pixel by computing a matrix multiplication between the convolution kernel and the original image [25]. It differs from filters that work with frequencies when an image is converted to frequency domain for applying specific masks such as Low-pass, High-pass and Band-pass filters *(e.g. Gabor Filters)*, which allow particular signals to pass. Theoretically, the Sobel convolution kernel is composed of two operations where each one is a $\mathbb{R}^{3x3}$, see below:

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \& \quad Gy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Where magnitude and angle are calculated as:

$$G_{Mag} = \sqrt{Gx^2 + Gy^2} \quad , \quad \theta = tan^{-1}(\frac{Gy}{Gx}) \tag{3.2}$$

The kernels are identical in terms of their values, but one is the other with a 90-degree rotation applied to it. The Sobel is an extension of the "*Roberts*" operator. However, the sole difference is the matrix's form, denoted by the notation $\mathbb{R}^{2x2}$. Further, various edge descriptors such as the "Canny," "Prewitt's," "Scharr," and "Laplacian" operators, along with a variety of additional operators, have the same capability to identify edges in digital images. However, in contrast to the previous edge approaches, Sobel has lately shown superior performance in smoothing, conserving features and producing sharp edges [38].

### 3.4.2 Random Forest (*Overview)*

Random Forest (RF) is a technique for Supervised Learning (SL) when labelled data is required. RF is classified as Traditional Machine Learning (TML) since it does not utilise any of the more recent Deep Learning (DL) or Artificial Neural Networks (ANN) approaches. Further, it is used for both classification and regression problems, which is able to predict categorical or numerical dependent variables, respectively. As the name suggests, a collection of decision trees (Forest) are picked in a random fashion (Random) by constructing multiple classifiers to achieve a high level of accuracy in its predictions [23]. The purpose of having more than one decision tree is to overcome the Decision Tree (DT) method limitations since it relies on a single tree for all training data, which might lead to model over-fitting. On the other hand, RF builds several decision trees by producing various *"N"* datasets, so-called *"Bootstrapped Dataset"* [41].

The generated bootstrapped dataset represents two-thirds of data or observations from the original dataset with different features but with the exact size of training data *(One observation might be repetitive)*. The purpose of having many datasets with various features is to create a unique classifier for each bootstrap and multiple for the entire data. Therefore, to obtain different predictions and then combine them based on the majority vote for getting one final output, so-called *"Aggregating"*, see Figure 3.5. In other words, random refers to Bootstrapping and Feature Selection. The first one ensures not the same data is used in each decision tree created, which helps to be less sensitive to the original training data. On the other hand, feature selection focuses on selecting unique N-features for each dataset, so every decision tree generated has different features. The number of features for each bootstrap dataset is calculated by taking the approximate square root of the total number of features. (*features_per_bootstrap* = $\sqrt[2]{x}$ ; where x is the total features in the training dataset and $\in \{1,2,...n\}$) [98]. Compared to DL approaches, which require a large volume of data, TML, such as Random Forest and Support Vector Machine (SVM), may perform better when working with sparse data or a limited dataset of hundreds of observations. Moreover, RF is simple, easy to train, and quick to predict on the test data. In addition, it is an efficient candidate for implementation on devices with limited resources.

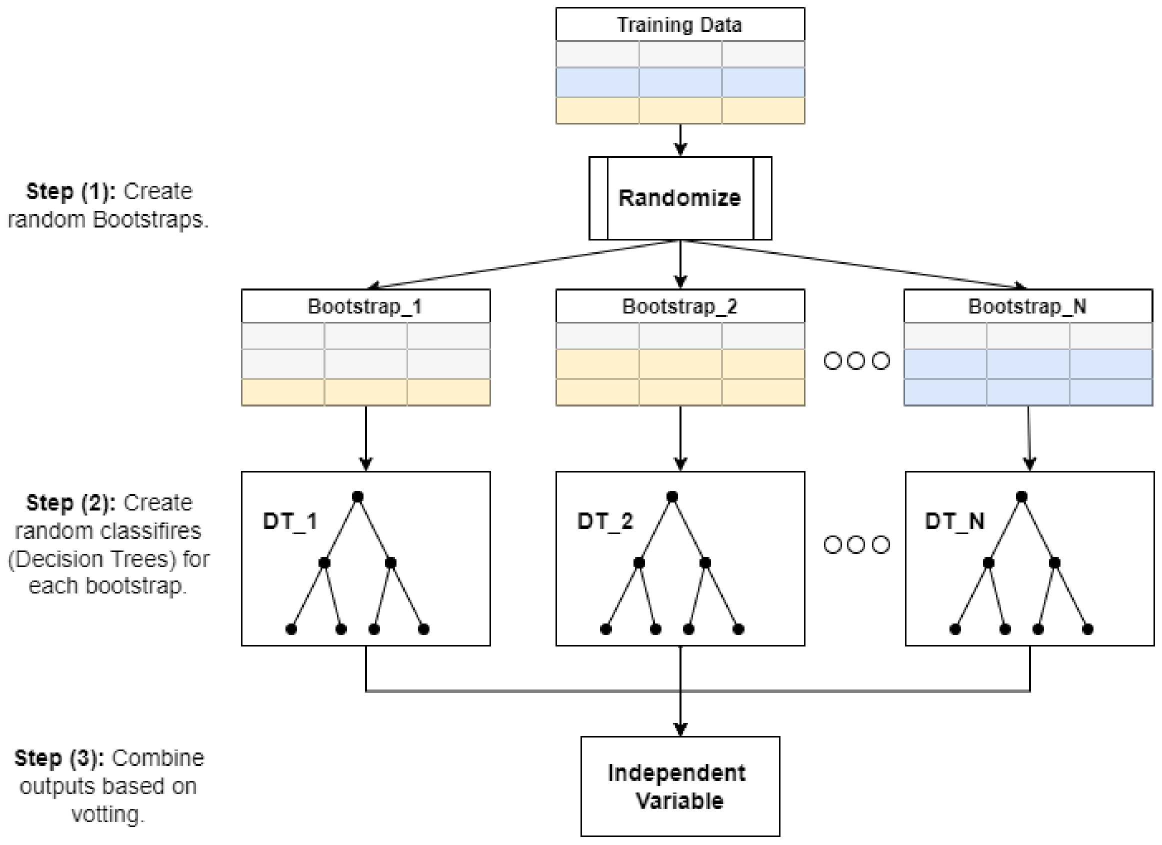

Training Data
Step (1): Create random Bootstraps.
Randomize
Bootstrap_1
Bootstrap_2
Bootstrap_N
Step (2): Create random classifires (Decision Trees) for each bootstrap.
DT_1
DT_2
DT_N
Step (3): Combine outputs based on votting.
Independent Variable

**Figure 3.5** Random Forest representation [83].

In a nutshell, the features that are generated by the recommended feature engineering approach where these features are represented as a bag of features to represent the following labels: $E_0$, $E_1$, $E_2$,...$E_N$, where the first label relates to bright scenes, which do not need to undergo the enhancement step, and the remaining labels represent various enhancement techniques, respectively. The experiments and setup of the proposed method are presented in the next chapter, along with the *RF* algorithm.

## 3.5 Edge Paradigm

Edge computing is the antithesis of cloud computing, particularly in paradigm location. It provides high responsiveness and ensures privacy since facilities and services for edge environments are located closer to the source. Moreover, edge computing offers advantages such as low latency, increased resilience against cyberattacks, and enhanced privacy preservation. In contrast, traditional cloud solutions are essential for handling large datasets and approaches but are susceptible to cyberattacks, which can compromise privacy and increase traffic on transmission channels resulting in longer response times in case assets are exchanged between the two paradigms.

However, it is challenging to get computationally intensive resources and rapid processing due to its reliance on scarce low-cost devices and network congestion most of the time to complete and execute tasks. For example, when inexpensive sensors are installed for real-time data collecting and processing, edge computing is a powerful and extensive technology for IoT applications. However, while executing visual computing and high-powered operations, various factors increase the likelihood of encountering several restrictions and gaps, which might be solved by proposing new strategies.

Recently, several studies have shown a variety of strategies that make it possible for methods based on deep learning to suit resource-constrained devices. These methods manipulate model architectures or offer new methodologies, such as quantisation, pruning, and others like the ones proposed by studies in the literature chapter, coming up with a smaller version by maintaining efficiency and robustness. Nevertheless, this research primarily emphasises using existing pre-existing and lightweight models for task completion. As mentioned earlier, the low-light enhancement models task these models were tested on cloud-based platforms by taking advantage of massive facilities provided by vital services for results evaluation using state-of-art object detection. Therefore, identical models are likewise applied on edge depending on the essential metrics resulting from the detector, processing speed and other factors. On the other hand, recalling the fact that many object detectors have quickly emerged and developed over the last few years, however, the objective of this project is too far from choosing the optimal approach for completing the task at hand or even conducting a compression among many.

Nevertheless, to provide a proof-of-concept for the suggested design. Thus, any outperformed algorithms are able to replace the proposed ones. Ultimately, the lightweight and pre-existing detector named "*Yolov5*" implementation developed by [14] was chosen for the detection phase, even whether the enhancement was required or not. In other words, all input images, regardless of brightness level, are given to the same detector at the end. The following sub-section presents a brief overview of model architecture and proprieties.

### 3.5.1 Yolov5 (Overview)

A one-stage real-time object detection and localisation implementation called *Yolo* (short for "You-Only-Look-Once") is designed and suitable for live video streaming and still images. "*Yolo*" divides an input image into separate grids when each grid detects an object within itself. Over the last several years, a few different versions have been suggested, beginning with "*Yolov1*" and continuing all the way up to "*Yolov7*" at the time of writing the current thesis [95]. However, each version has distinct architectural components, shortcomings, and benefits. Therefore, "*Yolov5s*" was selected to take on the detection phase among the other types; see Table 3.3 for the chosen highlighted and others [14].

As a result, the suggested version is capable of being exported into a wide variety of supported formats, such as TensorFlow-Lite, TensorFlow.js (for JavaScript), the OpenVino Distribution Toolkit, TensorRT, and the Open Neural Network Exchange (ONNX). Therefore, exporting Yolov5 to different frameworks makes it compatible with various operating systems, libraries, and programming languages. Indeed, there is no general rule of thumb for selecting an acceptable strategy for the detection task, except for the existing trade-off between accuracy and speed based on the application requirements. In addition, the model was retained in its initial and default configuration, indicating that no changes were made or any introduced methods, such as transfer learning or fine-tuning.

Therefore, "*Yolov5*" is superior to prior versions in a number of respects, including framework implementation, accessibility, training, and component availability. *Yolov1*-

*Yolov4*, for example, is derived from the "DarkNet" network and was coded in C-language, with the *.cfg* file serving as the configuration one. On the other hand, "*Yolov5*" utilises PyTorch software up to the most recent version, and its configuration file is in *.yaml* format, which simplifies a large number of tasks as well as updates to the network. Among the 5-versions, the minor version, "*Yolo5n*", is excellent for operating on the edge and mobile devices. In contrast, "*Yolo5s*" is ideal for performing inference on CPU for both cloud and edge platforms. On the other hand, larger versions such as "*Yolo5m*" provide a satisfactory compromise between speed and precision. In addition, "*Yolo5l*" works very well with datasets that include objects of relatively tiny sizes. Finally, the final algorithm, "*Yolo5x*", has the most remarkable accuracy but is also the slowest. Because of this, it is best suited for services with high-computational resources with multiple or single GPUs.

**Table 3.3** Yolov5 model versions, **n**: for Nano model size, **s**: for Small model size, **m**: for Medium model size, **l**: for Large model size, **x**: for Extra large model size & **M**: Number of parameters in millions.

| **Type** | **Num. of Parameters** | **Accuracy (mAP 0.5)** | **CPU (Time)** | **GPU (Time)** |
|---|---|---|---|---|
| **Yolov5n** | 1.9 M | 45.7 | 45 | 6.3 |
| **Yolov5s** | 7.2 M | 56.8 | 98 | 6.4 |
| **Yolov5m** | 21.2 M | 64.1 | 224 | 8.2 |
| **Yolov5l** | 46.5 M | 67.3 | 430 | 10.1 |
| **Yolov5x** | 86.7 M | 68.9 | 766 | 12.1 |

Object detection generally comprises the following stages; **Backbone, Neck & Head**. However, each state-of-art varies on the chosen network for the above components, especially among the "*Yolo*" editions. A pre-trained backbone network is the initial step of any object detector. This network is responsible for feature extraction and may operate on a CPU or GPU. The Neck section is primarily intended for use in multi-scale feature maps. On the other hand, Head is most known for forecasting bounding boxes, classes, and confidence scores. Compared to the earlier "*Yolo*" versions (1-4), the **Backbone** component was replaced with a network known as "CSP-DarkNet." CSP acronym stands for Cross Stage Partial Networks, making it possible to process data quicker even when using networks with deeper layers. Furthermore, PANet is employed at the **Neck** stage to get feature pyramids, making it possible to extract and identify features of the same object with varying sizes and scales. For the **Head** stage, retain the same network constructed in the previous "*Yolo*" versions (3 & 4) to predict bounding box coordinates, object category, and class probability [39]. Moreover, the network was given two distinct activation functions: "Leaky-ReLU" and "Sigmoid" for the hidden and final detection layers. These activation functions were designed to improve the performance of the network. In addition, two different types of optimization functions are available, "*SGD*" and "*ADAM*", where "*SGD*" is the default one. In conclusion, the cost-function or loss-function used was the Binary Cross-Entropy with logits loss from PyTorch to determine the scores for objectness, class probability, and Bbox regression. Figure 3.6 illustrates the network architecture of Yolov5 with previously discussed components.

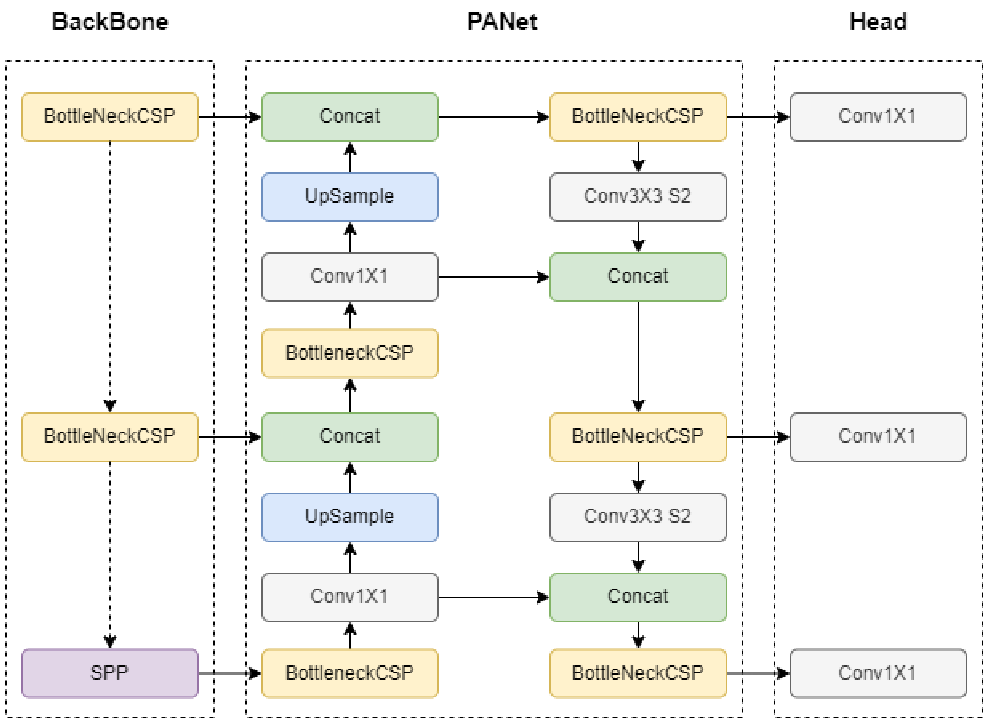


**Figure 3.6** Yolov5 network architecture [39].

## 3.5.2 The Intel® Distribution of OpenVINO™

The OpenVINO is an acronym for Open Virtualization for Inference and Neural Network Optimization. The Intel company released a toolkit under the name *OpenVINO Toolkit* in 2018 [130]. It is a toolkit based on Convolution Neural Networks (CNN) and Artificial Neural Networks (ANN), which enables quickly deploying different neural networks on various Intel platforms. One of the primary advantages of the toolkit is the ability to conduct the inference step more quickly, achieve lower latency, use less computing network bandwidth, and maintain privacy [86]. Moreover, it gives programmers and data scientists tools to accelerate the training and inference stages for deep learning and computer vision Software without requiring additional coding or implementation. The Deep Learning Deployment Toolkit (DLDT) is essential to the OpenVINO Toolkit since it comprises (1) Model Optimizer (MO), (2) Inference Engine (IE), (3) Software, and (4) Samples. **Model Optimizer** trains various neural networks with frameworks, including TensorFlow, PyTorch, Caffe, MxNet, Keras and ONNX. Further, optimization contains the model_downloader, which allows users to download both public and Intel models (***Public:*** models contributed by the mentioned frameworks and ***Intel:*** models contributed by Intel community).

In addition, a model_converter allows the transformation of any model created using the frameworks described above into the Intermediate Representation (IR) format, which results in the creation of two separate files with ".*xml*" and ".*bin*" extensions. All models must be in the (IR) format to be compatible with Intel-developed systems. The **Inference Engine** is the core of model execution and inference which supports multiple kinds of resources; CPU, GPU, Vision Processing Unit (VPU), Movidius Neural Compute such as Neural Computing Stick 2 (NCS2) and Field Programmable Gate Arrays (FPGA). Indeed, inference can be divided and executed in parallel on Multi-device plugin systems. These models are offered to be written in different **Software's**; C++ and Python; thus, it facilitates models deployment as well as a better comprehension of implementation regardless of users' backgrounds. The last components are **Samples or Demos**, a bunch of ready-written scripts for running many applications in Computer Vision (CV) and Natural Language Processing (NLP) domains, to name a few. Thus, a stand-alone script can execute multiple models of similar tasks with the supported languages.

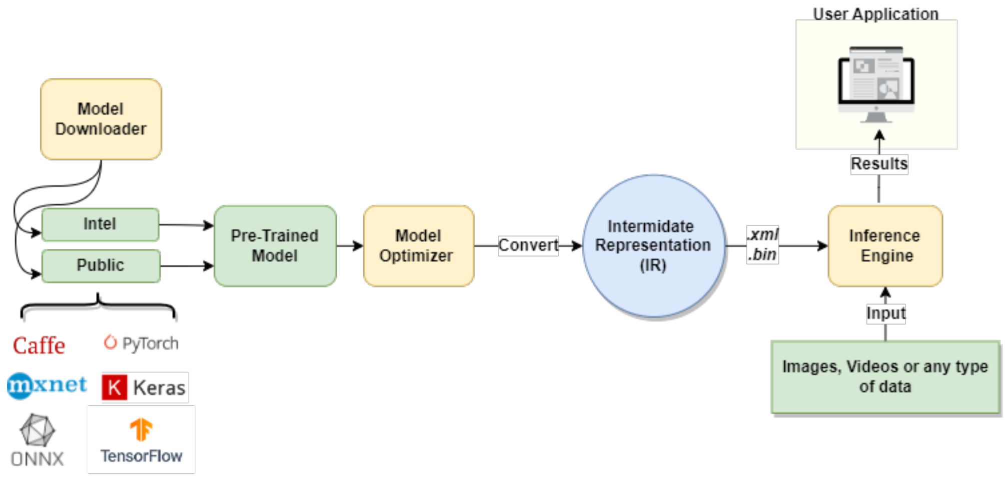


**Figure 3.7** Trained models deployment using OpenVINO Toolkit [16].

The diagram 3.7 depicts the stages required to deploy a model using the openVINO toolkit. The first step is to choose a model from the **Open Model Zoo** database, which includes a large number of pre-trained models useful for projects and solving problems such as Object Detection, Text Analysis and other high-driven tasks provided by Intel or Public frameworks. Afterwards, the **Model Downloader** retrieves a chosen model through name and other model properties specified by users. Each model is constructed with a unique floating-point or integer precision. In addition to INT-18, precision types such as FP16 and FP32 must be provided. As discussed previously, "*Yolov5s*" which consists of 191-layers, 7.2 and 7.46 million parameters and gradient, respectively, was selected to perform the detection task on the edge environment; however, some prerequisites are necessary before the deployment stage with the OpenVINO toolkit.

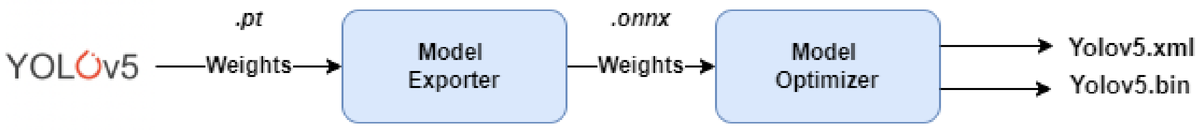


**Figure 3.8** Process of converting Yolov5 to Intermediate Representation (IR) format.

The selected model is a member of the Public group, which is explicitly implemented in the PyTorch framework. The first step is to convert "*.pt*" model weights to "*.onnx*" using the Model Exporter proposed by [14]. After acquiring the ONNX format, an additional conversion step is necessary to acquire the final extensions *(.xml & .bin)* for deployment and inference purposes. However, before converting the model to the Intermediate Representation (IR), the final neural network layers‘ names of ONNX weights must be identified, which was achieved by visualizing the network architecture using Netron [13]. The final layers before the "*transpose*" operation, Conv_487, Conv_471 & Conv_455, represent the target layers to be included in the conversion process. Once again, the Model Optimizer completes the final conversion step; see Figure 3.8. After obtaining the desired format, the Yolov5s-openvino model was used through the **Inference Engine** on the edge device Raspberry Pi (RPi) for the detection task; details are shown in the following Table 3.4. In addition, other device characteristics and attributes are presented in section 3.7.

**Table 3.4** Inference of Yolov5s on the edge device.

| Device | Toolkit Version | Model Used | Task | Plug-In Device | Software | Model Zoo |
|---|---|---|---|---|---|---|
| RPi | OpenVINO_ToolKit_2022 | Yolov5s.xml | Object Detection | Neural Computing Stick (2) | Python | Public |

Lastly, it is essential to note that OpenVino offers several models for cutting-edge object recognition than the proposed one. Indeed, Yolov7 [95] is the latest Yolo-versions released when writing this thesis. However, it still needs to be endorsed by the Intel OpenVINO community.

## 3.6 Dataset

Data collection is one of the leading and significant parts of any research. Nevertheless, it takes time and effort in terms of collecting and proposing at certain times. Since it contributes in many aspects, including model creation, results from visualization and measuring approaches performance. Some studies rely on the same chosen data for experimentation and evaluation by splitting data into train and test sets. On the other hand, others choose separate or a combination of different data to obtain a generalized method that likely works for diverse circumstances. However, data availability depends on a few factors, such as domain access restrictions and research field pervasiveness, where many have contributed to that area. Studies focusing on poor illumination are essential because dark is an integral and inevitable part of our daily life. In addition, illumination variance allows researchers to investigate novel approaches to handle diverse light levels for indoor or outdoor environments.

Moreover, areas covered by dark affect monitoring activities and performing tasks due to lack of visibility. Since low light emerges depending on daytime (e.g. twilight and nighttime) or location (indoor and outdoor) and light source (natural or artificial), leading object detection and other high-driven tasks struggle and perform unwell when it comes to localizing or classifying instances. The challenges posed by low-light conditions have necessitated the collection of a vast quantity of data, which focused on acquiring matched images of the same scene in low light and normal circumstances. In contrast, few works, such as the Exclusively Dark *(ExDark)*, focus on low-light imagery for visual tasks, as suggested by [59]. The ExDark is a suitable dataset for object detection tasks in low-light environments, as it provides bounding box coordinates and class names for various objects. Therefore, this dataset was introduced for two purposes: (1) To train a lightweight classifier that can classify low-light images based on features for selecting the optimal enhancement network among many, (2) to distinguish between bright and dark seances, and (3) To investigate and evaluate the object identification task, specifically for outdoor environments.

The following subsections provide a brief overview of the primary dataset used for both state-of-the-art classification and detection. In addition, the additional datasets are used for training the proposed classifier algorithm.

### 3.6.1 Exclusively Dark Dataset

The ExDark is a collection of 7363 low-light images whose lighting ranges from extremely dark nearly (Zero-Lux) to partly dark (Semi-dark). The data was collected by searching terms such as low light, dark, semi-dark and others on websites and search engines such as Flickr.com and Photobucket.com, to mention a few. Moreover, by extracting images from public datasets offered by [80] & [76], in addition to images captured by smartphones and digital cameras. As previously indicated, the dataset comprises several forms of light that are categorized and correspond to the following labels: (1) Low, (2) Ambient, (3) Object, (4) Single, (5) Weak, (6) Strong, (7) Window, (8) Shadow, and (9) Twilight. Some examples are shown in Figure 3.9. These labels assist in identifying light image types, which is helpful for feature extraction and learning-based classification algorithms.

The ExDark dataset includes various object classes, as shown in Table 3.5. Images were labelled based on the dominant object present in the image. For example, if an image contains five people and two cars, the label assigned would be *People* since images belonging to a specific dataset class may contain different classes.

In the current study, only the people and car instances were considered for experimentation and to map the requirements of the 5G Wales Unlocked project scenarios. However, additional classes, such as bus and motorcycle, were also considered. Moreover, the ground truth bounding box coordinates were generated using Pitor's Computer Vision Matlab toolbox [18]. However, these coordinates have been converted to the "Yolo-Coordinates Normalized" format to facilitate the evaluation stage since the majority of state-of-the-art object detection algorithms produce results in this format.

The detection output consists of the following details:

1. The class name (e.g. person, car, etc.).
2. The X and Y coordinates represent the normalized centre of the bounding box.
3. The W and H represent the height and width of the bounding box.

Furthermore, this dataset was used most heavily in the performance of the enhancement stage, followed by the detection stage due to the various light levels captured in the real-world scenario containing multi-classes. Moreover, arbitrary samples were used to build the classifier besides images taken in normal or bright circumstances.

**Table 3.5** The number of images per class in the ExDark dataset.

| Class Name | Num. of Images | Num. of Objects |
|---|---|---|
| Bicycle | 652 | 5% |
| Boat | 679 | 6% |
| Bottle | 547 | 7% |
| Bus | 527 | 3% |
| **Car** | **638** | 12% |
| Cat | 735 | 4% |
| Chair | 648 | 10% |
| Cup | 519 | 7% |
| Dog | 801 | 4% |
| Motorbike/Motorcycle | 503 | 5% |
| **People** | **609** | 31% |
| Table | 505 | 6% |
| **Total** | **7363** | **23,710** |

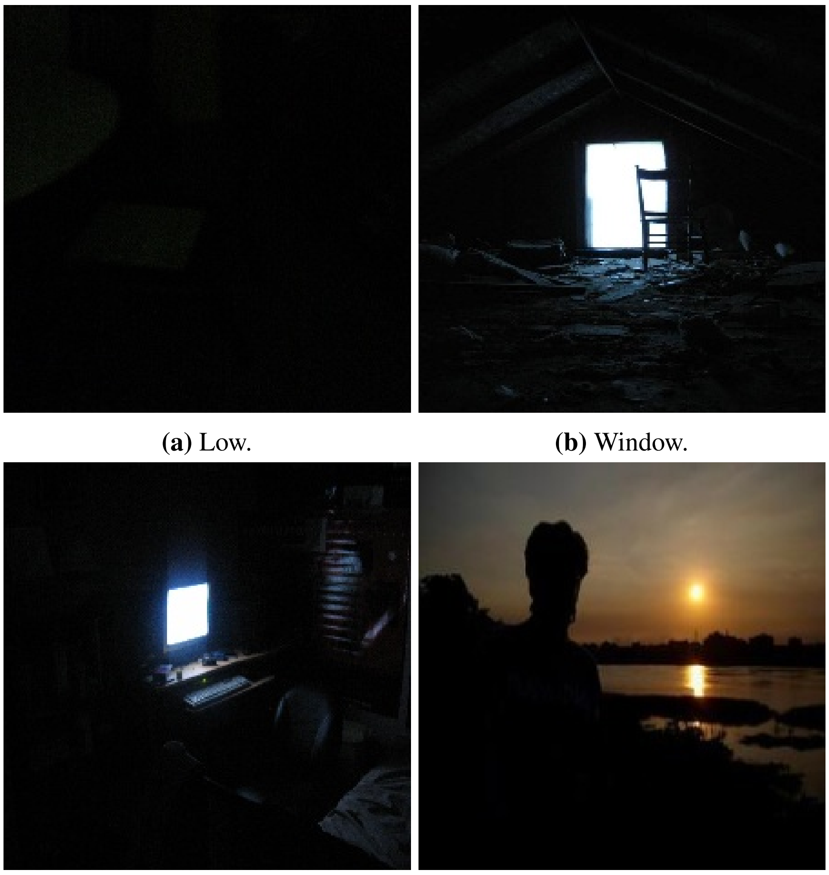

**(a)** Low. **(b)** Window.
**(c)** Screen. **(d)** Twilight.

**Figure 3.9** Illustration of various light types.

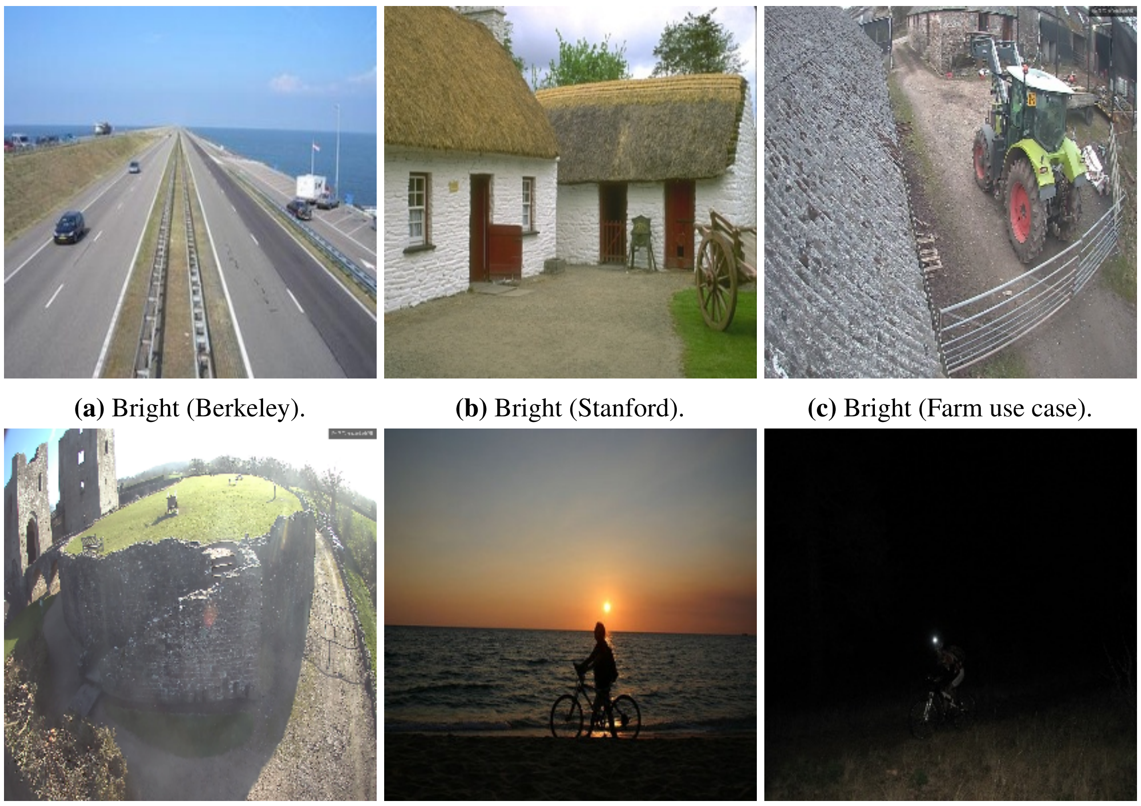
### 3.6.2 Additional Datasets

As mentioned, including relevant data in developing and analysing recommended approaches is paramount. More data samples have been brought in for constructing the proposed classifier, all coming from separate datasets. Because the ExDark dataset only contains scenes captured in dark and partially dark circumstances, the algorithm needs bright samples to differentiate between different light intensity levels. Therefore, bright images were extracted form Berkeley (BSDS-500) [68], Stanford (SBD) [57] & MS COCO [56] datasets. Moreover, sample images taken during the 5G Wales Unlocked project from *Farm & Castle* use cases were also introduced to join the training and testing phases to develop the final classifier discussed in section 3.4. The ExDark was used again to extract low-light features for images described as dark and semi-dark. In fact, there is no definitive method or formula for determining the precise light intensity unless it is exceedingly low-light "*Dark*" or excessively high-light "*Bright*". Eventually, in this study, after compiling the dataset obtained from the various sources, images are described according to the following criteria, also see Figure 3.10.

- **DARK**:
  - If an image is totally dark, lux equals zero.
  - If an object/s is visible for human perception, however, there is no natural or artificial light in the image.
  - If an image is partially dark and bright, light covering a region and dark covering other containing object/s. Also, during dusk and dawn time.
  - If light from several sources (e.g. sunlight, car & street lights) is visible but does not incident any objects in the scene.
- **BRIGHT**:

- If an image is totally bright, more than 100-lux, such as taking in the daytime.

- Even though there is no sun, the weather is cloudy, and the scene can be seen clearly.



**(a)** Bright (Berkeley). **(b)** Bright (Stanford). **(c)** Bright (Farm use case).

**(d)** Bright (Castle use case). **(e)** Partially Dark (ExDark). **(f)** Dark (ExDark).

**Figure 3.10** Illustration of the three light types used for building the classifier.

## 3.7 Nodes Description

This research introduced the incorporation of multi-resource constrained devices "*Nodes*" into the proposed design at the edge computing paradigm for task division and speeding up the inference stage. Because each node is responsible for a specific task, the process may be completed much more quickly while maintaining high efficiency and timely response. In addition, a more precise grasp of the duties performed by each node, the devices used for task accomplishment and the environment in which they operate is provided.

Back to the design 3.1, the first step is kicked off by the Meraki Camera when **(1A)** A pre-identify signal (or signals) is detected and delivered to "*Client 1*" to verify and guarantee message-content in terms of an empty or non-empty message (OID). If the message is empty, indicating a false alert and the message will be discarded.

In any other cases, **(1B)** "*Client 1*" is responsible for requesting a screenshot through the Application Programming Interface (API) provided by the Camera as a built-in function and **(1C)** pass it to "Client 1" for additional processing. Afterwards, **(2)** features are extracted from the input image, followed by a classifier to assign labels in terms of normal (bright) or dark with the appropriate enhancement model label based on the input features. However, in case more than one image is received, **(3)** a queue is designed for holding and storing images and releasing them one at a time immediately after the processing and completion of a single image. Once an appropriate label that accurately has been assigned representing the image features, **(4)** each image follows a unique route determined by the luminosity output. In other words, **(4)(A)** bright images are sent straight forward to the detector for object localization tasks on the same node. On the other hand, inputs that are dark or partially dark **(4)(B)** & **(4)(C)** are delivered by *"Client 2"* or *"Client 3"* to *"Server 1"* or *"Server 2"* for light viewing enhancement, respectively. Both servers *"Server 1"* and *"Server 2"* hold *"E1"* and *"E2"*, representing the *"RUAS"* and *"Zero-DCE++"* as the optimal chosen models for the enhancement task. Afterwards, outputs are returned to *"Client 3"*, where a lightweight object detection is hosted to find instances and generate object outcomes.

The different colours in the diagram represent locations where each activity was carried out on the chosen devices. For example, steps (2 to 4) are carried out on the Raspberry Pi, while the servers are executed on separate nodes on Jetson-Nano Developer Kit devices. Table 3.6 concisely explains each node regarding the task, device implemented and identification. Further, the Node-Red (NR) platform, which enables users to link devices, functions, and other services, was utilized to design, connect and test the proposed design. An example of some functionality NR provides is buttons that can control servers by turning them on and off, regulating and restricting the number of images stored in the queue and more. As a reminder, the NR was used to develop the suggested pipelines for the 5G Wales Unlocked project scenarios.

**Table 3.6** Nodes Ids, roles and edge devices.

| Node | Tasks | Device | Label |
|---|---|---|---|
| **(3)** | Classification, Queue, Decision Making & Detection | Raspberry Pi | Client(s) |
| **(2)** | Low-Light Enhancement | Jetson Nano | Server |
| **(1)** | Low-Light Enhancement | Jetson Nano | Server |

The internal characteristics of the devices utilized for the proposed design and execution are outlined in the following Table 3.7. The Raspberry Pi is one of the single-board computers often employed for various projects, particularly in the IoT domains. In addition, it is regarded as a low-cost gadget due to its adaptability in managing jobs and compatibility. However, when it comes to adapting deep learning and complex models, it performs poorly due to the absence of high-computational resources such as the Graphic Processing Unit (GPU). Therefore, the Jetson Nano Developer Kit devices were exploited in order to ease running deep-learning and computer vision models, referring to low-light image enhancement techniques.

**Table 3.7** Resource-constrained devices used and their proprieties.

| Device | Name | Model | Node | RAM | SD Card | Processor Architecture | OS | External Attachment | Power Adapter | GPU |
|---|---|---|---|---|---|---|---|---|---|---|
| **RPi** | Raspberry Pi | 4 Model B | (3) | 4 GB | 32 GB | ARM-7 | Raspbian | Neural Computing Stick 2 | Voltage: 5V Ampere: 1.5A | ✗ |
| **Jetson** | Jetson Nano | Developer Kit | (1) & (2) | 4 GB | 32 & 64 GB | ARM-64 | Ubuntu | N/A | Voltage: 5V Ampere: 2A | ✓ |

## 3.8 Chapter Summary

This chapter presented a detailed description of the recommended design for the conducted study. It began with a visual representation, illustrating the complete process pipeline at a high level. Each pipeline stage was then further examined and explained in separate subsections. Additionally, an introductory overview of the implemented models was provided, as this research aimed to leverage existing deep learning and computer vision models in conjunction with the Internet of Things (IoT) domain. Furthermore, the environments in which these techniques were tested and evaluated, both in Cloud and Edge-based settings, were discussed. Finally, it is essential to emphasize that all of the findings and assessments will be presented in the subsequent chapter with a higher level of specificity, supported by results presented in tables, figures, evaluations, and more.

# Chapter 4

# Results and Evaluation

In the previous chapter, the mechanisms and theories of each step were handled independently by presenting graphical depiction and overview of every recommended portion of the proposed system. However, to complete the whole picture, results and existing methodologies for evaluation must be clarified and illustrated. Therefore, in this chapter, all the findings and assessment techniques are presented. Furthermore, a concise review of the methodologies used for evaluating deep learning and computer vision techniques composed of classification, enhancement and detection models is also carried out. Moreover, physical measurements of resource-constrained devices and cloud-based have also been included to assess how well systems function while running heavy tasks.

The primary objectives of this study encompass a comprehensive evaluation of the classifier's performance in distinguishing between bright and dark images, as well as its ability to differentiate among various dark images when applied on test-data. To achieve this, several key performance metrics are employed. Firstly, by assessing the algorithm's accuracy to gauge its overall effectiveness, construct ROC curves and compute confusion matrices to gain deeper insights into the classifier's discrimination capabilities.

Furthermore, this study will delve into the realm of object detection by analyzing the accuracy of object detection both before and after applying enhancement techniques. This evaluation will involve metrics such as mean Average Precision (mAP) and Average Precision (AP). Additionally, measuring the inference time, shedding light on the algorithm's efficiency, and the number of bounding boxes generated, which can provide valuable insights into the algorithm's detection precision.

Beyond classifier and object detection performance, by exploring various metrics related to the edge computing environment. This includes an examination of speed, temperature, memory utilization, and CPU and GPU usage. All these metrics are obtained through building a unique dashboard for each edge device using *"Grafana"*, *"Node-exporter"* and *"Prometheus"*.

By comprehensively assessing these metrics, the aim is to gain a holistic understanding of the system's behavior and its performance under varying conditions. This multifaceted approach to evaluation ensures that we obtain a thorough and nuanced view of the system's capabilities and limitations.

## 4.1 Enhancement & Detection

In this part, the findings and assessment of the image enhancement and detection model will be carried out since both phases depend on one another. From the point of view of computer vision, studies concentrated on contrasting new methods concerning improving image quality. For instance, approaches based on paired data focus on producing a brighter image from a low-light one of the same image by working with both inputs simultaneously. On the other hand, systems that only depend on low-light inputs to gain a brighter one consider various characteristics, including colour artefacts and noise. Therefore, these methodologies focus and adopt metrics such as PSNR, SSMI, FSIM, and MSE [81] for measuring image clearance and quality. The higher or lower output obtained by these metrics depending on certain functionality decides the model performance and image quality. However, differing from the typical evaluation of image enhancement approaches, as was pointed out before in this research, the assessment is primary based on state-of-the-art object detection results. In other words, after enhancing an image, the detection determines the model performance compared to the initial low-light image *(before enhancement)*. For this research, the Object Detection Metrics (ODM) tool has been introduced to calculate the Average Precision (AP) and mean Average Precision (mAP) through True Positive (TP) & False Positive (FP) on the proposed data using different enhancer algorithms for accomplishing the evaluation task for both stages [73].

In summary, as discussed previously, calculating mAP, AP, or other metrics is done on the entire dataset for each technique, unlike in the classification task for a single image helping to understand the model's potential individually.

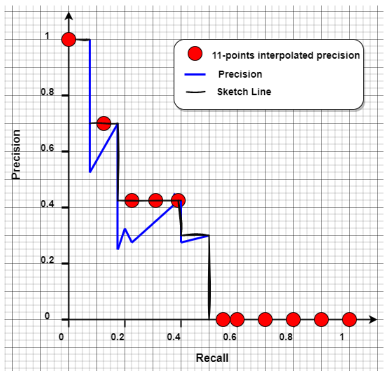
### *Average & mean Average Precision*

The mean Average Precision (mAP) metric is the primary metric for object detection to measure the model performance when finding objects in digital media (e.g. images and videos). Nevertheless, in some circumstances, Average Precision (AP) reveals beneficial information about the detector's behaviour concerning certain classes. The only distinction that can be made between *"mAP"* and *"AP"* is that the first one guarantees the object detection correctness across all classes. In contrast, the other chose to specify detector accuracy for each class independently instead of averaging all classes together and providing a single accurate representation. Moreover, calculating additional metrics is essential in order to acquire the **AP** and **mAP** measures when these metrics are ***Precision*** & ***Recall***. Following that, **TP** and **FP** are also required for the precision and recall estimation.

$$\boldsymbol{Precision} = \frac{TP}{TP+FP} \tag{4.1}$$

$$\boldsymbol{Recall} = \frac{TP}{\text{Num. of ROI in the GT}} \tag{4.2}$$

In the equations 4.1 and 4.2, the true positives and false positives variables indicate a single object's accurate and incorrect detection, respectively. In the event that a single image contains more than one object, then each object is evaluated based on the number of accumulated true and false positives it produces. Moreover, the total of all the annotated objects or bounding boxes in the dataset makes up the denominator in 4.2 when these predictions are referred to as actual predictions also called "ground truth".



| Precision | Recall |
|---|---|
| **1** | 0.0666 |
| 0.5 | 0.0666 |
| **0.5** | 0.1333 |
| 0.4 | 0.1333 |
| 0.25 | 0.4 |
| **0.4** | 0.4 |
| 0.35 | 0.4 |
| ⋮ | ⋮ |

**Figure 4.1** P x R 11-interpolated points curve [73]. **Figure 4.2** Sample values for example purpose.

After getting the precision and recall of a single instance among all the data through calculating *"TPs"* and *"FPs"*, an 11-interpolation precision curve is drawn to choose the most appropriate precision values. As the name suggests, 11-points reflect the maximum precision value of the repetitive recall values; see the values in the Table 4.2 highlighted in grey and light-grey. In other words, at every location where the blue curve reaches its peak, see Figure 4.1. Far from this method, the all-interpolated precision approach is also used whenever all points are utilised in the calculation of the *"AP"* metric in order to produce results that are generally equivalent to the 11-point method [128]. Regarding the table mentioned above, the precision values using the 11-interpolated method are (1, 0.5, & 0.4). Following that, *"AP"* is calculated at every interpolated point, based on a line *(sketch line)* crossing over all the points, creating a bar-chart shape. The following equation represents the formula to calculate *"AP"*.

$$\boldsymbol{AP} = \frac{1}{11} * [1 + 0.0666 + 0.43 + 0.43 + 0.43 + 0 + 0 + 0 + 0 + 0 + 0]$$

As was noted before, *"mAP"* is an extension of *"AP"*. First by determining the average precision per class in the entire dataset, then add up all of these *"APs"* and divided by the total number of classes. The formula of *"mAP"* is defined as:

$$\boldsymbol{mAP} = \frac{AP_{person} + AP_{car} + ...... + AP_{class\_name}}{\text{Total number of classes}}$$

It is worth mentioning that the Intersection Over Union (IOU) plays a crucial part in determining true positives and false positives based on selected threshold predictions that are above or below the threshold, either regarded as accurate or inaccurate to the corresponding ground truth, respectively. Therefore, *"AP"* may be expressed and computed depending on a selected threshold, such as "$AP_{50}$" or "$AP_{70}$" based on a specific value. However, this research considers the conventional and most often used criterion; "$AP_{50}$" when *IOU* is $\geq 0.5$. The above summary concisely describes the methodology for object detection assessment. Indeed, evaluating large-volume datasets consisting of hundreds or thousands of images may require more effort and time. In addition, the presence of several objects in a single image makes comparing actual and predicted instances much harder. Therefore, the Object Detection Metric Tool suggested by [73] has been used for the purpose of this study in order to facilitate the evaluation stage. The tool is simply a software interface that comprises a wide variety of choices and functions, including the following:

1. **Annotations (GT) field**: Require to upload the ground truth files.
2. **Images field**: Require to upload the image files.
3. **Classes field**$^{*}$: Require to upload a *".txt"* file containing all object classes.
4. **Annotations (Pred) field**: Require to upload the predictions produced by the object detection proposed.
5. **Coordinates Format options**$^{*}$: Let the user choose specific annotation formats such as *COCO*, *ImageNet*, *PASCAL_VOC* and *YOLO* in different file extensions like *(.xml, .txt, .csv & .json)* since each format has a particular order and representation of coordinates, class name and confidence.
6. **Metrics options**: Allow the user to choose certain metrics to measure by specifying the IOU, such as the $AP_{50}$ (*IOU*=0.5) and $AP_{70}$ (*IOU*=0.7). In addition, the $AP_{small}$, $AP_{medium}$ & $AP_{large}$ (for only evaluating images with small, medium and large objects, respectively) and other metrics.
7. **Output field**: Require to store results, including Recall and Precision graphs, dominant classes in a bar-chart representation and the mentioned metrics in (6) based on the user choice.

**Table 4.1** Low-Light Image Enhancement Models Metricises.

| Model Name | Time (Avg.) | Small Size | Large Size | Num. of Predictions | Person (AP %) | Car (AP %) | Bus (AP) | Motorcycle (AP %) | mAP (%) | Dataset Class | Num. of GT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **RUAS** | 0.155642109 | (220, 293) | (2906, 4372) | 1836 | 0.561224528 | 0.725613 | 0 | 0.163888889 | 0.483575393 | **Car** | **Car: 1700 Person: 2073** |
| | 0.116654272 | (220, 293) | (3240, 4320) | 2393 | 0.764149502 | 0.614653 | 0.602123802 | 0.068783069 | 0.512427411 | **Person** | |
| **MBLLEN** | 0.66023511 | (220, 293) | (2906, 4372) | 1810 | 0.565170199 | 0.717141 | 0 | 0.178289474 | 0.486866725 | **Car** | |
| | 0.547931034 | (220, 293) | (3240, 4320) | 2321 | 0.762791374 | 0.633442 | 0.705555556 | 0.247863248 | 0.587413037 | **Person** | |
| **DSLR** | 0.115511758 | (256, 128) | (2816, 2176) | 1442 | 0.396873364 | 0.610899 | 0 | 0.1 | 0.369257454 | **Car** | |
| | 0.108595364 | (256, 128) | (3072, 2048) | 1850 | 0.641212207 | 0.481265 | 0.45 | 0.037037037 | 0.402378515 | **Person** | |
| **CSDNet_LOL** | 0.004949153 | (220, 293) | (2906, 4372) | 1622 | 0.503275328 | 0.627325 | 0 | 0.169196429 | 0.433265456 | **Car** | |
| | 0.005011957 | (220, 293) | (3240, 4320) | 2226 | 0.718213306 | 0.563028 | 0.578190045 | 0.244444444 | 0.525968848 | **Person** | |
| **CSDNet_UPE** | 0.004997886 | (220, 293) | (2906, 4372) | 1752 | 0.69593783 | 0.569286 | 0 | 0.154230769 | 0.473151575 | **Car** | |
| | 0.004963689 | (220, 293) | (3240, 4320) | 2339 | 0.744311158 | 0.588396 | 0.685233285 | 0.166666667 | 0.546151881 | **Person** | |
| **CSDGAN** | 0.005002942 | (220, 293) | (2906, 4372) | 1322 | 0.354788032 | 0.512038 | 0 | 0.1 | 0.322275482 | **Car** | |
| | 0.004984538 | (220, 293) | (3240, 4320) | 1569 | 0.52368753 | 0.474782 | 0.486666667 | 0 | 0.371284167 | **Person** | |
| **LiteCSDNet_LOL** | 0.003251195 | (220, 293) | (2906, 4372) | 1654 | 0.504354897 | 0.504355 | 0 | 0.225 | 0.462414332 | **Car** | |
| | 0.003180135 | (220, 293) | (3240, 4320) | 2288 | 0.743488007 | 0.55304 | 0.442424242 | 0.063492063 | 0.450611137 | **Person** | |
| **LiteCSDNet_UPE** | 0.003255874 | (220, 293) | (2906, 4372) | 1752 | 0.559631785 | 0.692967 | 0 | 0.138461538 | 0.463686846 | **Car** | |
| | 0.00350241 | (220, 293) | (3240, 4320) | 2330 | 0.749761812 | 0.585606 | 0.533333333 | 0.185185185 | 0.513471591 | **Person** | |
| **SLiteCSDNet_LOL** | 0.002361181 | (220, 293) | (2906, 4372) | 1612 | 0.466353523 | 0.635202 | 0 | 0.146666667 | 0.416073941 | **Car** | |
| | 0.002368723 | (220, 293) | (3240, 4320) | 2129 | 0.704490473 | 0.547832 | 0.442424242 | 0.301587302 | 0.499083555 | **Person** | |
| **SLiteCSDNet_UPE** | 0.00260198 | (220, 293) | (2906, 4372) | 1763 | 0.562609064 | 0.701017 | 0 | 0.194230769 | 0.485952174 | **Car** | |
| | 0.002595422 | (220, 293) | (3240, 4320) | 2290 | 0.75938246 | 0.566169 | 0.52 | 0.088888889 | 0.483610027 | **Person** | |
| **ElightenGAN** | 6.387862654 | (220, 293) | (2906, 4372) | 1760 | 0.556398886 | 0.687834 | 0 | 0.051538462 | 0.431923678 | **Car** | |
| | 5.642735687 | (220, 293) | (3240, 4320) | 2363 | 0.753130348 | 0.639641 | 0.688888889 | 0.222222222 | 0.575970615 | **Person** | |
| **KinD** | 2.15504834 | (512, 512) | | 1417 | 0.409384109 | 0.665634 | 0 | 0.029375 | 0.368130944 | **Car** | |
| | 2.146603167 | | | 1807 | 0.650444146 | 0.543983 | 0.4 | 0.055555556 | 0.412495778 | **Person** | |
| **RetinexNet** | 7.574797142 | (220, 293) | (2906, 4372) | 1022 | 0.374192587 | 0.428496 | 0 | 0.083333333 | 0.295340715 | **Car** | |
| | 7.129853362 | (220, 293) | (3240, 4320) | 1592 | 0.569940018 | 0.421569 | 0.433333333 | 0.111111111 | 0.383988253 | **Person** | |
| **TBEFN** | 6.252371732 | (220, 293) | (2906, 4372) | 1683 | 0.517205266 | 0.678272 | 0 | 0.115 | 0.436825852 | **Car** | |
| | 5.773925743 | (220, 293) | (3240, 4320) | 2325 | 0.756089557 | 0.610158 | 0.63030303 | 0.206349206 | 0.55072487 | **Person** | |
| **Zero-DCE** | 0.002905036 | (220, 293) | (2906, 4372) | 1673 | 0.535303947 | 0.690317 | 0 | 0.138568723 | 0.45472981 | **Car** | |
| | 0.00227529 | (220, 293) | (3240, 4320) | 2324 | 0.762514419 | 0.624653 | 0.624242424 | 0.240740741 | 0.563037753 | **Person** | |
| **Zero-DCE++** | 0.001261152 | (220, 293) | (2906, 4372) | 1776 | 0.534454591 | 0.687074 | 0 | 0.174444444 | 0.465324245 | **Car** | |
| | 0.001186221 | (220, 293) | (3240, 4320) | 2382 | 0.778788608 | 0.595033 | 0.717016317 | 0.277777778 | 0.592154014 | **Person** | |

**Table 4.2** Baseline Metricises (Before Enhancement).

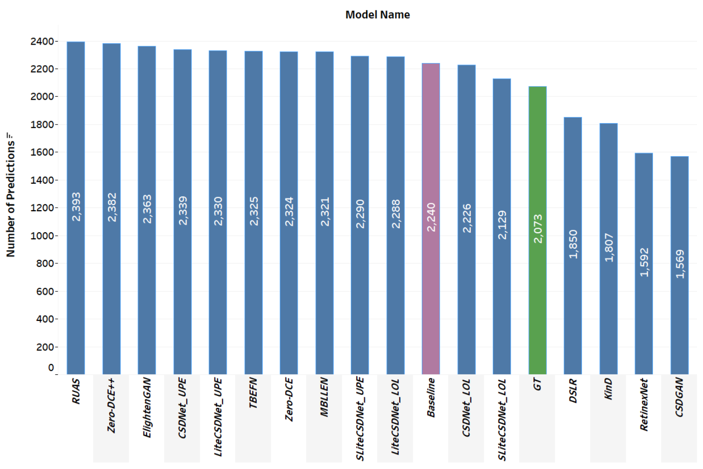
| Model Name | Time (Avg.) | Size (Smaller) | Size (Larger) | Num. of Predictions | Person (AP %) | Car (AP %) | Bus (AP %) | Motorcycle (AP %) | mAP (%) | Dataset Class | Num. of GT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Baseline** | 0.118526646 | (220, 293) | (2906, 4372) | 1704 | 0.577746343 | 0.725887436 | 0 | 0.066 | 0.45676682 | Car | 1700 |
| | 0.119407895 | (220, 293) | (3240, 4320) | 2240 | 0.750083478 | 0.593559626 | 0 | 0.0277 | 0.34285522 | Person | 2073 |

The metrics and coordinates style chosen for the current study are the following ones: *IOU@0.5*, $AP_{PerClass}$; {Person, Car, Motorcycle & Bus}, *mAP* and *YOLO*-Normalized coordinates which are sorted as *<class_id>, <confidence> & <x_center> <y_center> <width> <height>. * indicates same file or format used for both ground truth and predictions.*

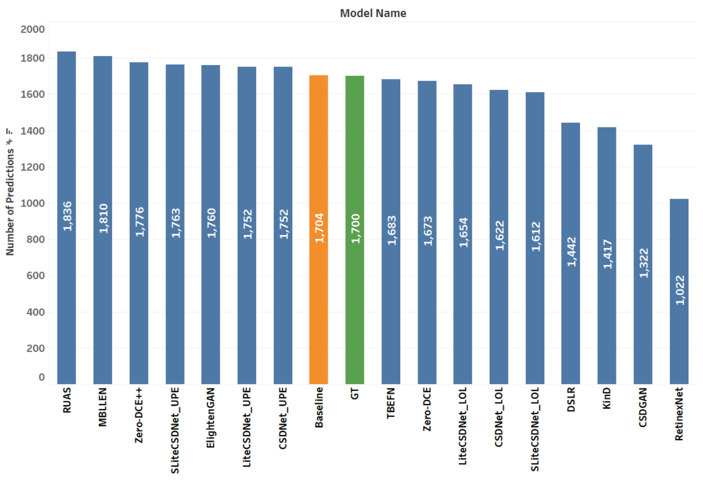
The assessment of the low-light image enhancement models based on the previously suggested approaches for the object identification algorithm is shown in Table 4.1. The dataset contains images captured through multiple devices, such as sensor cameras and smartphones, to name a few; thus, an average of all the timings was computed to evaluate how well the model performed on various input scales and sizes represented in the attribute *"Time (Avg.)"*. In addition, it is worth mentioning that only the time variable belongs to the enhancement model, whereas the remaining are calculated based on the detection outputs. Regarding image sizes, maximum and minimum *"Width"* (W) and *"Height"* (H) are also considered when input dimensions vary in that range. Unlike the classification stage, which resizes the inputs to *(128 x 128)* pixels to accelerate the process of extracting features and predicting the appropriate label, the original dimensions are passed to the subsequent tasks after the classification task takes place.

Additionally, identical metrics for the baseline model (*Directly applied to images before enhancement*) used for the detection part "*Detectron2*" are shown in Table 4.2. The time variable is the only change which more accurately indicates the acceleration of the detection process than the enhancement. As mentioned in previous chapters, the 5G Wales Unlocked project focused on identifying two types of instances; "*People*" & "*Car*". However, both tables displayed extra predicted classes such as "*Bus*" and "*Motorcycle*" since images were shot in random street areas, which may include various instances. In addition, by examining both tables, it is evident that in the majority of the cases, whether before or after enhancing, the detector performs poorly with low accuracy for the "*Bus*" class with an *"AP"* equal to or nearly equal zero and in some cases for "*Motorcycle*" class. Thus, it can be deduced that only a tiny portion of these classes contributed to the "*People*" and "*Car*" datasets since the ExDark dataset comprises several datasets per class. Moreover, the total Region of interests (ROIs) predicted are presented in Figures 4.3a & 4.3b for the "*People*" and "*Car*" datasets, respectively, when the detection task is applied directly to the original data and after enhancing through the mentioned enhancement approaches.

Additionally, the actual ROIs supplied by the dataset are represented as the ground truth. For example, figure 4.3a demonstrates that the detector is able to distinguish more items as the ground truth with the help of numerous tested models in comparison to the baseline model. In a similar manner, specific models perform better than the actual predictions when applied to the *"Car"* dataset 4.3b. On the other hand, these extra detections might represent false predictions or accurate instances that the annotators probably excluded during the annotation process. However, the *"mAP"* metric is calculated in order to confirm the correct ROIs to the corresponding ones in the ground truth. This is because the process and concept of evaluation only compare the position and size of anticipated ROIs to those already present in the ground truth. Therefore, incorrect predictions are believed to be extra or missing, affecting the model's accuracy.



**(a)** The number of ROIs for "People" dataset.



**(b)** The number of ROIs for "Car" dataset.

**Figure 4.3** The number of predictions (Before Enhancement vs After Enhancement vs Ground truth).

The following tables present the outcomes, encompassing a comprehensive overview of diverse model metrics. Nevertheless, certain models exhibit sub-optimal performance in specific areas. Thus, the next set of seven tables individually showcases the exemplary model or models for distinct aspects, aiding in the comparative analysis of the models.

❊ ***Speed of Processing***

After averaging the "*Time*", which represents the required time to enhance a single image despite size and scale, table 4.3 shows the slowest and fastest model in speed, including the top one highlighted in green and best second approach per dataset class.

**Table 4.3** The fastest and slowest enhancement model (in sec).

| Car | | Person | |
|---|---|---|---|
| **Slow** | **Fast** | **Slow** | **Fast** |
| **RetinexNet:** 7.574797142 | Zero-DCE++: **0.001261152** | **RetinexNet:** 7.129853362 | Zero-DCE++: **0.001186221** |
| **ElightenGAN:** 6.3878626564 | SLiteCSDNet_LOL: 0.002361181 | **TBEFN:** 5.773925743 | **Zero_DCE:** 0.00227529 |

❊ ***mean Average Precision (mAP)***

The *"mAP"* represents the overall accuracy of the state-of-art object detection algorithm on images after the enhancing stage. Different models obtained higher accuracy and performed better than the baseline, as illustrated in Table 4.4.

**Table 4.4** The high and low *"mAP"* model (in %).

| Car | | Person | |
|---|---|---|---|
| **Low** | **High** | **Low** | **High** |
| **RetinexNet:** 0.295340715 | MBLLEN: **0.486866725** | **Baseline:** 0.34285522 | Zero-DCE++: **0.592154014** |
| **CSDGAN:** 0.322275482 | **SLiteCSDNet_UPE:** 0.485952174 | **CSDGAN:** 0.371284167 | **MBLLEN:** 0.587413037 |

❊ ***Average Precision Per Class "AP"***

The following table(s) dedicate the model accuracy for a specific class, "*Person, Car, Bus & Motorcycle*" respectively, various models contribute by outperforming in a particular object class than the others. However, the "*Zero-DCE++*" is still dominant in the majority of classes.

**Table 4.5** The high/low *"AP"* model for *"Person"* class (in %).

| Car | | Person | |
|---|---|---|---|
| Low | High | Low | High |
| **CSDGAN:** 0.354788032 | **CSDNet_UPE: 0.69593783** | **CSDGAN:** 0.52368753 | **Zero-DCE++: 0.778788608** |
| **RetinexNet:** 0.374192587 | **Baseline:** 0.577746343 | **RetinexNet:** 0.569940018 | **RUAS:** 0.764149502 |

**Table 4.6** The high/low *"AP"* model for *"Car"* class (in %).

| Car | | Person | |
|---|---|---|---|
| Low | High | Low | High |
| **RetinexNet:** 0.428496223 | **RUAS:** 0.725612762 | **RetinexNet:** 0.421568551 | **ElightenGAN: 0.639641001** |
| **LiteCSDNet_LOL:** 0.504354897 | | **CSDGAN:** 0.474782473 | **MBLLEN:** 0.633441969 |

**Table 4.7** The high/low *"AP"* model for *"Bus"* class (in %).

| Car | | Person | |
|---|---|---|---|
| Low | High | Low | High |
| **All Models:** 0 | **All Models:** 0 | **Baseline:** 0 | **Zero-DCE++: 0.717016317** |

**Table 4.8** The high/low *"AP"* model for *"Motorcycle"* class (in %).

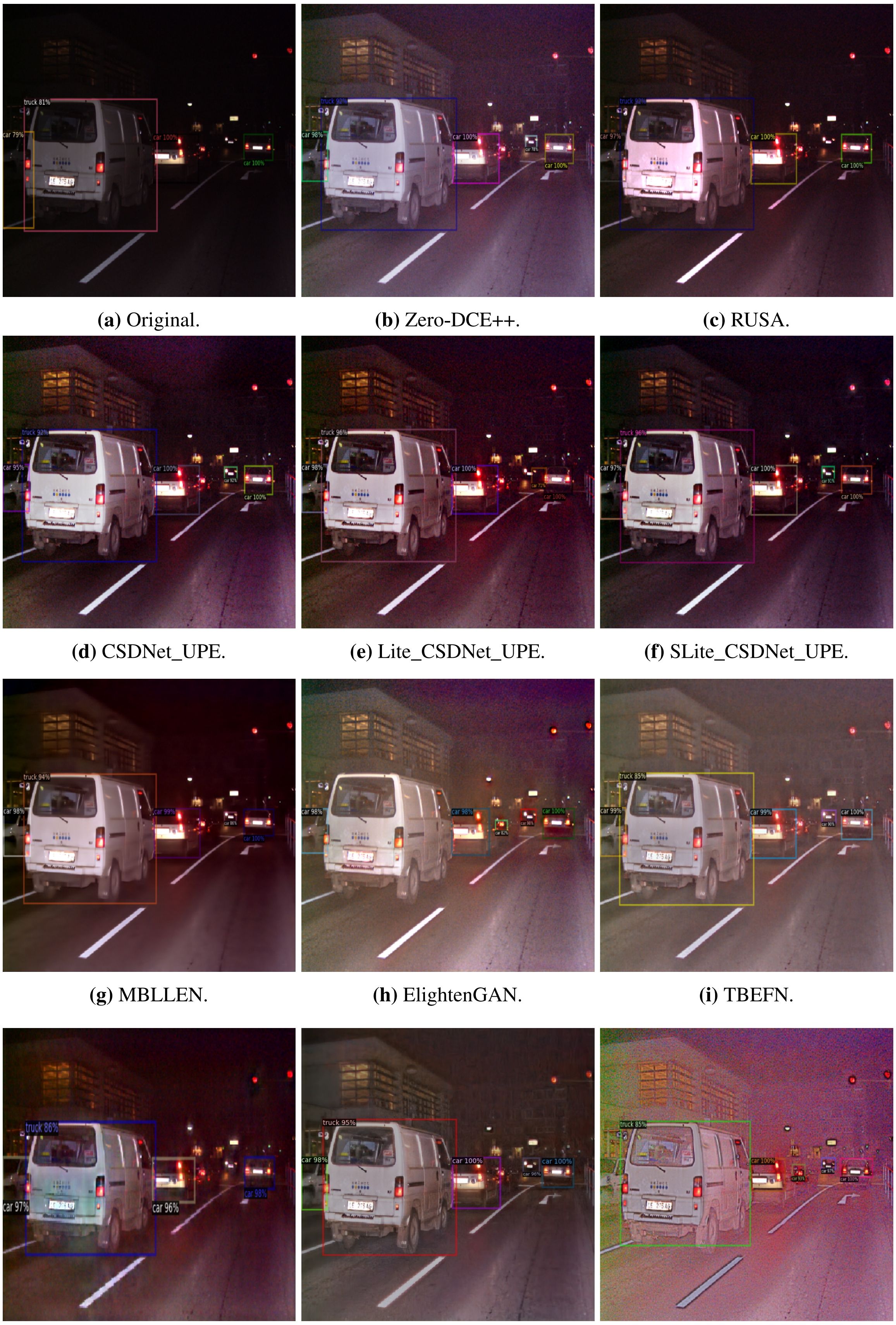
| Car | | Person | |
|---|---|---|---|
| Low | High | Low | High |
| **KinD:** 0.029375 | **LiteCSDNet_LOL: 0.225** | **CSDGAN:** 0 | **Zero-DCE++: 0.27777778** |
| **ElightenGAN:** 0.051538462 | **SLiteCSDNet_UPE:** 0.194230769 | **Baseline:** 0.027777778 | MBLLEN: 0.24786 |

❊ ***Number of ROIs***

Finally, the number of predictions in the Table below 4.9, made by the ideal models after enhancement. It is easy to notice that the baseline is not perfect and is not even included on the list of predictions with the lowest accuracy.

**Table 4.9** The high/low number of predictions.

| Car | | Person | |
|---|---|---|---|
| Low | High | Low | High |
| **RetinexNet:** 1022 | **RUAS: 1836** | **CSDGAN:** 1569 | **RUAS: 2393** |
| **CSDGAN:** 1322 | **MBLLEN:** 1810 | **RetinexNet:** 1592 | **Zero-DCE++:** 2382 |



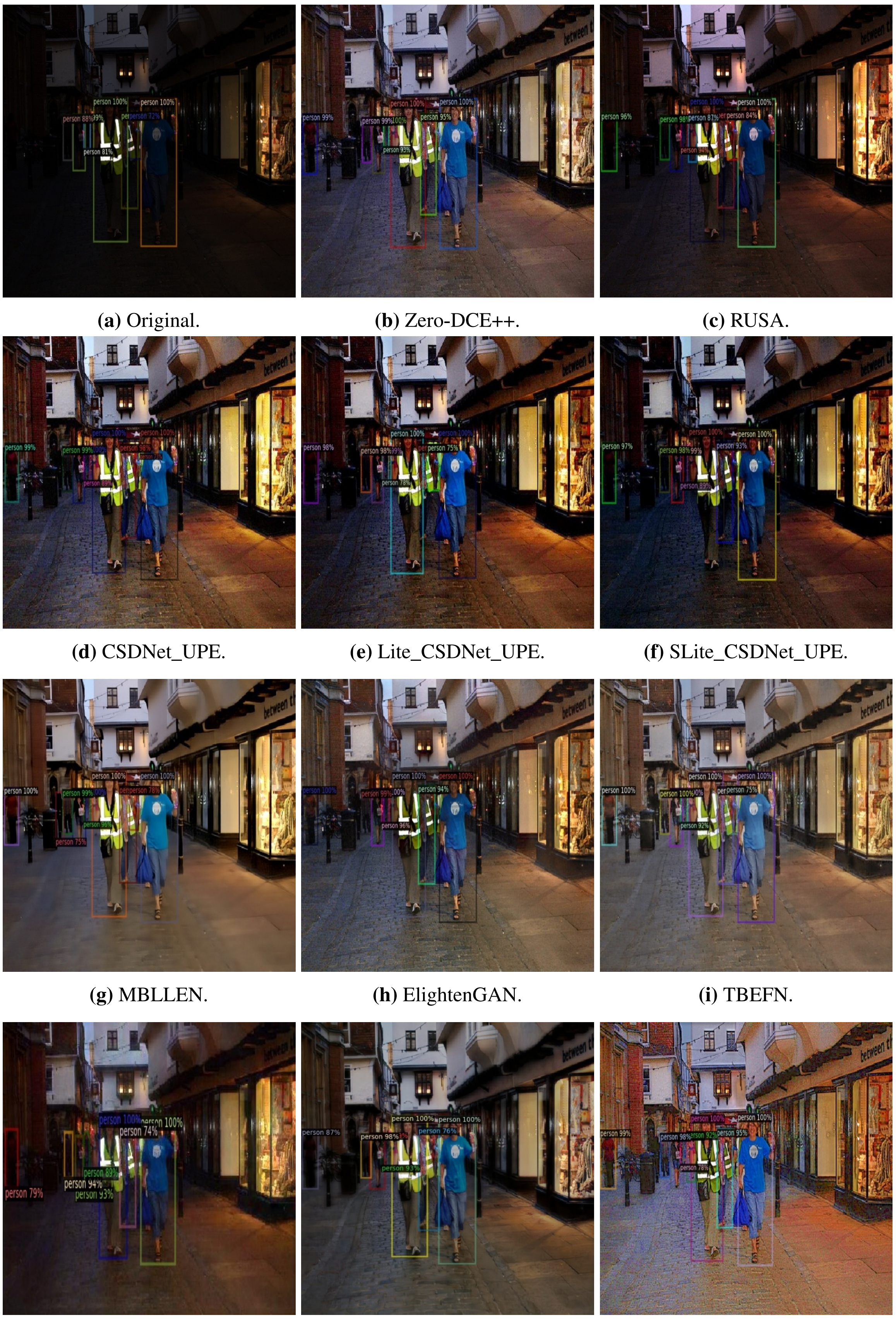
**(a)** Original. **(b)** Zero-DCE++. **(c)** RUSA.

**(d)** CSDNet_UPE. **(e)** Lite_CSDNet_UPE. **(f)** SLite_CSDNet_UPE.

**(g)** MBLLEN. **(h)** ElightenGAN. **(i)** TBEFN.

**(j)** DSLR. **(k)** KinD. **(l)** RetinexNet.

**Figure 4.4** Comparison of model enhancement for the detection stage on "*Car*" dataset, Sample "2015_02902.png".



**(a)** Original. **(b)** Zero-DCE++. **(c)** RUSA.

**(d)** CSDNet_UPE. **(e)** Lite_CSDNet_UPE. **(f)** SLite_CSDNet_UPE.

**(g)** MBLLEN. **(h)** ElightenGAN. **(i)** TBEFN.

**(j)** DSLR. **(k)** KinD. **(l)** RetinexNet.

**Figure 4.5** Comparison of model enhancement for the detection stage on "*Person*" dataset, Sample "2015_06337.jpg".

**Table 4.10** Predictions results of Figures 4.4 & 4.5, ***Green***: outperformed the ground truth and ***Blue***: outperformed the baseline and ground truth.

**(a)** Sample *"2015_02902.png"* form *"Car"* dataset.

| **Predictions from** | **Num. of Preds.** |
|---|---|
| **GT** | **6** |
| **Baseline** | **4** |
| CSDNet_UPE | **5** |
| DSLR | 4 |
| ElightenGAN | **5** |
| KinD | **5** |
| Lite_CSDNet_UPE | **5** |
| MBLLEN | **5** |
| RetinexNet | **5** |
| RUAS | 4 |
| SLite_CSDNet_UPE | **5** |
| TBFEN | **5** |
| Zero-DCE++ | **5** |

**(b)** Sample *"2015_06337.jpg"* from *"Person"* dataset.

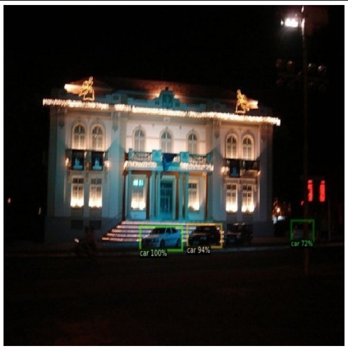
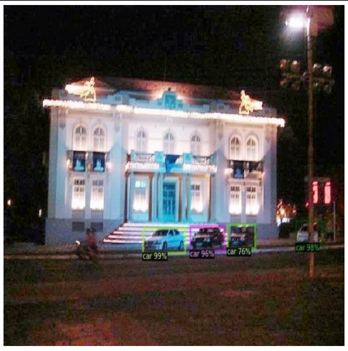
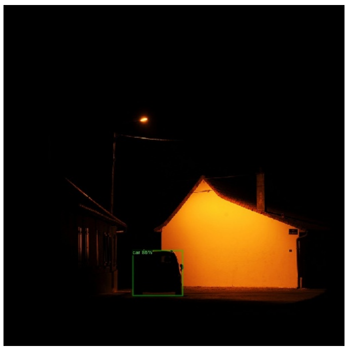
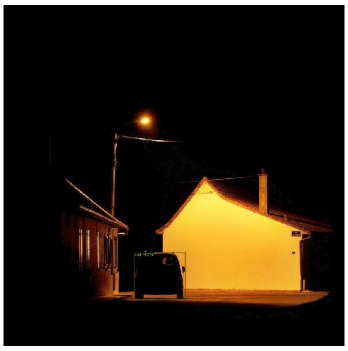
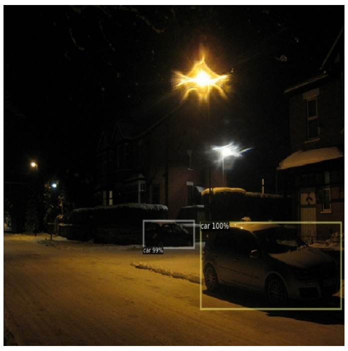
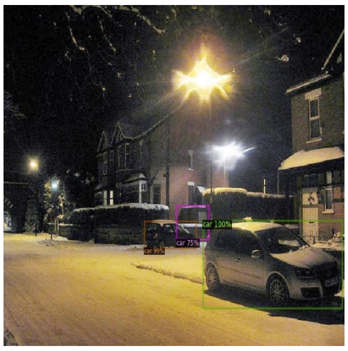
| **Predictions from** | **Num. of Preds.** |
|---|---|
| **GT** | **7** |
| **Baseline** | **7** |
| CSDNet_UPE | **8** |
| DSLR | 7 |
| ElightenGAN | 7 |
| KinD | 7 |
| Lite_CSDNet_UPE | **8** |
| MBLLEN | **9** |
| RetinexNet | **8** |
| RUAS | **9** |
| SLite_CSDNet_UPE | 7 |
| TBFEN | **8** |
| Zero-DCE++ | 7 |

Consequently, Figures 4.4 and 4.5 illustrate the various detection outputs after each enhancement step's running. For the sake of making an accurate comparison between the enhancement models for both datasets "*Car*" & "*Person*", the same sample was selected and kept the original image size and their results are shown in the Tables 4.10a & 4.10b. The highlighted rows correspond to actual predictions *"GT"* and a direct detection without undergoing the enhancement stage *"Baseline"*. It can be noticed that most approaches can detect all instances except one missing object compared to the ground truth detections. In addition, these models performed well and surpassed the baseline, which achieved only 4-detection out of 6 for the sample *"2015_02902.png"*. On the other hand, there was a dramatic increase in recognizing objects for *"2015_06337.jpg"* when nearly all techniques achieved more detections than the ground truth. For instance, after investigating the output by the *"MBLLEN"* technique, it was discovered that more (people) were covered by dark pixels, missed or excluded from the ground truth. In addition to *"MBLLEN"*, other models also achieved higher detection than the ones with *"8"*. The rows that have been highlighted are those that relate to actual predictions *"GT"* and direct detection without undergo through the enhancement step *"Baseline"*. Compared to the detections made using the ground truth, most methods can identify all instances except one missing item. In addition, these models performed exceptionally well and outperformed the baseline, which only detected four out of six instances of the sample *"2015 02902.png"* compared to the models' results. On the other hand, the ability to recognize objects for the file *"2015 06337.jpg"* was significantly improved when almost all of the methods obtained more detections than the ground truth.

For example, after investigating the results produced by the *"MBLLEN"* approach, it was found that a more significant number of (people) were obscured by dark pixels, ignored, or left out of the ground truth altogether. In addition to *"MBLLEN"*, other models also achieved greater detection than those with 8-predictions. It is worth mentioning that the enhancement stage improved in detecting more objects and exploring new objects covered by dark. However, there are situations in which using these approaches has no impact or cannot identify new objects owing to a wide variety of circumstances including, but not limited to, image quality, object dimensions and sizes, incident light on objects, and a few more. Thus for further demonstration, outputs in the Table of figures 4.11 show sample results using "*Zero-DCE++*". The results are specified as the following conditions: "*Better*", "*Same*" and "*Worst*" detection.

In conclusion, this section helped to rapidly identify and locate the best solutions for the low-light image enhancement task. It was owing to the fact that the assessment was carried out on the whole dataset as opposed to the image level as in the classification task described in earlier parts. In addition, testing a detection technique on a large dataset after enhancement has been introduced helps limit what models have the best and worst performance. Therefore, in the following sections, only best practises techniques were studied and assessed to save time and prevent any overload on the resource-constrained devices. In addition, the methodologies were selected based on the investigated metrics that were discussed while considering the variables that are compatible with the devices with limited resources.

**Table 4.11** Sample outputs in terms of "***Better***": outperformed after enhancing, "***Same***": no effect with enhancing & "***Worst***": perform poorly after enhancing.

| Cond. | Original | Before IE | After IE |
|---|---|---|---|
| **Better** | |  |  |
| **Same** | |  |  |
| **Worst** | |  |  |

## 4.2 Lightweight Dynamic Classification

In this section, we will delve into the classifier outcomes and assessment, commencing with data pre-processing, feature extraction, and training the proposed Random Forest classifier.

The dynamic classifier acquires attributes and assigns images to the optimum enhancement model category, contingent upon the low-light input characteristics. In simpler terms, images with inadequate illumination are classified to be enhanced by the optimal enhancement network.

### 4.2.1 Pre-Processing

Before progressing to feature extraction and model training, this section encompasses several stages for data preparation. Firstly, the entire dataset was utilized for constructing the classifier, unlike the previous stages of enhancement and detection, where only the person and car sub-datasets were employed for evaluation, saving time in exploring the optimal techniques.

Drawing upon the prior examination of object detection alongside enhancement techniques, exclusively the most optimal models were selected to construct the classifier. These models exhibit diversity in speed, image quality, average precision (AP), and mean average precision (mAP) accuracy. Each model undertook the enhancement of the complete dataset alongside object detection. Subsequently, inputs displaying an mAP $\geq 0.9$ concordance between prediction and ground truth were exclusively considered and relocated to distinct directories, effectively representing unique images embodying a specific technique. The workflow exemplified in Figure 4.6 illustrates the methodology employed for acquiring these distinct samples. Moreover, Table 4.12 provides an overview of the number of distinct images corresponding to each technique after removing duplicates. It is evident that the number of acquired unique samples is relatively modest in comparison to the overall size of the dataset.

**Table 4.12** Unique samples with mAP $\geq 0.9$ after applying image enhancement techniques and detection on the whole 7k ExDark dataset.

| Model Name | Number of unique images | Time (sec) |
|---|---|---|
| CSDNet_UPE | 33 | 0.005 |
| LiteCSDNet_UPE | 23 | 0.0035 |
| LiteCSDNet_LOL | 52 | 0.0032 |
| **RUAS** | **113** | **0.12** |
| SLiteCSDNet_UPE | 59 | 0.002 |
| **Zero_DCE++** | **102** | **0.0012** |

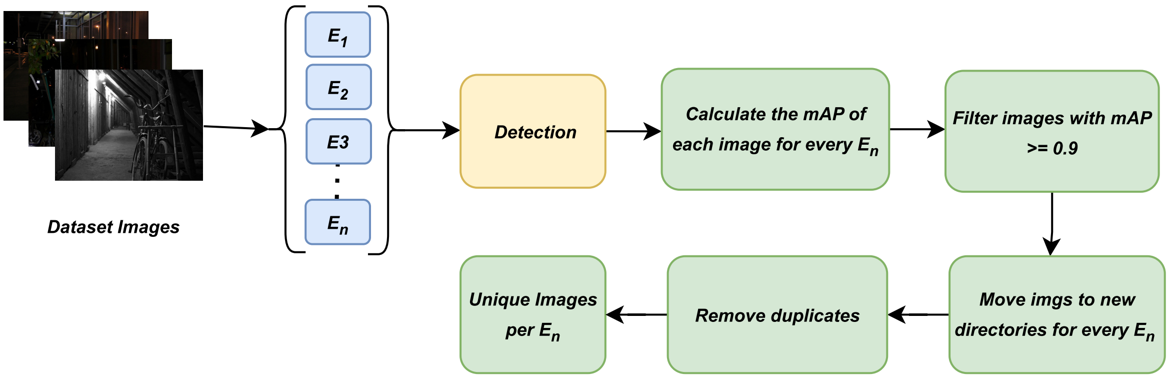

Dataset Images
$E_1$
$E_2$
E3
$E_n$
Detection
Calculate the mAP of each image for every $E_n$
Filter images with mAP >= 0.9
Move imgs to new directories for every $E_n$
Remove duplicates
Unique Images per $E_n$

**Figure 4.6** Unique images per enhancement technique, where each technique is represented as $E_1$, $E_2$,...$E_N$.

Henceforth, solely the *RUAS* and *Zero-DCE++* techniques were deemed eligible for inclusion in the classifier development process representing labels for low-light inputs. This decision was based on their exceptional achievement in yielding the most significant number of unique images and their commendable performance across various aspects. On the other hand, samples from Berkeley, Stanford, MSCOCO, and the 5G Wales Unlocked datasets were utilized as representatives of bright scenes. Scenes were systematically categorized and organized based on predefined criteria and assumptions to determine whether they belonged to the bright or dark category. For example, images captured under daylight conditions with ample illumination were classified as bright, while those taken during dawn, dusk, and nighttime were classified as low-light scenarios.

### 4.2.2 Features Extraction and Random Forest

As mentioned before, the *"RUAS"* and *"Zero-DCE++"* strategies were chosen to contribute to the system that was implemented at the edge for the enhancement phase and classification phase based on the unique samples obtained. These techniques are represented by the labels $E_1$ and $E_2$, respectively. In addition, the label $E_0$ indicates that no enhancement is required for inputs deemed to be bright.

During the extraction phase, multiple filters were implemented; however, not all contributed to the model's ability to produce an acceptable prediction during the training phase. Further details can be found in the following kernels: (1) The Gabor filter bank, (2) The

Sobel, Scharr, Laplacian and Prewitt's operators for edge detection, (3) The Gaussian blur, (4) The Median filtering, (5) The Variance filter, (6) Sharpen filter, and, finally, the most critical feature, (7) Original pixels. It should be noted that while many filters apply to data sources that use RGB and grayscale colour space, only filters applicable to RGB colour were introduced and applied to the collected data, as RGB provides more information and attributes for describing the entire image in different aspects. For example, grayscale representation, ranging from 0 to 255, only describes areas through black, white, or in-between pixel colours. A Gabor filter was defined using a range of values, such as $\sigma$ is equal to 1 & 3, $\theta$ equal to 0 & 0.785, $\gamma$ to 0.05 & 0.5, $\lambda$ to 0 & 0.0785 & 1.57 & 2.356, Ksize equal to 9 and $\phi$ to 1, allowing the production of a unique Gabor kernel (e.g. $G1, G2, ...GN$). In contrast, the Sobel operator was utilised in its default configuration without any adjustments except for extracting original pixel values at the beginning of each test. These convolution kernels were then applied to the images. After feature extraction, the resulting attributes were reshaped into a 1D-vector and introduced into the classifier for training and predicting on the test set, see Figure 4.7.

The data was split into 90% and 10% for training and testing sets, respectively. For faster processing, inputs are resized into 128x128. Moreover, the number of trees in the random forest classifier was set to *n_estimators=48*, which was determined as the optimal value using the *GridSearchCV* technique for hyperparameters with a *random_state=42* for all trials. Figure 4.8 illustrates that the mean test score of 93% can be attained by applying 48 trees.

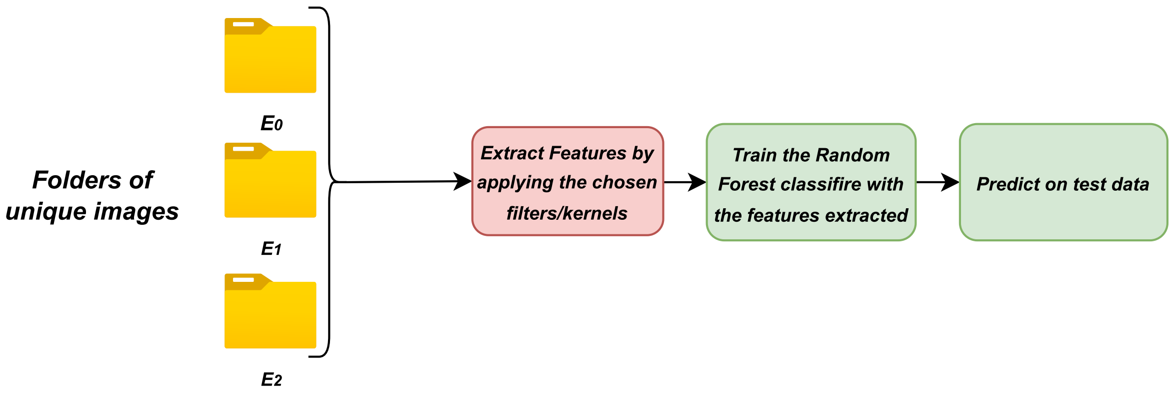

Folders of unique images
E0
E1
E2
Extract Features by applying the chosen filters/kernels
Train the Random Forest classifire with the features extracted
Predict on test data

**Figure 4.7** Unique images per enhancement technique, where each technique is represented as $E_1$, $E_2$,...$E_N$.

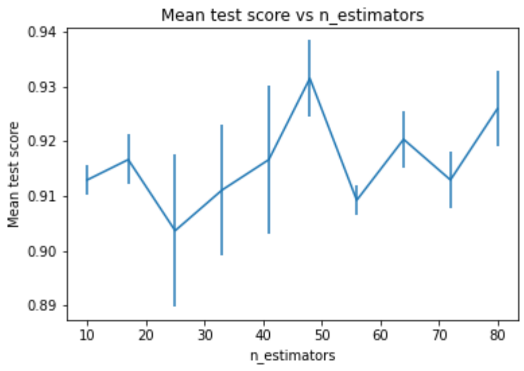


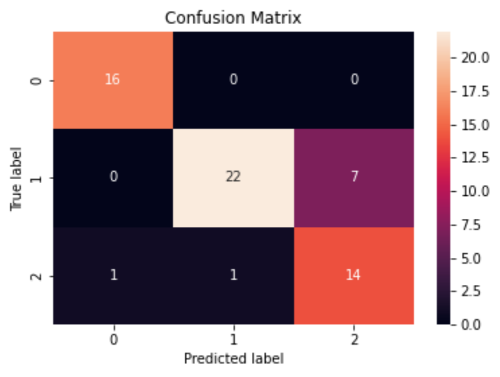
**Figure 4.8** The plot represents the n_estimators in a range of 10 to 80 with the mean test scores. It can be noted that the best score is achieved when trees=48, obtaining a score of 93%.

Table 4.13 displays the top-5 accuracies, where more than 20 trials were conducted, including traditional machine learning and modern approaches. Feature creation was achieved by arbitrarily utilizing an assortment of filters, applying up-sampling for imbalanced data and finding the optimal values using hyperparameters. It is worth mentioning that upsampling was introduced to evaluate the classifier's accuracy with more sample data, which showed a dramatic increase in accuracy. The accuracy was calculated using the "*metrics.accuracy_score*" function, which compares the predicted labels on the test set with the actual labels. As a result, when combining the Gabor filter with pixel values and employing only the Sobel operator. Experiment (5) outperformed the ones that relied on producing high-order kernels for the Gabor filter and modern pre-trained weights Convolutional Neural Networks (e.g. VGG-16) as a feature extractor. These features were reshaped into a one-dimensional vector and were generated by applying six kernels with different values. The algorithm can complete the extraction and prediction phases on a single input with a random size within *0.2 msec* on a Raspberry Pi, adhering to real-time processing requirements with a low response time at the edge. In addition, the accuracy of data validation on the test set is 85.24%, suggesting that an appropriate technique for the enhancing phase may distinguish certain dark features and differentiate dark from bright scenes. However, it should be noted that this conclusion was reached through a single experiment on a small portion of the data. The confusion matrix depicted in Figure 4.9a indicates the classifier's ability to differentiate between bright and low-light images and between the two chosen techniques.

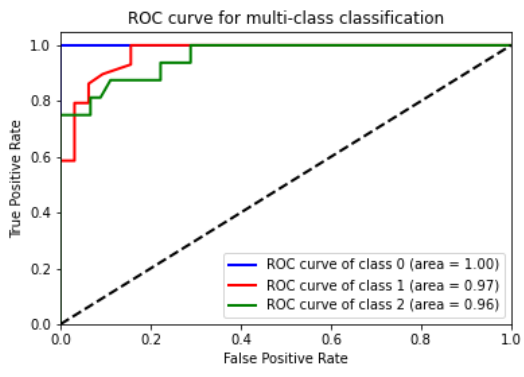
Table 4.13 Details of the conducted experiments with different convolution kernels and VGG-16 CNN (as a feature extractor), where **OP**, **GB**, **US** and **HP** stand for Original Pixels, Gabor Bank Filter, Up-Sampling and Hyperparameters, respectively.

| Exp ID | Filters/ Model | US | HP | Accuracy |
|---|---|---|---|---|
| 1 | OP, GBF and Sobel | ✗ | ✓ | 85% |
| 2 | OP, GBF and Sobel | ✓ | ✗ | 81% |
| 3 | VGG-16 | ✓ | ✗ | 75% |
| 4 | VGG-16 | ✓ | ✓ | 78% |
| 5 | **OP, GBF and Sobel** | ✓ | ✓ | **85.24%** |

Nevertheless, the model encounters challenges in discriminating images exhibiting *RUAS* or *Zero-DCE++* characteristics. Moreover, Figure 4.9b displays the Receiver Operating Characteristic (ROC) curve mainly used to evaluate the performance of binary classification models. Moreover, the ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different threshold values for classification. The TPR represents the proportion of valid positive instances correctly identified as positive. In contrast, the FPR represents the proportion of valid negative instances incorrectly classified as positive [21]. The classifier has a ROC accuracy of 96.6%, indicating that the model can discriminate between positive and negative instances, which implies that the classifier can correctly classify positive instances while minimizing false positives.



(a)



(b)

**Figure 4.9** (a) Confusion matrix, "0" for *Bright*, "1" for the *RUAS* model and "2" for the *Zero-DCE++* model. (b) ROC curve for the multi-classification.

**Table 4.14** Samples results from test set with actual and predicted labels ("**A**" stands for **Actual** and "**P**" for **Predicted** label).

| **Index** | **A** | **B** | **C** |
|---|---|---|---|
| **1** | **A:** $E_0$ **P**: $E_0$ | **A:** $E_0$ **P**: $E_0$ | **A:** $E_1$ **P**: $E_1$ |
| **2** | **A:** $E_1$ **P**: $E_1$ | **A:** $E_2$ **P**:$E_2$ | **A:** $E_2$ **P**: $E_2$ |
| **3** | **A:** $E_1$ **P**: $E_2$ | **A:** $E_2$ **P**: $E_1$ | **A:** $E_2$ **P**: $E_1$ |

As a direct consequence, random samples were chosen from the validation data and shown in the Table of figures 4.14 with their actual labels and predicted labels by the proposed classifier. The classifier successfully determined the appropriate labels in the indices (1 & 2). As previously said, "$E_0$" scenes such as (1A & 1B) are acquired when daylight is still and bright pixels dominate. When at least a few items that can be identified by human vision or computer machines, like the ones in (1C & 2A), are characterized as being dark or partly dark. In addition, the photos (2B & 2C) have a dark appearance. On the other hand, the findings in the third index show that incorrect predictions may be made when inputs represented as "$E_1$" are interpreted as "$E_2$" or vice versa.

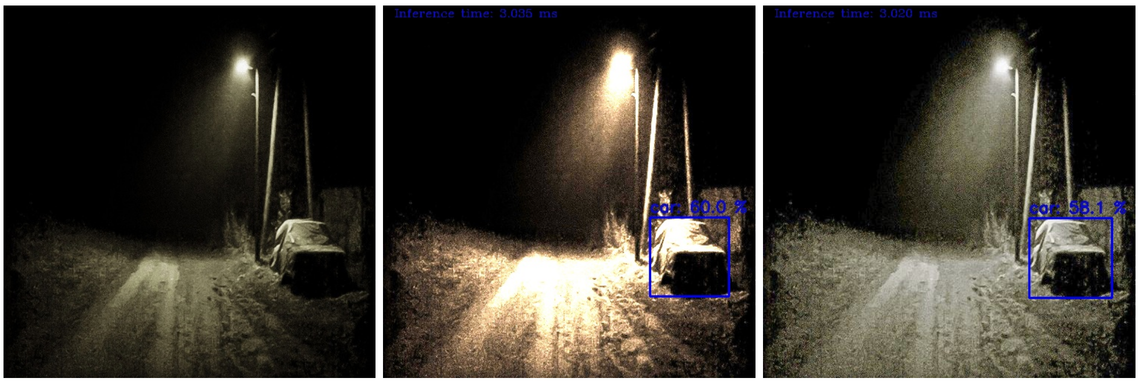
## 4.3 Edge Environment

This section presents the best practice enhancement models through their paces and assesses them for use in an edge environment. The models have been selected based on the findings of benchmarking performed as well as the classifier creation described in the previous section. The performance of the large model *"Detectron2"* on both paradigms is shown in Table 4.15. Both datasets show that accuracy after the enhancement model has been applied improved and remains unaffected even on the edge paradigm. The only distinction is the time required to detect instances on a single image.

**Table 4.15** Evaluation of the large model "Detectron2" on the Cloud and constrained device, **w**: with enhancement & **wo**: without enhancement.

| System | mAP % | AP (Person) % | AP (Car) % | Speed (in sec) | Size |
|---|---|---|---|---|---|
| Cloud (**wo**) | 0.42103767 | 0.625056609 | 0.638056402 | **0.09** | 500 x 500 |
| Cloud (**w**) | **0.425116397** | 0.573534606 | **0.701814585** | | |
| Edge (**wo**) | 0.42103767 | 0.625056609 | 0.638056402 | **4.4** | 500 x 500 |
| Edge (**w**) | **0.425116397** | 0.573534606 | **0.701814585** | | |

Consequently, "*Yolov5-tiny*" object detection was used on edge in conjunction with the enhancement models for the suggested design, due to the device's availability. The test bed consisted of 40-images taken randomly from both datasets of various brightness levels (e.g. different low-light & bright) with only "*People*" and "*Car*" instances. In addition, images were scaled down to 500 by 500 pixels in both paradigms for fair comparison and fast processing. Moreover, the evaluation of the tiny detector was carried out using the same evaluation methods of the large model. The first row highlighted in Table 4.16 expresses metrics of a straightforward inference of the detector with the original data without enhancement. As an example, "*RUAS (UPE)*" for the enhancement stage boots each of *"mAP"* and *"AP"* for both classes in contrast to the baseline model. In addition, the *"ElightenGAN"* produces more accurate findings regarding the *"Car"* category, although it takes a very long time to process. On other hand, the remain models perform poorly in comparison to "*Yolo-tiny*". Consequently, "*Zero-DCE++*" & "*RUSA*" have been selected in order to contribute to the edge design implementation based on the assessment results on the cloud environment; as was stated in section [3.4]. Additionally, a sample output comparing results before and after enhancement models are applied shown in Figure 4.10.



**(a)** Yolo (**Only**). **(b)** RUAS **with** Yolo **(c)** Zero-DCE++ **with** Yolo.

**Figure 4.10** Detection outputs on the resource-constrained devices using Yolov5-tiny.

Further, the computational resources metricise of each device, also known as *"Node"* or *"Edge Server"* contributed to the proposed design, were considered throughout this investigation. For experimentation, three images per category; bright as ($E_0$) and dark as ($E_1$) and ($E_2$) were carried out to evaluate device computation for a unique scenario. Table 4.17 shows different measurements simultaneously by running an input image for each class when resources such as RAM usage, Temperature readings, CPU & GPU usages are also gauged and considered for each test. For instance, the ideal mode indicates device stability, and no processing occurs across all nodes. On the other hand, a bright image was used as the input for "*Test 1*". Thus, since no processing is needed, the detection job is immediately executed on the same node, and it is unnecessary to be forwarded to surrounding nodes for enhancement. In addition, it is essential to note that the CPU utilisation and temperature of "*Node 3*" rise due to the execution of the classification and detection tasks only. Following the same principle, "*Test 2 & Test 3*" with dark images as inputs that required to have their brightness improved through "*Node 1 & Node 2*" prior to the recognition task. Similarly to "*Node 3*", resources increased dramatically since classification and detection are involved, in addition to the GPU utilisation for the "Jetsons" devices *(e.g. 99 % of usage)*, which these enhancement models rely upon during processing.

**Table 4.16** Yolo-tiny object detection with/without enhancement metrics on constrained devices.

| Model Name | Time inference (in sec) | mAP % | AP (Person) % | AP (Car) % |
|---|---|---|---|---|
| **Yolo-tiny** | **0.003** | **0.298226188** | **0.340724316** | **0.553954248** |
| **RUAS (UPE)** | **0.03** | **0.333046362** | **0.387374462** | **0.611764624** |
| **RUAS (Dark)** | 0.055 | 0.243440646 | 0.268292683 | 0.462029256 |
| **Zero-DCE++** | **0.001** | 0.257393576 | 0.302882483 | 0.469298246 |
| **Zero-DCE** | 0.0067 | 0.245649369 | 0.195121951 | 0.541826156 |
| **SLiteCSDNet_UPE** | 0.025 | 0.275467783 | 0.288930582 | 0.537472767 |
| **SLiteCSDNet_LOL** | 0.025 | 0.241038582 | 0.189701897 | 0.533413849 |
| **CSDNet_UPE** | 0.05 | 0.251191625 | 0.241685144 | 0.511889731 |
| **ElightenGAN** | 9.81 | 0.280634461 | 0.285178236 | 0.556725146 |

**Table 4.17** Nodes computational resource metrics, where **N/A**: Not applicable, **N/U**: Not used in processing, **C**: Classification task, **OD**: Object Detection task, **IE**: Image Enhancement task, **Node 1**: Enhancement model (1), **Node 2**: Enhancement model (2) & **RPi**: Raspberry Pi.

| Test ID | CPU Usage | | | GPU Usage | | RAM Usage | | | Temp Usage | | | | | Task involved? | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RPi (Node 3) | Jetsons | | Jetsons | | RPi (Node 3) | Jetsons | | RPi (Node 3) | Jetsons | | | | RPi (Node 3) | Jetsons | |
| | | Node 1 | Node 2 | Node 1 | Node 2 | | Node 1 | Node 2 | | Node 1 | | Node 2 | | | Node 1 | Node 2 |
| | | | | | | | | | | CPU | GPU | CPU | GPU | | | |
| **Idle Mode** | **1.47%** | **7.23%** | **14.5%** | **0%** | **0%** | **800 MB** | **1.23 GB** | **1.28 GB** | **41 C°** | **29 C°** | **27.5 C°** | **27 C°** | **25.5 C°** | **N/U** | **N/U** | |
| **1** | **15%** | N/A | | N/A | | **1.15 GB** | N/U | | **49.9 C°** | N/U | | N/U | | **C + OD** | N/U | |
| **2** | **14.70%** | N/U | **65.60%** | N/U | **13%** | **1.19 GB** | N/U | **3.16 GB** | **50 C°** | N/U | | **31 C°** | **29.5 C°** | **C + OD** | N/U | **IE** |
| **3** | **17.20%** | **31%** | N/U | **98%** | N/U | **1.13 GB** | **2.68 GB** | N/U | **49.2 C°** | **36 C°** | **33.5 C°** | N/U | | **C + OD** | **IE** | N/U |

Moreover, an experiment that lasted for twenty-four hours was carried out to guarantee the system's consistency and prevent any disconnection in the servers. Table 4.18 illustrates resource measurements in the ideal mode by keeping the servers alive for an entire day when no processing is required or needed. The amount of RAM used by both nodes is roughly half the amount supplied (4GB), which suggests that more tasks may be completed or take place beside the running ones. In addition, managing various low-light levels may be handled by combining optimal approaches (multiple networks) inside a single node. Moreover, additional nodes for various purposes (e.g. rain & fog removal, noise suppression...etc.) can join the proposed design to handle different circumstances. The moderate "*Temp*" values for both nodes indicate that device environments are steady with no risks of overheating or side detriments.

**Table 4.18** Metrics of the Edge-Servers *(Nodes)* running for a whole day (ideal mode).

| Measure/Node | Node 1 | Node2 |
|---|---|---|
| **RAM Usage** | 2.72 GB | 2.78 GB |
| **CPU/GPU Temp** | 34 C°/ 33 C° | 29 C° / 28.5 C° |
| **CPU/GPU Usage** | 7.93% / 0% | 14.90% / 0% |

The "*Latency*" is a vital aspect to consider since the implemented design is suggested on devices with limited resources; thus, it is one of the most critical variables. In other words, the whole amount of time required to perform **Classification**, **Enhancement**, and

**Detection** on a single picture. The following Tables 4.19a and 4.19b illustrate the amount of time required to complete a separate task and the overall one across all tasks. In addition, it is interesting to note that the sole difference between *RUAS* and *Zero-DCE++* pertains to the enhancement process, whilst all other tasks are completed using the same approaches. Since the classification stage only resizes for label prediction, original dimensions are passed to subsequent nodes.

**Table 4.19** Total time taken for processing one-image (from source to destination).

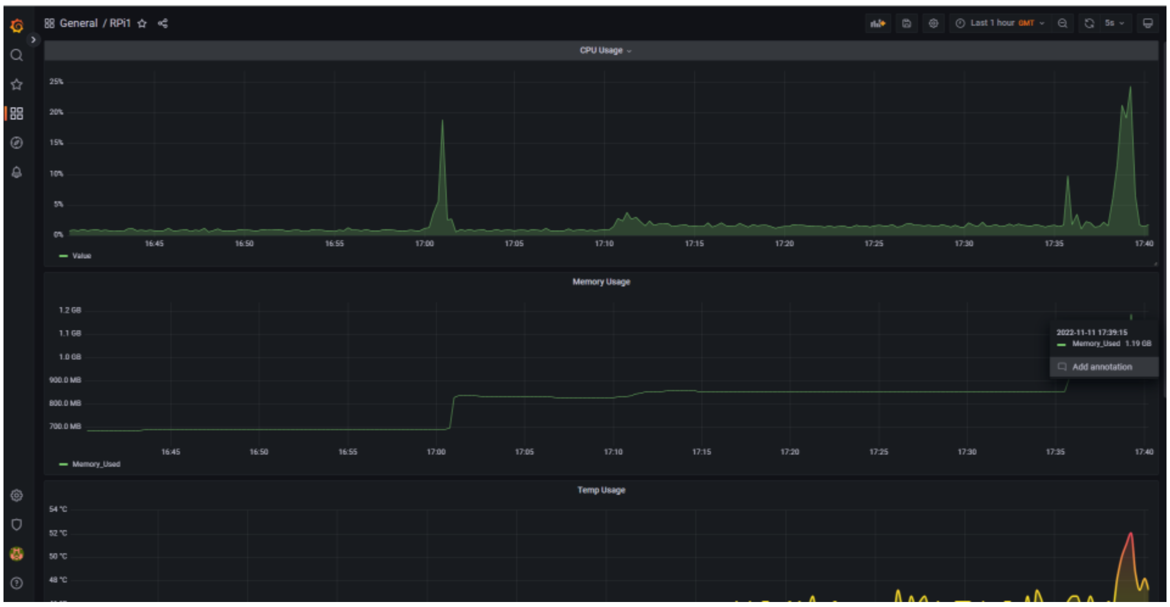
**(a) RUSA.**

| Node Id | Task | Time (in sec) |
|---|---|---|
| **1** | Classifier | 0.4 |
| **2** | Enhancement | 0.03 |
| **3** | Detection | 0.03 |
| **Total Time:** | | **0.46** |

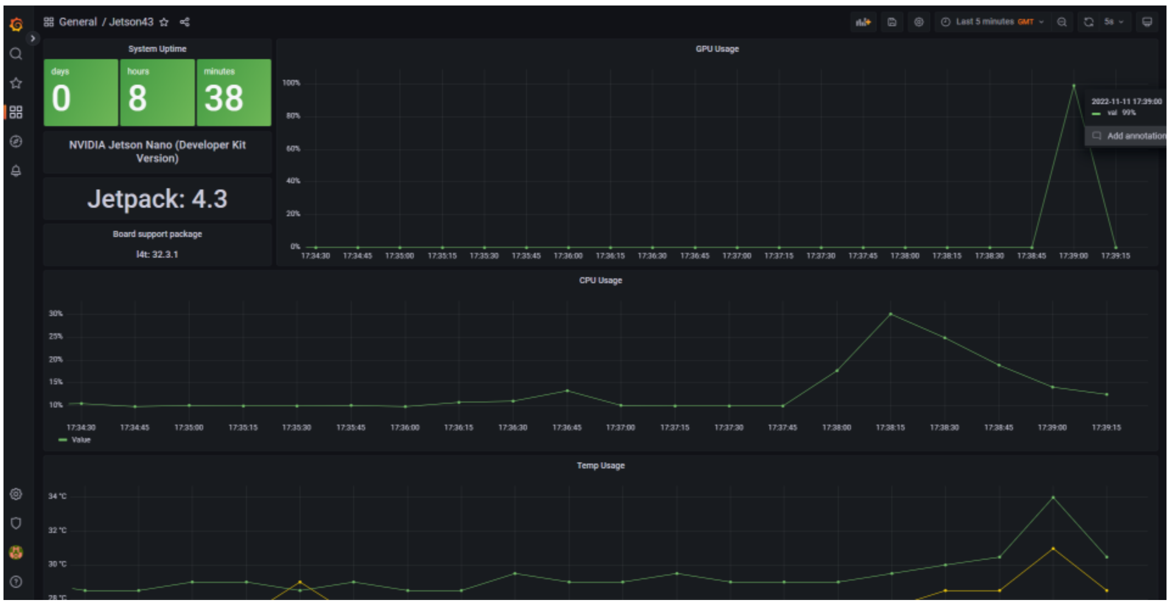
**(b) Zero-DCE++**

| Node Id | Task | Time (in sec) |
|---|---|---|
| **1** | Classifier | 0.4 |
| **2** | Enhancement | 0.001 |
| **3** | Detection | 0.03 |
| **Total Time:** | | **0.431** |

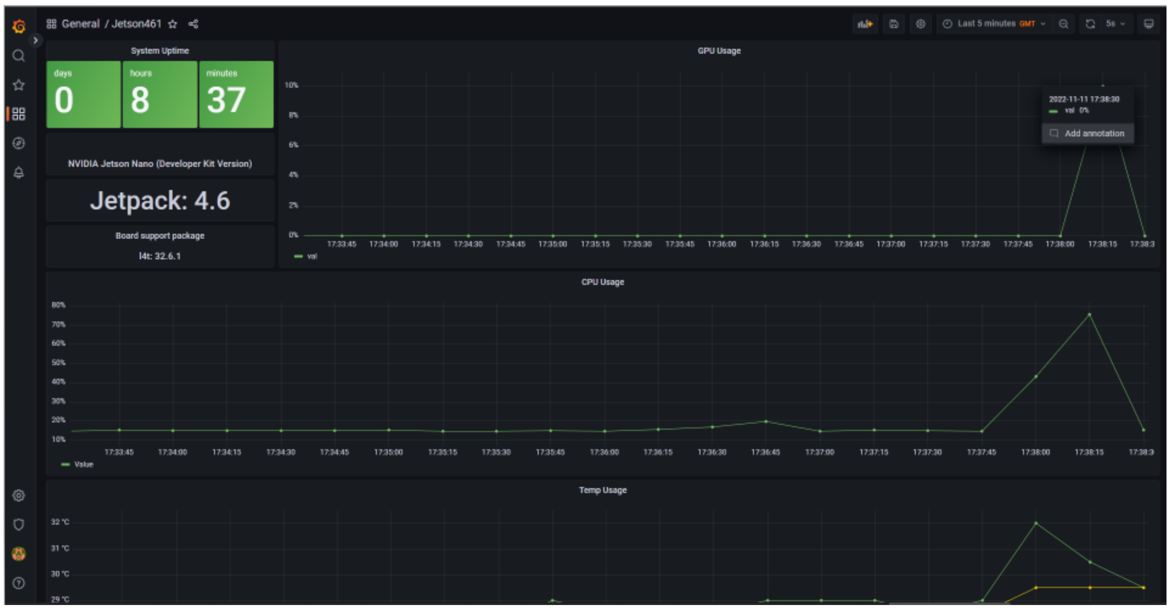
Furthermore, the "*Grafana*" software was used in order to record the constrained-resources devices metrics and measurements presented in the previous Tables 4.17 & 4.18. Grafana is an open-source platform for data visualisation that enables users to express their data in the form of charts, graphs, and other sorts of representations [15]. Therefore, this study utilised the tool to measure the examined metrics by providing each device with an independent dashboard for performance monitoring and documenting any changes in the parameters being tracked. In addition, Grafana is much more helpful for single-board computers when connected to "*Node-Exporter*" and "*Prometheus*". Since "*Node-Exporter*" is used to extract the essential metrics values from a particular device such as RAM, CPU and Temp, to name a few, whereas "*Prometheus*" allows direct communication between the machine and grafana platform by sending the desired metrics directly to the dashboards for illustration and monitoring purposes [12]. Therefore, a distinct dashboard is developed for each device. These dashboards are shown in the following Figures 4.11a, 4.11b & 4.11c to represent Raspberry Pi *(for the classification and detection tasks)* and the two Jetsons *(for the enhancement task)*, respectively. The computational resources were recorded and monitored while the jobs were carried out. For instance, Figure 4.11b, specifically the GPU usage, displays a peak signal of 99% of the total usage while the enhancement step is active and taking place. In addition, other critical mete-data, such as *"timestamps"*, makes it possible to link a specific utilisation with the activity being performed.



**(a)** Raspberry Pi-4B Dashboard *(Node 3)*.



**(b)** Jetson Nano Developer Kit Dashboard *(Node 1)*.

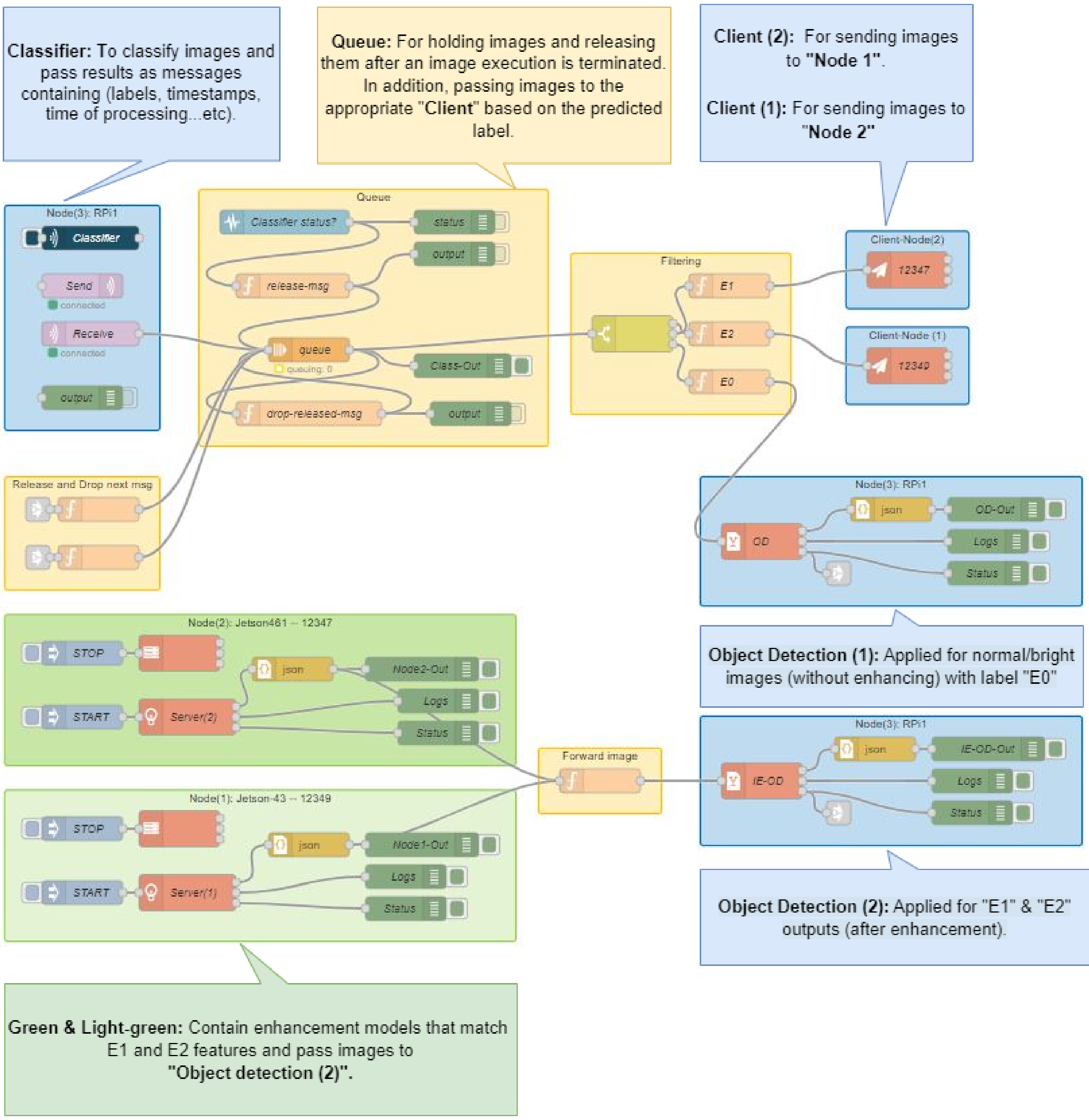


(**c**) Jetson Nano Developer Kit Dashboard *(Node 2)*.

**Figure 4.11** Grafana dashboards for constrained-resources devices.

As stated before, the complete implementation of the suggested approaches was carried out and tested using the "*"Node-Red"* platform. The NR allows numerous machines and services to connect and interact with one another either locally or remotely. Figure 4.12 illustrates the design components from the source to the final destination. To differentiate between the many stages, each colour in the figure represents a particular "*Node*" with a brief description of each node's role. The "*Blue*" colour refers to all the tasks running on the Raspberry Pi (Node 3) whereas "*Green & Light-green*" for Jetson Nanos devices (Node 1 & Node 2). On the other hand, the "*Yellow*" represents built-in functions and nodes in the NR workspace for various purposes (e.g. Decision making, splitting and merging messages, converting formats and further).

Consequently, the outputs consist only of messages or notifications that include the following components: the number of detected objects, the object classes (e.g. people, vehicles, etc.) and the timestamp, to name a few. In addition, the image output is produced for a particular scene, complete with instances, their bounding boxes, confidences and class names.



**Classifier:** To classify images and pass results as messages containing (labels, timestamps, time of processing...etc).

**Queue:** For holding images and releasing them after an image execution is terminated. In addition, passing images to the appropriate "**Client**" based on the predicted label.

**Client (2):** For sending images to **"Node 1"**.

**Client (1):** For sending images to **"Node 2"**

**Object Detection (1):** Applied for normal/bright images (without enhancing) with label "E0"

**Object Detection (2):** Applied for "E1" & "E2" outputs (after enhancement).

**Green & Light-green:** Contain enhancement models that match E1 and E2 features and pass images to **"Object detection (2)"**.

**Figure 4.12** The proposed design implemented in Node-Red.

In designing the Node-RED experiment, reliance is placed on several fundamental assumptions to facilitate smooth operations. First and foremost is the recognition that the quality of input images plays a crucial role, influencing both the accuracy of image classification and subsequent decisions regarding enhancement techniques. A pivotal assumption concerns the consistent and reliable capabilities of the edge devices involved. Any variations in these capabilities could potentially disrupt accurate image classification and the subsequent selection of appropriate enhancement methodologies.

The stability of Node-RED is also regarded as essential, as it guides decision-making processes related to the selection of enhancement techniques based on image classification outcomes. Ensuring a dependable and steady internet connection for seamless data transfer between stages is another key assumption. Furthermore, the accuracy of the classification model is considered paramount, influencing the appropriateness and effectiveness of chosen enhancement techniques.

The relationship between stages, particularly the presumed positive impact of enhancements on object detection, introduces complexity and requires careful consideration. Object detection precision, real-time processing capabilities, and the assumption of continuous processing collectively contribute to the overall effectiveness and efficiency of the experimental testbed. Lastly, the assumption of minimal user interaction emphasizes a preference for a streamlined and automated workflow, acknowledging the potential influence of unexpected inputs on classification and enhancement decisions. Documenting these assumptions is critical for maintaining transparency and facilitating a nuanced interpretation of results in real-world applications.

## 4.4 Chapter Summary

In this chapter, all the assessments and outcomes were conducted, and their presentation was in-depth. First, every component of the proposed system was broken down and explained individually, beginning with the classification stage, which included the following steps; data preparation, the feature extraction part, followed by the traditional machine learning algorithm "*Random Forest*" for classifying and producing labels during the training and testing stages. Afterwards, methods for enhancing low-light images were tested and assessed alongside the cutting-edge object identification algorithm. Indeed, enhancement models that excelled in some areas after being benchmarked on the cloud paradigm for the feature extraction phase were selected to contribute to the whole design. On the other hand, for the detection component, it is recommended to use a "*Large-model*" for the Cloud and a "*Small-model*" for the edge environment, which is comprised of many restricted devices. In addition, the computing resource metrics of these devices were considered to evaluate the system's performance and stability while a process was being carried out. Lastly, the findings presented are reviewed and discussed in the next and last chapter. Also, future directions are given, along with particular constraints encountered throughout this research.

# Chapter 5

# Discussion & Conclusion

In this last chapter, findings and analysis are given significant attention and addressed in depth. Moreover, it details the challenges encountered throughout the study and offers potential ways to address them. In addition, by putting forth innovative concepts and approaches for enhancing the scalable nature and adaptability of the design suggested. Ultimately, this thesis comes to a close with a brief overview of the endeavour. Additionally, an extra example has been included in Appendix A to demonstrate and better understand the obtained findings. Finally, all references relating to this thesis may be found at the very end of this document.

## 5.1 Discussion

In order to conduct a comprehensive analysis, each stage suggested in the proposed design is discussed independently. Starting by the core of this research, the Meraki cameras are installed across all the 5G Wales Unlocked project use cases and acted like sensors. The cameras initiate a pre-identify signal in case a potential anomaly occurs in the fields with the aid of a tiny AI algorithm embedded within the camera (referred to as Edge camera; *a camera that can carry out a job in an intelligent and automated way without the need for any involvement from a human being*). Therefore, by leveraging the potential signal produced by the edge camera, further analysis may be carried out to ensure event correctness through more effective procedures and methodologies to prevent anomalous occurrences and mitigate annoying false warnings. One of the most critical capabilities of the camera is providing scenes brightness, also known as *"Lux"* value, and it represents the overall brightness of the image with a single number that may vary from zero (for wholly dark or black settings) and greater (for bright scenes). In addition, the numbers that fall in the middle of the spectrum indicate varying degrees of brightness. Despite this, the brightness function of the camera was not used in any way throughout this study, nor was it ever considered for different reasons. First, the lux-value is determined by taking the average of all the pixel values, which may be done by converting an RGB image into various colour spaces (e.g. Gray-scale or YCbCr, where "Y" represent the luminance map of an input image). Because of this, it is difficult to represent a whole scene with a monocular value accurately. Secondly, weather conditions are unpredictable and inevitable in most cases leading to mashups of scene representations in terms of cloudy bright or dark, foggy and rainy, to name a few. Therefore, more than a single value is needed to contribute to a better understanding of scene properties. On top of that, it is challenging to achieve diverse brightness levels in different scenarios when cameras are fixed in position and height, similar to the 5G Wales Unlocked scenarios.

Furthermore, since this study focuses on captured images with varying short exposures, producing different short-exposure images of the same scene takes time and effort. Based on that, the primary emphasis was placed on retrieving low-light images from publicly available datasets such as the Exclusively Dark Image Dataset, also known as the "*ExDark*". The dataset is postulated and represented as the input data captured from multiple types of cameras (e.g. surveillance systems, mobile and camera sensors) placed in various indoor or outdoor environments. In addition, to facilitate the assessment of several enhancement models on various low-light scenes, the dataset was collected under various situations, such as varying angles, resolutions, the amount of noise that had accumulated, and different kinds of weather. These situations were considered during inspecting and understanding the utilized data. However, neglecting the luminosity feature provided by the Meraki camera for the reasons mentioned earlier encourages the necessity of implementing an algorithm to separate and distinguish brightness levels or at least normal images captured in the daytime with sufficient lighting from ones that lack bright pixels taken in darkness, such as nighttime.

Therefore, a lightweight dynamic classifier was introduced for the current study for managing brightness levels. The new suggested pipeline with an additional classification method is substantially different from the standard pipeline proposed during the 5G Wales Unlocked project, which only uses images captured in the daytime with sufficient light and needs to be designed to distinguish images with various light intensities. Whether it was a binary or categorical label, many different approaches, procedures, and tactics were researched and examined for the classification task. Most of these studies concentrate on contemporary techniques such as Deep Learning (DL), whereas others focus on Traditional Machine Learning (TML). When selecting acceptable methods, it is necessary to consider several essential aspects before making a decision. For instance, the computing resources available (e.g. GPUs, storage, etc.), place of the processing and network availability, to mention a few. Consequently, classification techniques that depend on deep learning methodologies demand a large amount of data for training purposes, generating heavy models that may not fit on devices with limited resources.

Furthermore, this research examined the proposed design in the context of both computing paradigms: cloud-based and edge-based. It aimed to identify the differences between them. Generally, large models slow down the inference stage, increasing the time-responsiveness *"Latency"* at the end-user side. Moreover, research projects often use heavy methods hosted and supported with high-computational resources (e.g. cloud or fog paradigms) to complete and accelerate processing in a shorter period. On the other hand, studies rely on partly processing what is known as a "*Joint Modelling*" when tasks are shared between the Cloud and Edge computing paradigms. In this scenario, specific challenges may be overcome by using a large model for handling heavy workloads and completing on-edge cameras or constrained devices, for example. However, network congestion and bandwidth size affect factors like latency since some tasks are required to be transmitted to the Cloud.

, with a particular emphasis on the edge paradigm, which is one of the primary objective.

This research emphasizes implementing stages on multiple resource-constrained devices within the edge paradigm and studying associated metrics as part of a proof-of-concept. In other words, each explained and stated phase in previous chapters are conducted dependently on a separate device *"Node"* obtaining lower time-response and maintaining exact or rough performance. Feature engineering for feature extraction along with the traditional machine learning algorithm *"Random Forest"* were used to develop a lightweight approach capable of running on a device with limited resources (e.g. Raspberry Pi, with limited storage and CPU usage). Instead of relying on modern approaches to extract features, a feature engineering approach uses the Gabor filter, original pixel values, and the Sobel filter. Alongside various image enhancement algorithms exhibiting varying performance and functionality, the feature extractor can identify unique features representing each specific enhancement technique. For instance, the image enhancement algorithm $E_1$ may excel in handling noisy dark scenes, while $E_2$ may be more effective in addressing foggy dark scenes.

This study only targets detecting anomalies *People and Car* classes. The presented ExDark dataset contains 7363 images that have been organised and stored within individual class folders, each representing a single class making retrieving the appropriate images for a given task more effortless. Moreover, the collection only contains samples in completely dark or moderately dark environments. As a result, additional samples were collected with enough light from well-known public databases such as Berkeley, Stanford, and MS-COCO datasets. They were combined to produce a range of different brightness levels. In addition, by enabling the classifier to study the data, it can discern between bright and dark situations. The process of discriminating between dark and bright scenes is simple and logical. However, no general guideline or rule of thumb exists for deciding on partially-dark scenes. Therefore, humans have different ways of seeing semi-darkness, and the necessities of different situations call for different interpretations.

For example, the timing of sunrise and sunset and areas partly covered by black pixels are some of the criteria that might represent scenes as semi-dark. However, selecting a particular enhancement model for a given sort of low-light conditions cannot be aided by these viewpoints. For instance, $I_1$ (representing an image sample that is only partially dark) is improved by $E_1$, whereas $E_2$ for completely darkened scenes $I_2$. Therefore, several enhancement techniques were tested and evaluated against each other, with the primary objective of improving object detection in order to enhance images with appropriate technique based on the image features. All enhancement algorithms were put through their paces by being tested and evaluated on the complete datasets. After going through the first step, which consisted of enhancement (using low-light image enhancement techniques; see Table 4.4, images moved on to the second stage, the detection *(with Detectron2)*. Therefore, allowing for the calculation of essential metrics helped put constraints on optimal approaches besides the trade-offs between them.

As a result, consideration has been given to incorporating aspects such as processing speed and training during inference and especially the object detection performance to the development of the classifier. According to the findings, the *"Zero-DCE++"* method is the optimum approach for the vast majority of situations. In terms of more rapid inference on several different input dimensions, around (0.001 second). In addition, the highest was accomplished using the following methodologies: *"Zero-DCE++" (on the Person dataset)* and *"MBLLEN" (on the Car dataset)*. The baseline model produced an *"mAP"* value of (0.34 & 0.45) for the *"Person and Car"* datasets. Both models achieved a higher *"mAP"* value, equivalent to (0.59 & 0.48), than the baseline model. Again, when assessing a single class *"AP"*, the *"Zero-DCE++"* outperforms other approaches for the *"Person, Bus, and Motorcycle"* classes with accuracies of (0.77, 0.71, & 0.27), respectively. However, the baseline results for several classes had extremely few advanced placements or almost none. The *"EnlightenGAN"*, on the other hand, did quite well in the *"Car"* class, obtaining an *"AP"* score of (0.64). In addition, the *"RUAS"* was the only model that produced more significant prediction after enhancement than the others did for both datasets, about 2393 and 1836 bounding boxes, respectively, compared to the actual predictions; 2073 and 1700. Indeed, the graphs in 4.5 demonstrate that at least seven different methods are able to identify a greater number of objects than either direct detection *(without enhancement)* or even the ground truth ones.

Consequently, the following is a list of best practice approaches for the enhancement component, depending on evaluation and findings. Instead of relying on the techniques investigated in this study, it is essential to note that alternative techniques, either those now in use or those not yet developed, may be used to brighten dark images. Similarly, alternative state-of-the-art object identification algorithms that exceed the one picked or trained on dark or night data may achieve greater accuracy and better performance than the one chosen.

The below-listed techniques are selected and preferred for joining and contributing to the proposed design on the edge environment and the classifier modelling part.

- Zero-DCE++
- RUAS
- CSDNet_UPE
- LiteCSDNet_UPE
- SLiteCSDNet_UPE
- MBLLEN
- ElightenGAN
- SLiteCSDNet_LOL

After benchmarking the models on the Cloud-paradigm, those models were evaluated on the edge environment. However, to obtain quicker inference and fitting restricted devices, the large detector *"Detectron2"* was switched out with a lighter one *"Yolo-tiny"*. Due to the fact that the purpose of this study is to measure the detection performance after enhancement rather than evaluate enhancement methods based on image quality—the enhancement techniques provided exact results on limited resources as in environments with high-computational resources. However, when it comes to the detection stage, outputs vary compared to the large detector. The reason behind this is crucial and logical since tiny versions undergo several techniques, such as pruning or quantisation, which allow us to obtain a smaller version of a neural network model that is smaller and more efficient. Therefore, these techniques scarify some accuracy to achieve that goal. As a result for the edge environment, the *"RUAS"* among all the techniques outperformed with an *"mAP"* equal to 33% compared to the baseline *"Yolo-tiny"*, which achieved 29% on randomly chosen samples of 40-images.

Moreover, inference speed remains as reached during benchmarking with 0.001 seconds & 0.03 seconds for the *"Zero-DCE++"* and the *"RUAS"* respectively. Consequently, the findings suggest that utilising and executing these models on devices with minimal resources is feasible to acquire results in real-time, *"Low latency"*. Moreover, the detection results also highlight the importance of image-enhancing approaches as a step of pre-processing, which helps to increase the number of objects that can be identified and discovered in low-light settings, *"High-accuracy"*. As stated, all phases may be replaced with modern or optimal approaches.

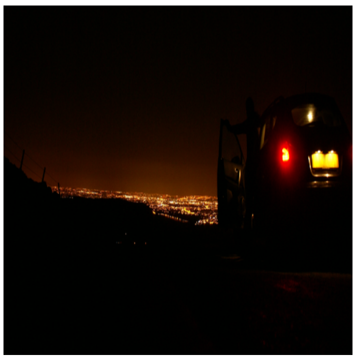
Furthermore, there were instances in which the enhancement step led to the attainment of the same outcomes as those obtained by direct inference or even worse, as was depicted in Table 4.14. The reason might be that some techniques add additional artefacts to the output image after enhancing, such as noise, unwanted colours or distortion and brightening bright pixels. All these factors can potentially affect existing objects by covering or vanishing objects' characteristics such as edges and texture. Therefore, relying solely on one approach might not produce the desired results. Thus, the classification task was built to manage the enhancement stage for a single image depending on the technique‘s capabilities and attributes. During the classifier generation step, the models listed in 5.1 were utilised since they are considered the most effective approaches; this was done to reduce time and effort while ensuring that the findings were correct. Further, calculating the *"mAP"* for a solo image and only outputs with values equal to or greater than 0.9 were taken into consideration. Afterwards, since similar samples are used across all the techniques, only unique images are filtered to represent the outperform enhancers.

**(a)** CSDNet_UPE.

**(b)** ElightenGAN.

**(c)** MBLLEN.

**(d)** Zero-DCE++.


**(e)** SLiteCSDNet_UPE.

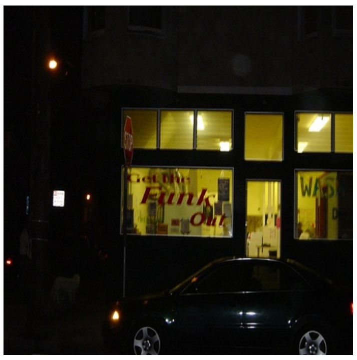

**(f)** RUAS.

**Figure 5.1** Unique samples outperformed by specific techniques.

Following the filtering process of unique inputs, Figure 5.1 exhibits a few original samples with high mean average precision . For example, sample 5.1a depicts one of the inputs that *"CSDNet_UPE"* used to enhance and identify all objects obtaining an accuracy of ($\geq$ 0.9) and outperforming the other techniques. In contrast, other methods accomplish a lower level of success than the threshold. Consequently, input properties and features of similar or comparable ones may be improved by the same model, *"CSDNet_UPE"*. In addition, these samples assist in differentiating between many methods and appropriately for the classifier creation by isolating those characteristics that may reflect a distinct and unique enhancing approach via a feature extraction stage. Nonetheless, the findings demonstrated a restricted number of images acquired from each method; see Table 4.13. As a point of comparison, the lowest number of samples obtained is "4" by *"LiteCSDNet_LOL"*, and the highest number of samples achieved is "30" by *"MBLLEN"*. As a result, the feature extraction process, which relies on a minimal quantity of data, could not perform as efficiently with large-volume of data. Because of this, only two enhancement strategies could contribute to this research because of their fast inference and high level of true positives prediction compared to the ground truth, where these approaches are; the *"Zero-DCE++"* and *"RUAS"*. As mentioned earlier, the *"Zero-DCE++"* showed promising results in different aspects as the fastest model and performed well on the detection task. On the other hand, the *"RUAS"* has similar attributes besides the ability to detect more instances.

As a result, the classifier achieved an accuracy of 85.24 % for categorising inputs to their respective labels. Moreover, images taken under normal circumstances may easily be differentiated from low-light ones. On the other hand, it might be challenging to differentiate between the suggested enhancing approaches. One possible explanation is that each approach can only outmatch a certain number of distinct samples. Further, putting the suggested design into action on limited devices demonstrated substantial performance and the system's scalability. Despite distinct nodes being responsible for completing the duties given to them differently, however, a high processing speed may still be accomplished when a single sample of *H x W* dimensions are processed roughly in *one second* through all of the steps *(Classification, Enhancement, and Detection)*. Thus, the minimal time satisfies the real-time requirements, and lower latency is achieved on the end-user side when receiving notifications or alerts.

In summary, this research narrowed its focus to two main key points. The first pertains to the 5G Wales Unlocked project, addressing specific limitations encountered during its implementation. The second point deals with the gap identified in prior studies, as outlined in the literature review. The 5G project encountered numerous limitations that resulted in inefficient anomaly detection, particularly in scenes with insufficient light, such as dark nights. Indeed, a significant percentage of false alerts was received during nighttime, suggesting a probability of anomalies and due to the poor image quality and low light, validating the presence of objects was challenging. Furthermore, additional challenges arose from blurring and other types of noise. Despite these issues, the object detection approaches still managed to identify some instances, especially when compared to images with low-light problems.

On the other hand, previous studies that explored anomaly detection in edge computing primarily focused on recognizing objects in clear image data, where lighting conditions are sufficient (e.g., daytime) and the images are free from noise, maintaining high quality. Additionally, they often utilized a limited and the same dataset for validation.

Therefore, in this research, a technique was developed to address the gap created after mapping the primary challenge in the 5G project with the gaps identified in literature studies. This was achieved by testing several low-light image enhancement techniques to brighten dark images as a pre-processing step, thereby boosting detection accuracy. Additionally, the study involved a detailed analysis of each enhancement technique to determine the best practice approaches to join the proposed design. Moreover, the data collected during the 5G project was integrated into both the creation and validation phases to test the technique on real-world data, specifically images captured in the daytime to diverged from the common practice of relying on the same dataset used in all studies in the literature.

Moreover, the creation of a lightweight classifier helped distinguish between images captured with low-light conditions and those with ample light. As a result, images with sufficient light are directly sent to the detection task to identify objects, requiring no enhancement.

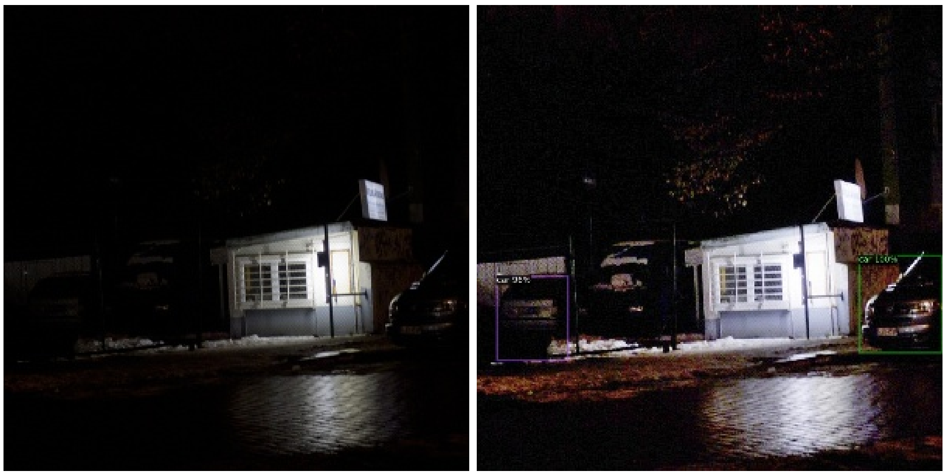
Additionally, this approach serves to differentiate between the old proposed pipeline in the 5G project and the new one.

Furthermore, the proposed technique was implemented on both edge and cloud-based computing to assess its reliability on both paradigms, with a particular emphasis on the edge. Finally, a proof-of-concept was developed to implement the entire design on the edge, rather than relying on high-computational resources such as the cloud or dividing tasks between the edge and the cloud, as studied in previous research. Additionally, the goal was to demonstrate the feasibility of incorporating other tasks into the proposed nodes and adding additional nodes for different purposes in the design.

## 5.2 Limitations & Future Work

Several constraints and gaps have been faced during the implementation and evaluation phases of the studied methodologies. Firstly, the ExDark dataset collection includes images taken in low-light environments with varying amounts of available light. However, the data collection and creation goal was not aimed at resolving low-light image enhancement tasks while emphasising more on high-driven tasks such as object detection and classification by providing objects bounding boxes coordinates and environment types based on brightness levels, respectively. This was explored while applying the detection stage after the enhancement.

The findings and analysis that compared predictions produced by the detector to ground truth, apart from human inspection and investigation, the following was determined: (1) After brightening methods were used for low-light images, in some occasions, more items seemed covered and obscured by darkness or black pixels. Thus, these recently found items are absent in labels that allow them to be included in the ground truth and be considered through the evaluation step; see Figure 5.2. For a better understanding, the original picture [5.2a] includes two instances, both vehicles; one on the right is part of the ground truth predictions, while on the left was discovered by enhancement, 5.2b. Moreover, it should be noted that direct detection is also unable to identify the labelled *"Car"* on the right; however, this was the goal of the current research. As a result, the newly discovered items considerably impact the object identification performance as a whole, leading to obtaining an inadequate *"mAP"* and set of results. In addition, (2) A significant number of instances are missed or un-labelled in the ground truth, which includes far away and tiny objects, in addition to only partially displayed objects; see Figure 5.3 for more illustration. Once again, this leads to a significant reduction in the overall accuracy of the detection. Further, it is necessary to consider that modern state-of-arts object detection algorithms are being created or will seek to increase resilience and reliability. Consequently, it is necessary to take into account objects captured under a variety of situations. (3) Regarding the classification task, the number of unique images that can outperform for single enhancement model is limited, especially during the feature extraction phase. Because of this, it is recommended that the additional public datasets might boost the likelihood of obtaining more unique images per enhancement technique from which to extract relevant characteristics.



**(a)** Before Enhancement. **(b)** After Enhancement.

**Figure 5.2** Sample image of *"Car"* hidden in the dark *(Purple Bounding Box)*.

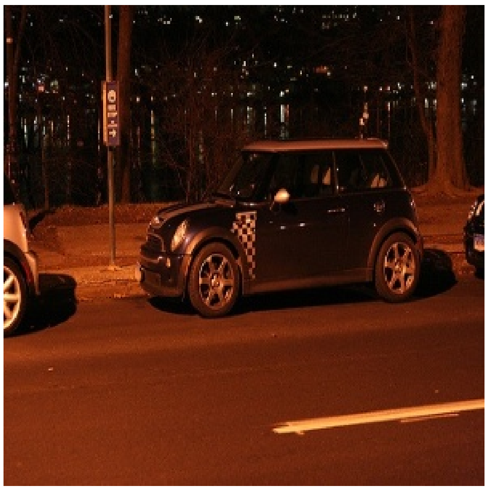


**Figure 5.3** Sample image with objects that only appear partially of the whole image *(Left & Right "Cars").*

As stated before, the goal is to utilize pre-existing methods for low-light image enhancement and object recognition roles; therefore, novel or other existing approaches for both tasks might replace the currently suggested and studied ones supposing that they provide promising results. Moreover, the numerous techniques for dealing with poor light conditions produced varying results due to the fact that the methods' development included a variety of distinct strategies. During the learning phase, for example, methods that merely relied on low-light data paired and unpaired images. In addition, the availability of a wide variety of datasets that differ in scenarios, existent objects, and resolution, to name a few, may develop a one-of-a-kind method that is well suited to a particular low-light environment and use cases. Indeed, several methods demonstrated varying degrees of success in illuminating dark scenes and correctly recognizing objects, in addition to discovering new instances in shadowy and gloomy regions. Therefore, it is advisable not to depend on just one method; nevertheless, mixing many best practices may assist in covering a more significant number of facets. The restricted devices' observed metrics and processing capacity pointed to a major orientation toward consolidating more jobs and computations onto a single device.

In a future study, several networks *(containing numerous low-light enhancing approaches)* may be carried out in a single *"Node"* to control issues relating to inadequate illumination. Thus, the appropriate enhancer from among the selected networks is carried out depending on the acquired input attributes. Further, moving techniques related to a particular problem to a separate and independent node enables the addition of more *"Nodes"* to the design, which can then be used to handle different conditions, such as Deblurring, Deraining or Defogging, and noise removal, to name just a few. In addition, an image may be pre-processed and directed to numerous nodes before the actual detection operation is carried out when a classification algorithm produces multi-labels representing the input content and circumstances. Regarding the ExDark utilized, it is possible to consider upgrading the dataset ground truth as a different line of study. In other words, methods for enhancing dark pixels allow the discovery of new objects not defined or labelled to be included with the actual predictions.

As a result, the entire dataset can be run through the same models studied or only outperformed ones to enhance and then re-labelling those objects, which assists researchers who focus on related areas in evaluating new methodologies in low-light image enhancement domain with the high-driven tasks; object detection and classification obtaining reasonable results.

## 5.3 Conclusion

This research aimed to enhance object detection in low-light conditions using existing image enhancement techniques, with a particular focus on detecting people and vehicles (referred to as anomalies). To achieve this goal, various image enhancement techniques were tested in combination with object detection. In addition, a classifier to distinguish between bright and dark images and classify different dark levels based on image features was introduced. Furthermore, by implementing the proposed design on multiple IoT edge devices within the edge computing paradigm and compared the system's performance with the cloud-based system.

This research was carried out in accordance with the 5G Wales Unlocked project, which was supported by the Department for Digital, Culture, Media & Sport (DCSM-UK) and intended to identify abnormalities in a variety of settings as well as domains. Along the project, several restrictions and gaps were encountered. The most notable was a pre-identify signal received during and after sundown, followed by an automated *"API"* requesting a screenshot from the edge camera. Most of the time, the requested images have undesirable singles and distortion due to many black pixels, making it difficult to distinguish and identify the contents and objects within the images, especially anomalies. Therefore, this study's primary goal was to use previously developed methods in deep learning and computer vision to improve the performance of identifying abnormalities in situations when the available light was limited. Indeed, intensive methodologies and strategies have been studied over the past few years, only open-sourced and readily accessible techniques were presented and investigated to achieve the goal, allowing us to investigate further techniques that provide reliable results for future works.

Moreover, several advantages and disadvantages were revealed throughout the assessment stage, demonstrating strategies, capabilities and differences. In fact, the findings demonstrated a discernible enhancement in both the visual content and its quality in some instances. However, it is essential to remember that the primary purpose was to identify or detect objects within an image rather than comparing strategies for the enhancement role itself. In addition, more is needed to develop an innovative approach in the computer vision domain that may be compared or competitive to prior techniques.

Further, compared to direct detection when enhancement is not required, outputs detection after enhancement demonstrated a significant increase in accuracy as well as in the number of detected instances, regardless of whether they were performed on all classes or a single one. Logically, increased detection might relate to erroneous detection in most instances. However, monitoring and analysing the detector via the *"mAP"* and other metrics ensured the recognition of correct extra or hidden objects in challenging settings. Thus, introducing this stage as a prerequisite is essential for managing low-light circumstances in domains supported by surveillance systems and vision sensors. To name a few, *Zero-DCE++, RUAS and MBLLEN* have all shown considerable progress in improving detection accuracy. For example, the best practice model *"Zero-DCE++"* demonstrated a percentage increase of 73.5 % on the *"Car"* class, while the same for *"MBLLEN"* showed a percentage increase of 6.6 %. However, the two percentages have significant gaps due to various characteristics regards the *"Person"* class, including the amount of noise accumulated, resolutions, dimensions, and others. As a result, the findings of this study recommend putting these strategies into practice as a preliminary step before moving on to further procedures and analysis. Moreover, prior research has suggested that state-of-the-art object identification algorithms be trained on enormous low-light datasets to be used effectively in nighttime or gloomy environments. However, the problem stills have been experiencing relates to items being entirely hidden by darkness and low-value pixels almost reaching zero in value.

On the other hand, novel approaches or existing ones consisting of integrated feature extraction stages for enhancement and detection to develop an entire model may produce a large and heavy one that needs to improve in performance and fitting regarding resource-constrained devices. For example, the proof-of-concept stated at the beginning of this research was achieved by implementing the proposed system on limited resource devices showing a possible execution of approaches regarding low-light enhancement separately by maintaining each speed roughly to *1 second* with high-performance in detecting anomalies. Thus, having jobs running independently on a single *("Node")* encourages moving many systems to the edge instead of relying on high-computational resources such as the Cloud and Fog paradigms. In addition, partial processing requires data transmission or placing a specific task in the Cloud paradigm or near the edge environment, increasing latency, potential cyber-attacks, and channel congestion, to name a few.

Furthermore, most studies classify scenes according to the level of brightness present; for example, a particular label may signify low, moderate, or high levels of light. On the other hand, there is no general rule of thumb for determining the different degrees of darkness except for entirely dark or bright. The recommended classifier presents an innovative method for categorising images with low light levels based on enhancement techniques' ability to boost the detection phase. The *"Zero-DCE++"* and *"RUAS"* techniques provided the highest mean average accuracy on a single input during the detection stage. As a result, these approaches obtained the most significant counts of correct detection images for both dataset classes as well as unique images. In addition, the *"MBLLEN"* performed better than the others, including the ones mentioned above; nonetheless, the selection considers the amount of time required for the inference, given that the primary goal is to reduce the time response on the edge nodes. The *"Zero-DCE++"* and *"RUAS"* that obtained a higher number of unique images as well as performed better on the datasets labelled *"Car"* and *"Person"* respectively, were trained each on the performed dataset in order to extract features and train the lightweight method for classifying low-light samples. In addition, to distinguishing them from images captured with sufficient light. The classification results demonstrated the possibility of using feature engineering approaches along with traditional machine learning algorithms while operating on restricted devices with an inference time of around *0.2 seconds*.

Despite the small number of samples used in the process of creating the classifier, an accuracy of 85.24% was achieved on the test set, indicating the ability to differentiate between totally bright images from low-light ones, as well as among various low-light properties for the appropriate enhancement method. Indeed, the findings demonstrated rivalry amongst all approaches in some facets, such as when model "$X_1$" performed better than model "$X_2$" on "$Image_1$" or vice versa. Therefore, the capabilities of the various models permit a combination and integration of several models for handling various input conditions and directing it to the most confident approach based on particular features within the image, making it feasible by the fact that the models can work together rather than relying on a single method and improving the objects identification task.

# Bibliography

[1] Ajay, B. S. and Rao, M. (2021). Binary neural network based real time emotion detection on an edge computing device to detect passenger anomaly. *Proc. IEEE Int. Conf. VLSI Des.*, 2021-Febru:175–180.

[2] Al-Haija, Q. A., Smadi, M. A., and Zein-Sabatto, S. (2020). Multi-Class Weather Classification Using ResNet-18 CNN for Autonomous IoT and CPS Applications. *Proc. - 2020 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2020*, Dl(Dl):1586–1591.

[3] Angelo, N. P. and Haertel, V. (2003). On the application of gabor filtering in supervised image classification. *International Journal of Remote Sensing*, 24(10):2167–2189.

[4] Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. *Dl*, page 6.

[5] Chen, C., Chen, Q., Xu, J., and Koltun, V. (2018). Learning to see in the dark. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3291–3300.

[6] Chen, D., He, M., Fan, Q., Liao, J., Zhang, L., Hou, D., Yuan, L., and Hua, G. (2019). Gated context aggregation network for image dehazing and deraining. *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 1375–1383.

[7] Chen, N., Chen, Y., Blasch, E., Ling, H., You, Y., and Ye, X. (2017). Enabling smart urban surveillance at the edge. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 109–119. IEEE.

[8] Cisco, V. (2017). Cisco visual networking index: Forecast and methodology 2016–2021.(2017).

[9] Dalal, N. and 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, pages 886–893. Ieee.

[10] Danuser, G. (2011). Computer vision in cell biology. *Cell*, 147(5):973–978.

[11] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

[12] Dl (2022). prometheus/node_exporter: Exporter for machine metrics. https://github.com/prometheus/node_exporter.

[13] Dl (Dla). Github - lutzroeder/netron: Visualizer for neural network, deep learning, and machine learning models). https://github.com/lutzroeder/netron. Accessed: 15/12/2022.

[14] Dl (Dlb). Github - ultralytics/yolov5: Yolov5 in pytorch > onnx > coreml > tflite. https://github.com/ultralytics/yolov5. Accessed: 12-12-2022.

[15] Dl (Dlc). grafana/grafana: The open and composable observability and data visualization platform. visualize metrics, logs, and traces from multiple sources like prometheus, loki, elasticsearch, influxdb, postgres and many more. https://github.com/grafana/grafana. Accessed: 01/12/2022.

[16] Dl (Dld). Model optimizer usage — openvino™ documentation — version(2022.2). https://docs.openvino.ai/2022.2/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html. Accessed: 14/12/2022.

[17] Dl (Dle). Supercomputing wales portal – resources and support for users of the supercomputing wales services. https://portal.supercomputing.wales/. Accessed: 08-12-2022.

[18] Dollár, P. (Dl). Piotr's Computer Vision Matlab Toolbox (PMT). https://github.com/pdollar/toolbox.

[19] Dong, C., Loy, C. C., He, K., and Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307.

[20] Everingham, M. and Winn, J. (2011). The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 8:5.

[21] Fan, J., Upadhye, S., and Worster, A. (2006). Understanding receiver operating characteristic (roc) curves. *Canadian Journal of Emergency Medicine*, 8(1):19–20.

[22] Fogel, I. and Sagi, D. (1989). Gabor filters as texture discriminator. *Biological cybernetics*, 61(2):103–113.

[23] Gashler, M., Giraud-Carrier, C., and Martinez, T. (2008). Decision tree ensemble: Small heterogeneous is better than large homogeneous. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 900–905. IEEE.

[24] Ghose, S., Singh, N., and Singh, P. (2020). Image denoising using deep learning: Convolutional neural network. *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 511–517.

[25] Gonzalez, W. (1992). Digital image processing second edition. *Dl*, Dl(Dl):414–428.

[26] Guerra, J. C. V., Khanam, Z., Ehsan, S., Stolkin, R., and McDonald-Maier, K. (2018). Weather classification: A new multi-class dataset, data augmentation approach and comprehensive evaluations of convolutional neural networks. *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 305–310.

[27] Guo, C., Li, C., Guo, J., Loy, C. C., Hou, J., Kwong, S., and Cong, R. (2020). Zero-reference deep curve estimation for low-light image enhancement. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1780–1789.

[28] Halkarnikar, P., Khandagle, H., Talbar, S., and Vasambekar, P. (2010). Object detection under noisy condition. *AIP Conference Proceedings*, 1324(1):288–290.

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

[30] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596.

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

[32] Hruska, J. (2018). Self-driving cars still can't handle snow rain or heavy weather. *ExtremeTech, By Ziff Davis, LLC*.

[33] Hussain, M., Bird, J. J., and Faria, D. R. (2018). A study on cnn transfer learning for image classification. In *UK Workshop on computational Intelligence*, pages 191–202. Springer.

[34] Ibrahim, M. R., Haworth, J., and Cheng, T. (2019). WeatherNet : Recognising weather and visual conditions from street-level images using deep residual learning. *Dl*.

[35] Jagadeesh, B. and Patil, C. M. (2016). Video based action detection and recognition human using optical flow and svm classifier. In *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 1761–1765. IEEE.

[36] Jiang, Y., Gong, X., Liu, D., Cheng, Y., Fang, C., Shen, X., Yang, J., Zhou, P., and Wang, Z. (2021). Enlightengan: Deep light enhancement without paired supervision. *IEEE Transactions on Image Processing*, 30:2340–2349.

[37] Jocher, G. and Chaurasia, A. (2022). ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. *Dl*.

[38] Juneja, M. and Sandhu, P. S. (2009). Performance evaluation of edge detection techniques for images in spatial domain. *International journal of computer theory and Engineering*, 1(5):614.

[39] Kamnardsiri, T., Charoenkwan, P., Malang, C., and Wudhikarn, R. (2022). 1d barcode detection: Novel benchmark datasets and comprehensive comparison of deep convolutional neural network approaches. *Sensors*, 22(22):8788.

[40] Kim, B., Son, H., Park, S.-J., Cho, S., and Lee, S. (2018). Defocus and motion blur detection with deep contextual features. *Computer Graphics Forum*, 37(7):277–288.

[41] Kotsiantis, S. (2011). Combining bagging, boosting, rotation forest and random subspace methods. *Artificial intelligence review*, 35(3):223–240.

[42] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[43] Kurihata, H., Takahashi, T., Ide, I., Mekada, Y., Murase, H., Tamatsu, Y., and Miyahara, T. (2005). Rainy weather recognition from in-vehicle camera images for driver assistance. *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 205–210.

[44] Lamba, M. and Mitra, K. (2021). Restoring extremely dark images in real time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3487–3497.

[45] Li, B., Peng, X., Wang, Z., Xu, J., and Feng, D. (2017a). An all-in-one network for dehazing and beyond. *arXiv preprint arXiv:1707.06543*.

[46] Li, C., Guo, C., and Chen, C. L. (2021a). Learning to enhance low-light image via zero-reference deep curve estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[47] Li, C., Guo, C., Han, L.-H., Jiang, J., Cheng, M.-M., Gu, J., and Loy, C. C. (2021b). Low-light image and video enhancement using deep learning: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, DI(01):1–1.

[48] Li, C., Guo, J., Porikli, F., and Pang, Y. (2018a). Lightennet: A convolutional neural network for weakly illuminated image enhancement. *Pattern recognition letters*, 104:15–22.

[49] Li, H., Hu, C., Jiang, J., Wang, Z., Wen, Y., and Zhu, W. (2018b). Jalad: Joint accuracy- and latency-aware deep structure decoupling for edge-cloud execution. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 671–678.

[50] Li, Q., Kong, Y., and Xia, S. M. (2014). A method of weather recognition based on outdoor images. *VISAPP 2014 - Proc. 9th Int. Conf. Comput. Vis. Theory Appl.*, 2:510–516.

[51] Li, X., Wang, Z., and Lu, X. (2017b). A multi-task framework for weather recognition. *MM 2017 - Proc. 2017 ACM Multimed. Conf.*, pages 1318–1326.

[52] Li1, Z., Li1, Y., Zhong2, J., and Chen, Y. (2020). Multi-class weather classification based on multi-feature weighted fusion method. *IOP Conf. Ser. Earth Environ. Sci.*, 558(4).

[53] Lim, B., Son, S., Kim, H., Nah, S., and Mu Lee, K. (2017). Enhanced deep residual networks for single image super-resolution. *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144.

[54] Lim, S. and Kim, W. (2020). Dslr: deep stacked laplacian restorer for low-light image enhancement. *IEEE Transactions on Multimedia*, 23:4272–4284.

[55] Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. (2016). Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944.

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

[57] Liu, B., Gould, S., and Koller, D. (2010). Single image depth estimation from predicted semantic labels. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1253–1260.

[58] Liu, R., Ma, L., Zhang, J., Fan, X., and Luo, Z. (2021). Retinex-inspired unrolling with cooperative prior architecture search for low-light image enhancement. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10561–10570.

[59] Loh, Y. P. and Chan, C. S. (2019). Getting to know low-light images with the exclusively dark dataset. *Computer Vision and Image Understanding*, 178:30–42.

[60] Lore, K. G., Akintayo, A., and Sarkar, S. (2017). Llnet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition*, 61:650–662.

[61] Lu, C., Lin, D., Jia, J., and Tang, C. K. (2017). Two-Class Weather Classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2510–2524.

[62] Lu, K. and Zhang, L. (2020). Tbefn: A two-branch exposure-fusion network for low-light image enhancement. *IEEE Transactions on Multimedia*, 23:4093–4105.

[63] Lv, F., Lu, F., Wu, J., and Lim, C. (2018). Mbllen: Low-light image/video enhancement using cnns. *BMVC*, 220(1):4.

[64] Lyons, M., Kamachi, M., and Gyoba, J. (1998). The Japanese Female Facial Expression (JAFFE) Dataset. *Dl.* The images are provided at no cost for non- commercial scientific research only. If you agree to the conditions listed below, you may request access to download.

[65] Ma, L., Liu, R., Zhang, J., Fan, X., and Luo, Z. (2021). Learning deep context-sensitive decomposition for low-light image enhancement. *IEEE Transactions on Neural Networks and Learning Systems*.

[66] Maltezos, E., Lioupis, P., Dadoukis, A., Karagiannidis, L., Ouzounoglou, E., Krommyda, M., and Amditis, A. (2022). A Video Analytics System for Person Detection Combined with Edge Computing. *Computation*, 10(3):35.

[67] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802.

[68] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2.

[69] Melvin, A. A. R., Kathrine, G. J. W., Ilango, S. S., Vimal, S., Rho, S., Xiong, N. N., and Nam, Y. (2021). Dynamic malware attack dataset leveraging virtual machine monitor audit data for the detection of intrusions in cloud. *Transactions on Emerging Telecommunications Technologies*, page e4287.

[70] Memon, M. M., Hashmani, M. A., Junejo, A. Z., Rizvi, S. S., and Arain, A. A. (2021). A novel luminance-based algorithm for classification of semi-dark images. *Applied Sciences*, 11(18):8694.

[71] Narasimhan, S. G. and Nayar, S. K. (2002). Vision and the atmosphere. *International journal of computer vision*, 48(3):233–254.

[72] Narasimhan, S. G. and Nayar, S. K. (2003). Contrast restoration of weather degraded images. *IEEE transactions on pattern analysis and machine intelligence*, 25(6):713–724.

[73] Padilla, R., Passos, W. L., Dias, T. L., Netto, S. L., and Da Silva, E. A. (2021). A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3):279.

[74] Patrikar, D. R. and Parate, M. R. (2022). Anomaly detection using edge computing in video surveillance system: review. *Int. J. Multimed. Inf. Retr.*

[75] Pham, V., Pham, C., and Dang, T. (2020). Road damage detection and classification with detectron2 and faster r-cnn. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5592–5601.

[76] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

[77] Piccardi, M. (2004). Background subtraction techniques: a review. In *2004 IEEE international conference on systems, man and cybernetics (IEEE Cat. No. 04CH37583)*, volume 4, pages 3099–3104. IEEE.

[78] Qu, Y., Chen, Y., Huang, J., and Xie, Y. (2019). Enhanced pix2pix dehazing network. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8160–8168.

[79] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, v1:234–241.

[80] Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1):157–173.

[81] Sara, U., Akter, M., and Uddin, M. S. (2019). Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 7(3):8–18.

[82] Schuler, C. J., Hirsch, M., Harmeling, S., and Schölkopf, B. (2015). Learning to deblur. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1439–1451.

[83] Shaik, A. B. and Srinivasan, S. (2019). A brief survey on random forest ensembles in classification model. In *International Conference on Innovative Computing and Communications*, pages 253–260. Springer.

[84] Shaw, S., Gupta, R., and Roy, S. (2020). A review on different image de-hazing methods. *Emerging Technology in Modelling and Graphics*, pages 533–540.

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

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

[87] Shwartz, S. and Schechner, Y. (2006). Blind haze separation. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2:1984–1991.

[88] Sivasubramaniam, N., Easwaramoorthy, D., and Abinands, R. (2020). An extensive review on recent evolutions in object detection algorithms. *International Journal of Emerging Trends in Engineering Research*, 8:3766–3776.

[89] Srivastava, S. and Singh, S. P. (2016). A survey on latency reduction approaches for performance optimization in cloud computing. In *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, pages 111–115. IEEE.

[90] Sun, F., Hu, D., and Liu, H. (2014). Foundations and practical applications of cognitive systems and information processing: Proceedings of the first international conference on cognitive systems and information processing, Beijing, China, Dec 2012 (CSIP2012). *Adv. Intell. Syst. Comput.*, 215.

[91] Takahashi, F. and Abe, S. (2002). Decision-tree-based multiclass support vector machines. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, volume 3, pages 1418–1422. IEEE.

[92] Tian, C., Xu, Y., Fei, L., and Yan, K. (2018). Deep learning for image denoising: A survey. *International Conference on Genetic and Evolutionary Computing*, pages 563–572.

[93] Tripathi, S., Lipton, Z. C., and Nguyen, T. Q. (2018). Correction by projection: Denoising images with generative adversarial networks. *arXiv preprint arXiv:1803.04477*.

[94] Walambe, R., Marathe, A., Kotecha, K., and Ghinea, G. (2021). Lightweight Object Detection Ensemble Framework for Autonomous Vehicles in Challenging Weather Conditions. *Comput. Intell. Neurosci.*, 2021.

[95] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

[96] Wang, R., Zhang, Q., Fu, C.-W., Shen, X., Zheng, W.-S., and Jia, J. (2019). Underexposed photo enhancement using deep illumination estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6849–6857.

[97] Wang, W., Wu, X., Yuan, X., and Gao, Z. (2020). An experiment-based review of low-light image enhancement methods. *Ieee Access*, 8:87884–87917.

[98] Waske, B., Schiefer, S., and Braun, M. (2006). Random feature selection for decision tree classification of multi-temporal sar data. In *2006 IEEE International Symposium on Geoscience and Remote Sensing*, pages 168–171.

[99] Wei, C., Wang, W., Yang, W., and Liu, J. (2018). Deep retinex decomposition for low-light enhancement. *arXiv preprint arXiv:1808.04560*.

[100] Welch, G., Bishop, G., et al. (1995). An introduction to the kalman filter. *Dl*.

[101] Whyte, O., Sivic, J., Zisserman, A., and Ponce, J. (2012). Non-uniform deblurring for shaken images. *International journal of computer vision*, 98(2):168–186.

[102] Wikipedia contributors (2022). Gsm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=GSM&oldid=1089309488. [Online; accessed 23-May-2022].

[103] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE.

[104] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. https://github.com/facebookresearch/detectron2.

[105] Xia, J., Xuan, D., Tan, L., and Xing, L. (2020). ResNet15: Weather Recognition on Traffic Road with Deep Convolutional Neural Network. *Adv. Meteorol.*, 2020.

[106] Xiao, H., Zhang, F., Shen, Z., Wu, K., and Zhang, J. (2021). Classification of Weather Phenomenon From Images by Using Deep Convolutional Neural Network. *Earth Sp. Sci.*, 8(5):1–9.

[107] Xu, J., Zhang, L., Zuo, W., Zhang, D., and Feng, X. (2015). Patch group based nonlocal self-similarity prior learning for image denoising. *Proceedings of the IEEE international conference on computer vision*, pages 244–252.

[108] Xu, K., Yang, X., Yin, B., and Lau, R. W. (2020). Learning to restore low-light images via decomposition-and-enhancement. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2281–2290.

[109] Yang, H.-M., Liu, C.-L., Liu, K.-H., and Huang, S.-M. (2003). Traffic sign recognition in disturbing environments. *International Symposium on Methodologies for Intelligent Systems*, pages 252–261.

[110] Yang, W., Wang, S., Fang, Y., Wang, Y., and Liu, J. (2020). From fidelity to perceptual quality: A semi-supervised approach for low-light image enhancement. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3063–3072.

[111] Yaser, A., Omer, R., and Charith, P. (2023). Anomaly detection on the edge using smart cameras under low-light conditions. *Sensors*. https://drive.google.com/file/d/1VuwB-A6gmwN9AGHTZme2b1Ls4Yf1gekp/view?usp=drive_link.

[112] Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32.

[113] Zeng, X. and Martinez, T. R. (2004). Feature weighting using neural networks. *IEEE Int. Conf. Neural Networks - Conf. Proc.*, 2:1327–1330.

[114] Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017). Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155.

[115] Zhang, L., Zhang, L., Liu, X., Shen, Y., Zhang, S., and Zhao, S. (2019a). Zero-shot restoration of back-lit images using deep internal learning. *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1623–1631.

[116] Zhang, Y., Guo, X., Ma, J., Liu, W., and Zhang, J. (2021). Beyond brightening low-light images. *International Journal of Computer Vision*, 129(4):1013–1037.

[117] Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., and Fu, Y. (2018). Image super-resolution using very deep residual channel attention networks. *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301.

[118] Zhang, Y., Zhang, J., and Guo, X. (2019b). Kindling the darkness: A practical low-light image enhancer. *Proceedings of the 27th ACM international conference on multimedia*, pages 1632–1640.

[119] Zhang, Z. and Ma, H. (2015). MULTI-CLASS WEATHER CLASSIFICATION ON SINGLE IMAGES Zheng Zhang , Huadong Ma Beijing Key Lab of Intelligent Telecomm . Software and Multimedia Beijing University of Posts and Telecomm ., Beijing 100876 , China. *2015 IEEE Int. Conf. Image Process.*, pages 4396–4400.

[120] Zhao, B., Li, X., Lu, X., and Wang, Z. (2018a). A CNN–RNN architecture for multi-label weather recognition. *Neurocomputing*, 322:47–57.

[121] Zhao, X., Liu, P., Liu, J., and Tang, X. (2018b). Real-Time Human Objects Tracking for Smart Surveillance at the Edge. *IEEE Int. Conf. Commun.*, 2018-May(May).

[122] Zhao, X., Liu, P., Liu, J., and Tang, X. (2021a). Toward Intelligent Surveillance as an Edge Network Service (iSENSE) Using Lightweight Detection and Tracking Algorithms. *IEEE Trans. Serv. Comput.*, 14(6):1624–1637.

[123] Zhao, Y., Yin, Y., and Gui, G. (2020). Lightweight Deep Learning Based Intelligent Edge Surveillance Techniques. *IEEE Trans. Cogn. Commun. Netw.*, 6(4):1146–1154.

[124] Zhao, Z., Xiong, B., Wang, L., Ou, Q., Yu, L., and Kuang, F. (2021b). Retinexdip: A unified deep framework for low-light image enhancement. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1076–1088.

[125] Zhao, Z.-Q., Zheng, P., Xu, S.-T., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232.

[126] Zheng, C., Shi, D., and Shi, W. (2021). Adaptive unfolding total variation network for low-light image enhancement. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021*, pages 4439–4448.

[127] Zhu, A., Zhang, L., Shen, Y., Ma, Y., Zhao, S., and Zhou, Y. (2020). Zero-shot restoration of underexposed images via robust retinex decomposition. *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.

[128] Zhu, M. (2004). Recall, precision and average precision. *Department of Statistics and Actuarial Science*, 2.

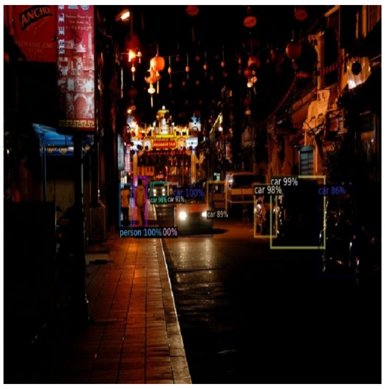
[129] Zhu, Z., Zhuo, L., Qu, P., Zhou, K., and Zhang, J. (2017). Extreme weather recognition using convolutional neural networks. *Proc. - 2016 IEEE Int. Symp. Multimedia, ISM 2016*, pages 621–625.

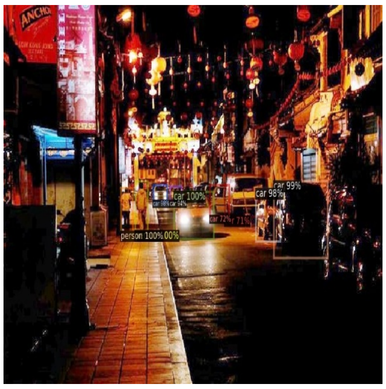
[130] Zunin, V. (2021). Intel openvino toolkit for computer vision: Object detection and semantic segmentation. In *2021 International Russian Automation Conference (RusAutoCon)*, pages 847–851. IEEE.

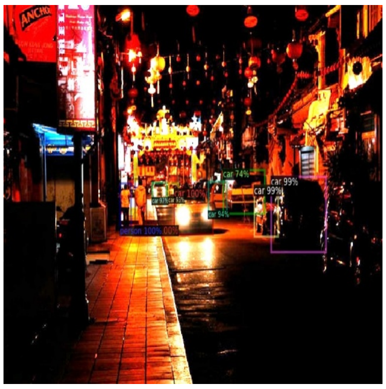
# Appendix A

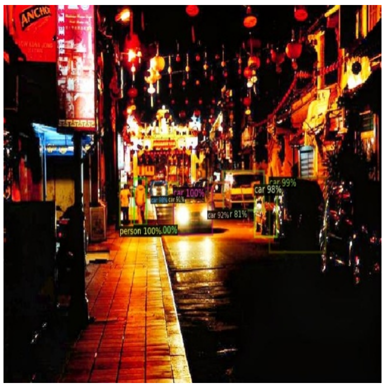
In this section, additional results are presented for better comprehension of how image enhancement methods help achieve higher accuracy for the detection stage compared to direct detection (before enhancement) and the ground truth itself. It is worth mentioning that only best practice methods were selected and considered for the following examples. Figures A.1, A.2, A.3 & A.4 demonstrate randomly selected samples to examine the approach's functionality and capabilities. After more investigation, the optimal technique is shown by the green box surrounding the model names, which provide extra correct predictions compared to ground truth and direct detection; see the figure's captions. Moreover, the results regarding several predictions for each method are presented in the following figures A.5, A.6, A.7 & A.8.

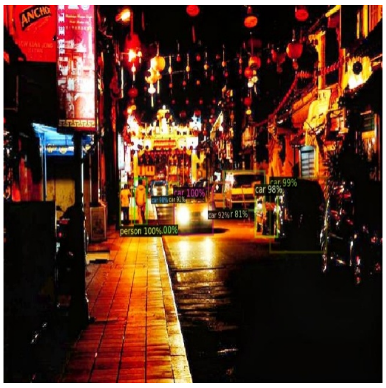
The bars coloured *"Green"* represent the ground truth *(GT)*, and *"Orange"* represent detection before enhancement *(Direct)*. On the other hand, the remaining coloured *"Blue"* represent low-light image enhancement models. In addition, the total number of predictions is divided into two numbers, a top and bottom value. The top and bottom values correspond to the number of *"Car"* and *"Person"* instances within an image, respectively. While Figure A.7 only represent the *"Person"* class for all instances. It can be noted that objects are correctly detected and discovered in comparison to when no enhancement is applied, as well as to original predictions. Moreover, the outperformed methods among all samples differ, indicating that a single method might perform in a specific context and condition. Thus, this emphasises the need for these methods to operate together instead of relying on only one in order to recognise objects in low-light environments with various circumstances.

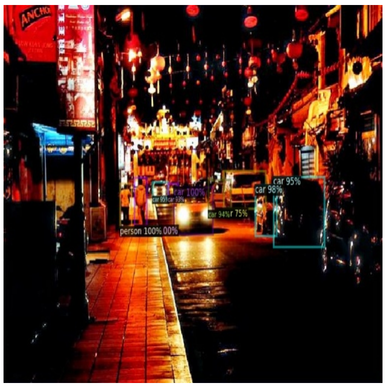

**(a)** Original (Direct).

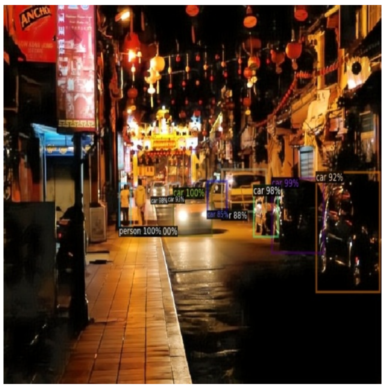

**(b)** Zero-DCE++.

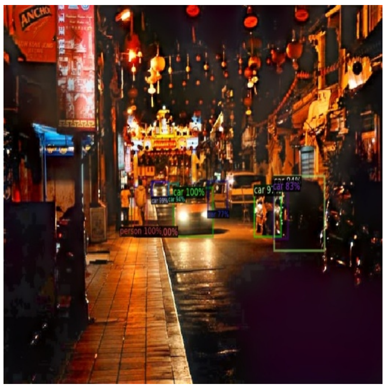

**(c)** RUSA.

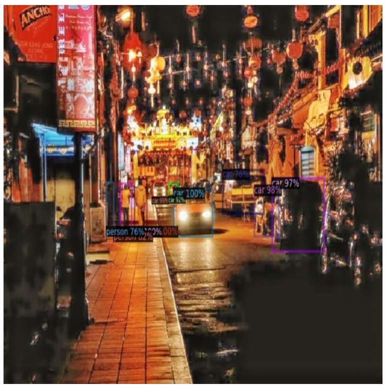

**(d)** CSDNet_UPE.

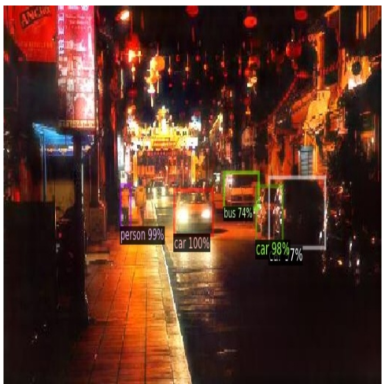

**(e)** Lite_CSDNet_UPE.

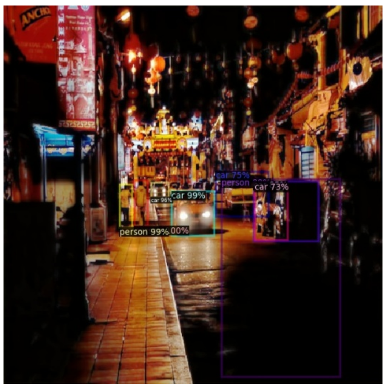

**(f)** SLite_CSDNet_UPE.


**(g)** **MBLLEN.**


**(h)** ElightenGAN.


**(i)** TBEFN.


**(j)** DSLR.


**(k)** KinD.

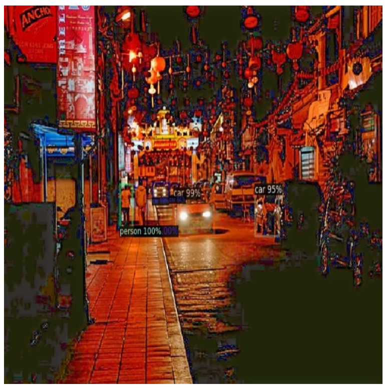
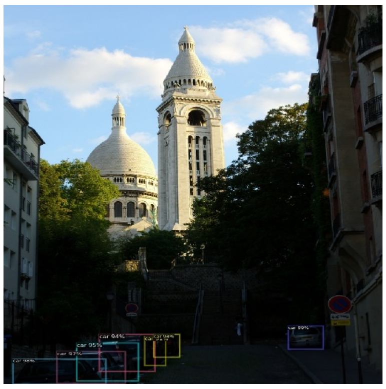
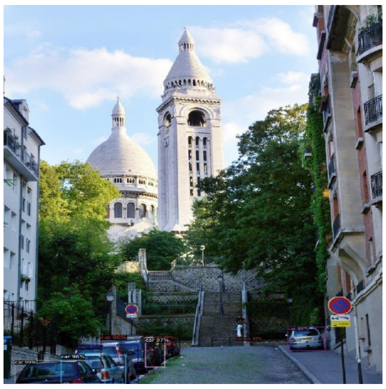
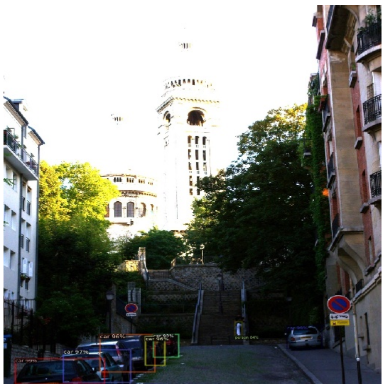
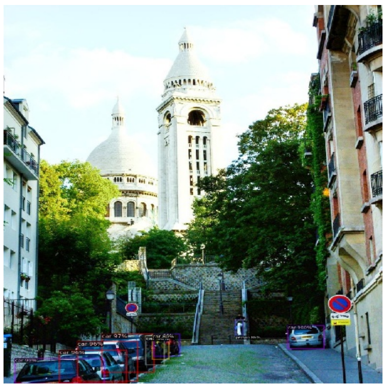

**(l)** RetinexNet.

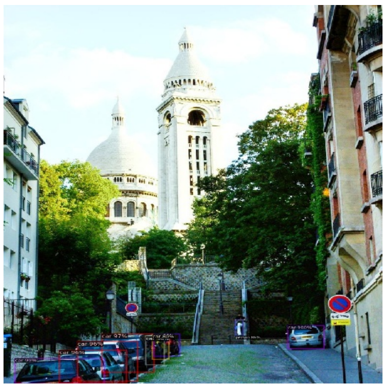
**Figure A.1** Comparison of model enhancement for the detection stage on sample *"2015_02448.jpg"*.

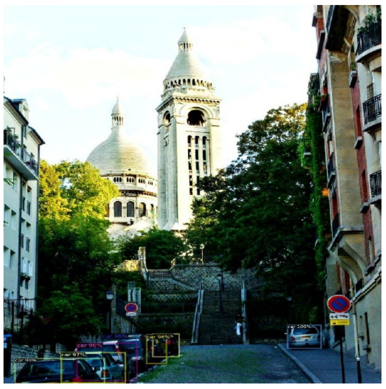

**(a)** Original (Direct).

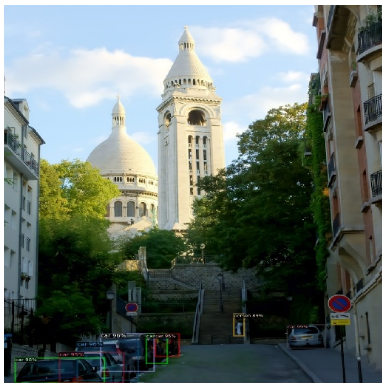

**(b)** Zero-DCE++.

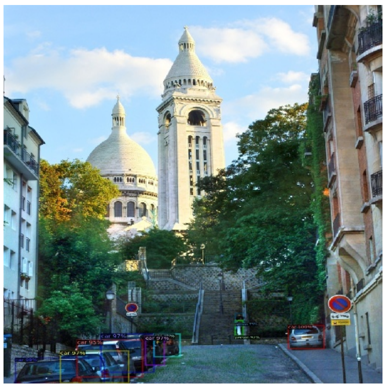

**(c)** RUSA.

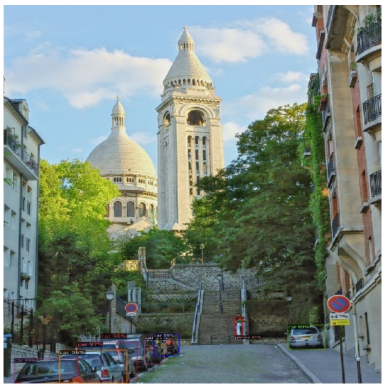

**(d)** **CSDNet_UPE.**

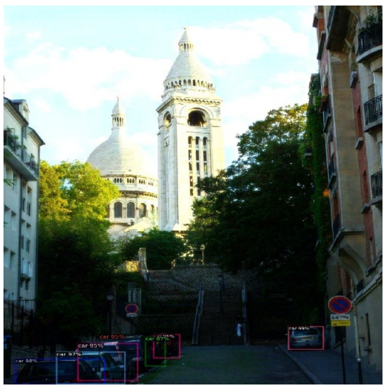

**(e)** Lite_CSDNet_UPE.

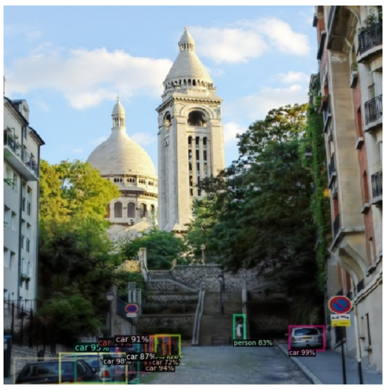

**(f)** SLite_CSDNet_UPE.


**(g)** MBLLEN.


**(h)** ElightenGAN.


**(i)** TBEFN.


**(j)** DSLR.


**(k)** KinD.

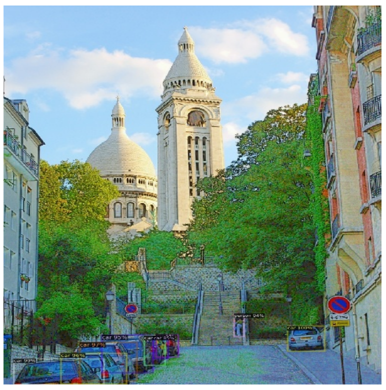
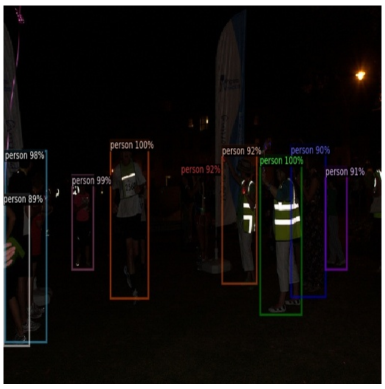
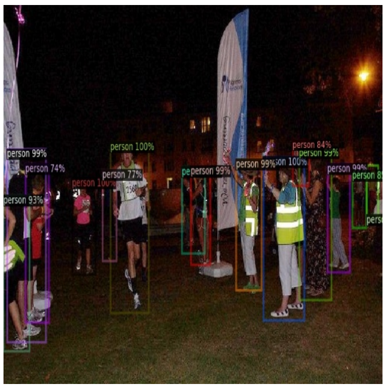
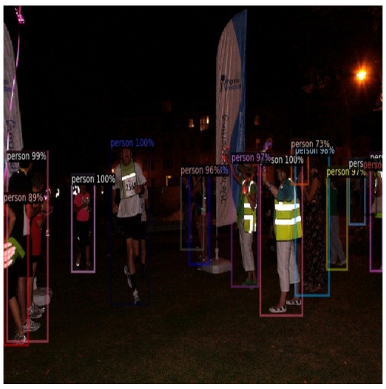
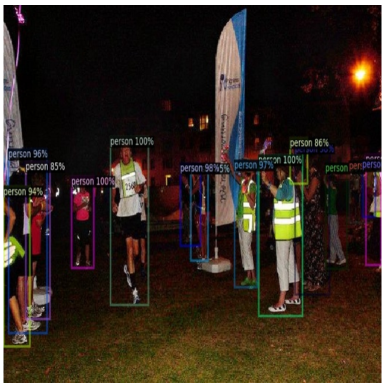

**(l)** RetinexNet.

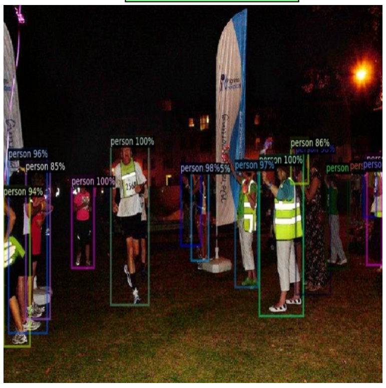
**Figure A.2** Comparison of model enhancement for the detection stage on sample *"2015_02926.jpg"*.

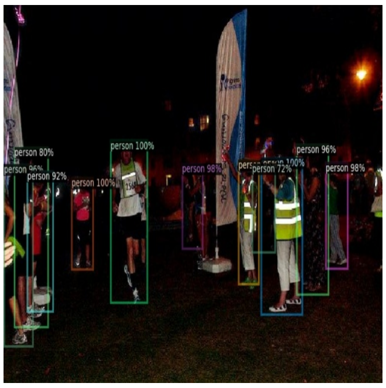

(**a**) Original (Direct).

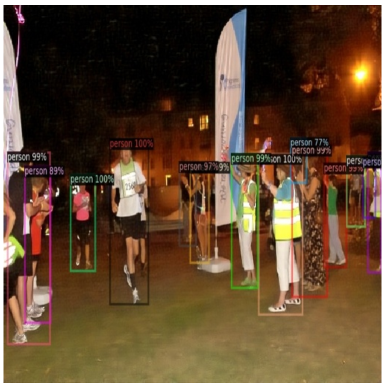

(**b**) **Zero-DCE++.**

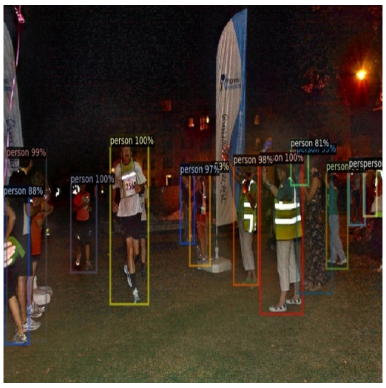

(**c**) RUSA.

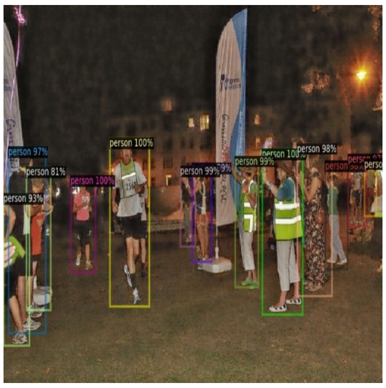

(**d**) CSDNet_UPE.

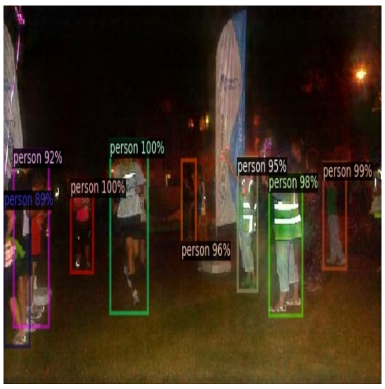

(**e**) Lite_CSDNet_UPE.

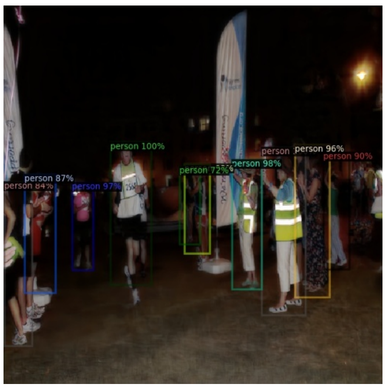

(**f**) SLite_CSDNet_UPE.


(**g**) MBLLEN.


(**h**) ElightenGAN.


(**i**) TBEFN.


(**j**) DSLR.


(**k**) KinD.

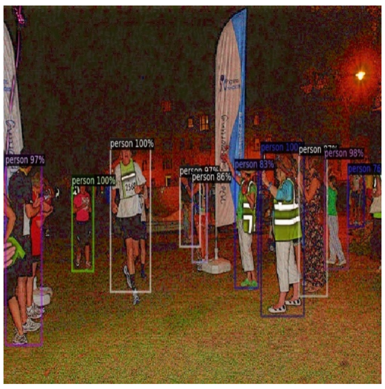
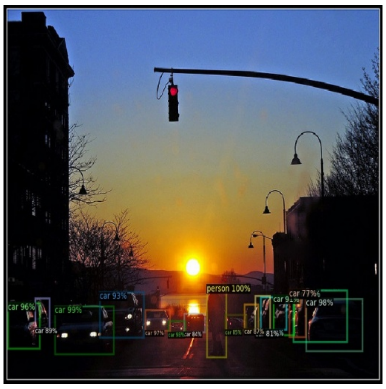
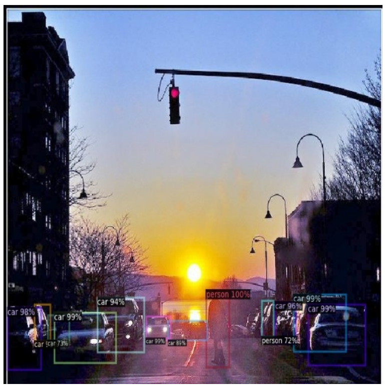
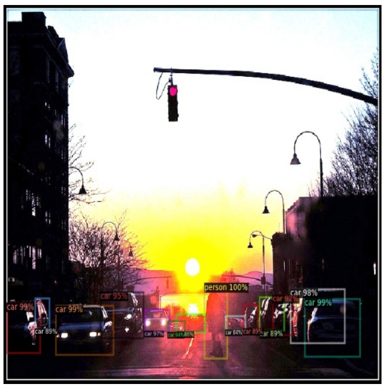
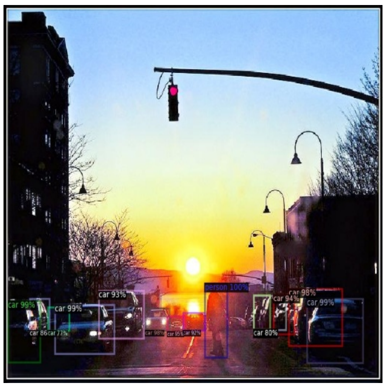

(**l**) RetinexNet.

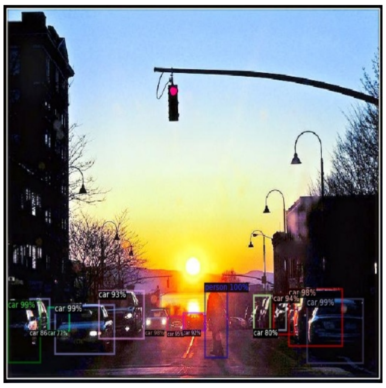
**Figure A.3** Comparison of model enhancement for the detection stage on sample *"2015_06339.jpg"*.

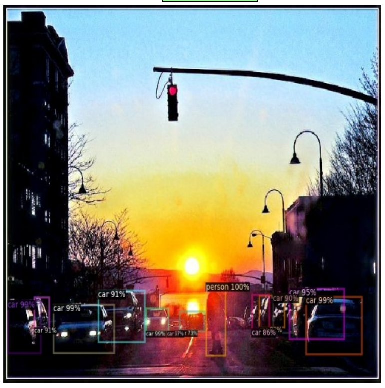

**(a)** Original (Direct).

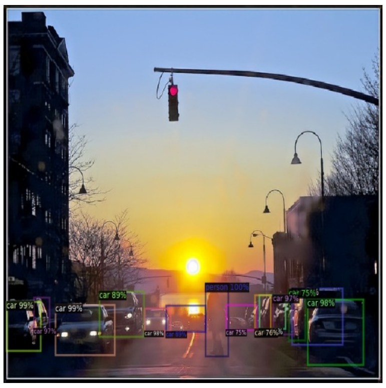

**(b)** Zero-DCE++.

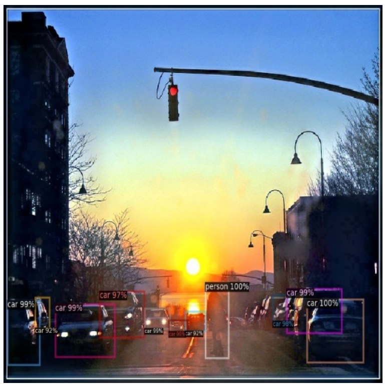

**(c)** **RUSA.**

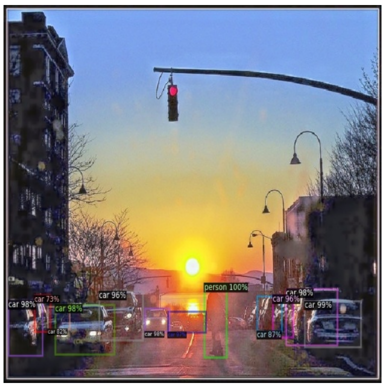

**(d)** CSDNet_UPE.

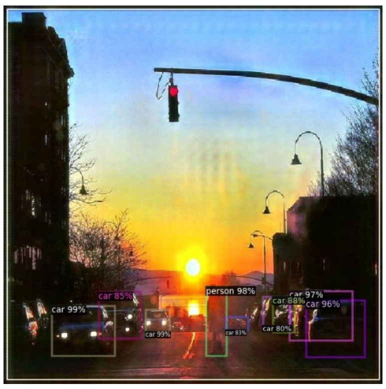

**(e)** Lite_CSDNet_UPE.

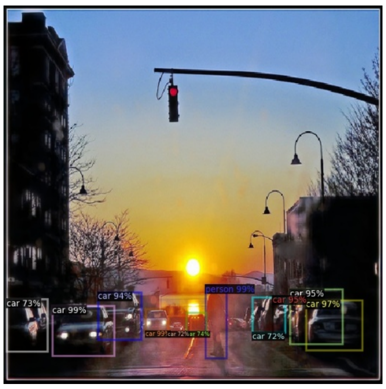

**(f)** SLite_CSDNet_UPE.


**(g)** MBLLEN.


**(h)** ElightenGAN.


**(i)** TBEFN.


**(j)** DSLR.


**(k)** KinD.

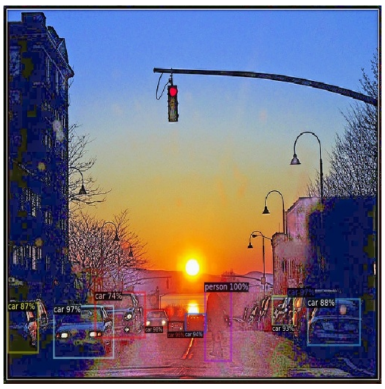

**(l)** RetinexNet.

**Figure A.4** Comparison of model enhancement for the detection stage on sample *"2015_06574.jpg"*.

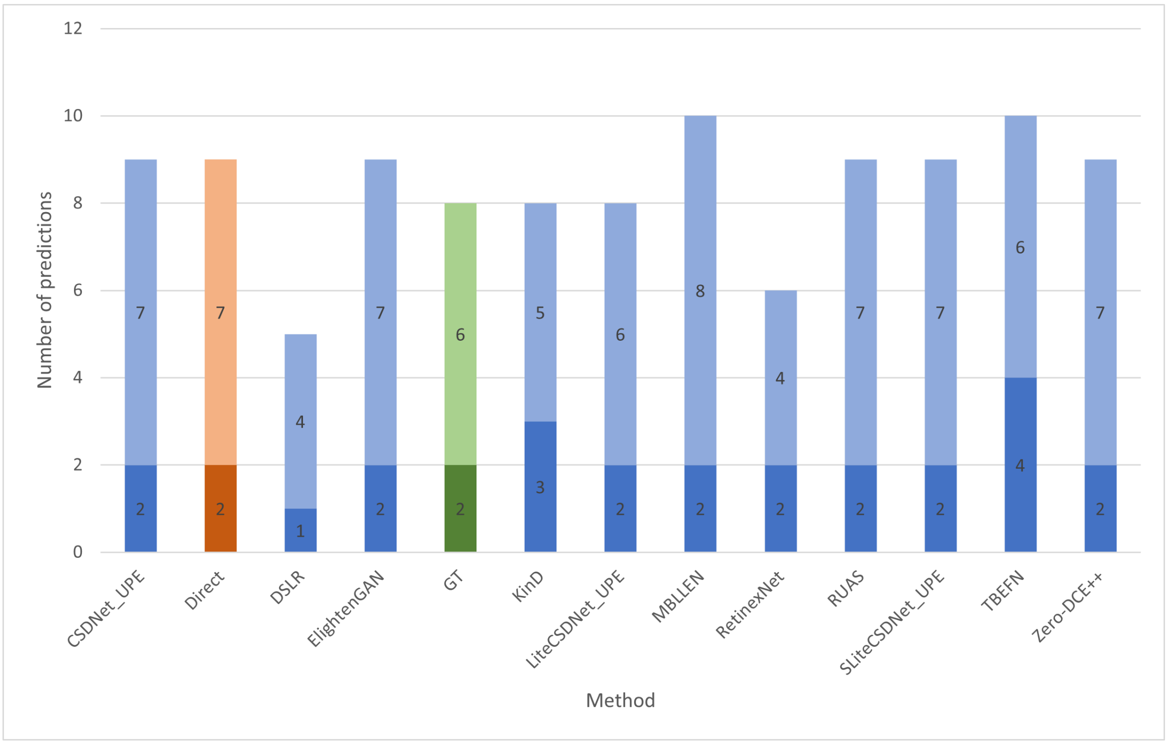


**Figure A.5** Ground Truth vs Direct vs After Enhancement for sample *"2015_02448.jpg"*.

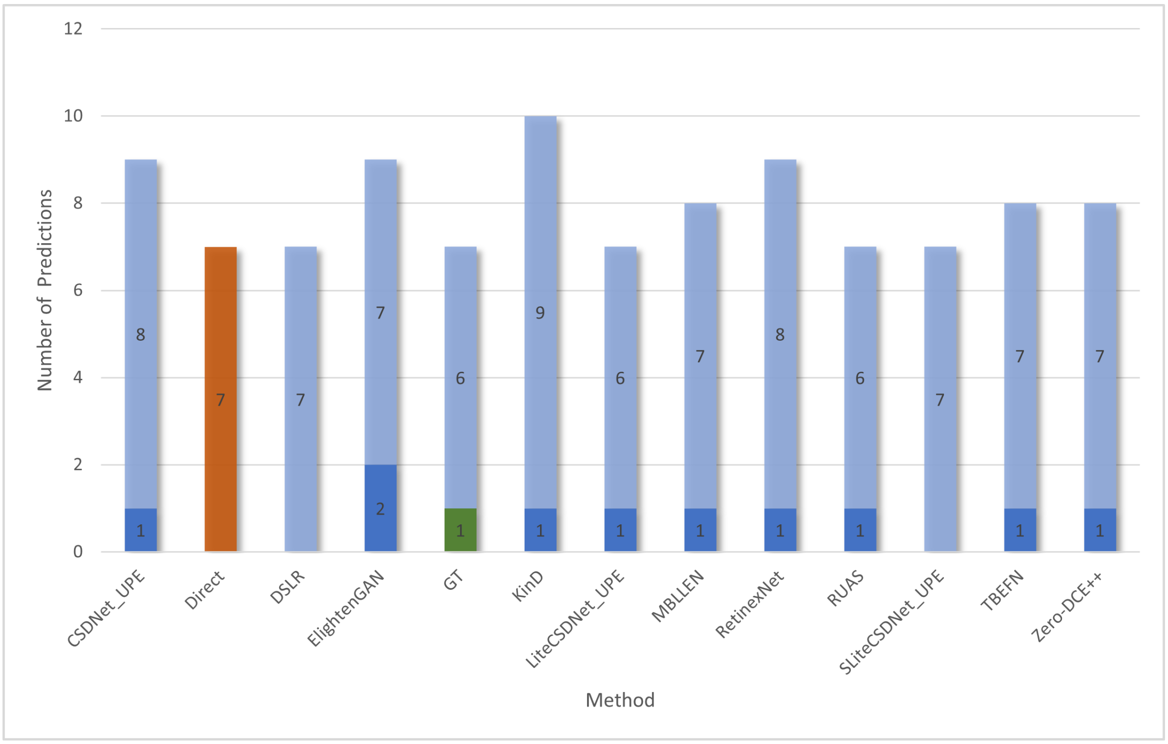


**Figure A.6** Ground Truth vs Direct vs After Enhancement for sample *"2015_02926.jpg"*.

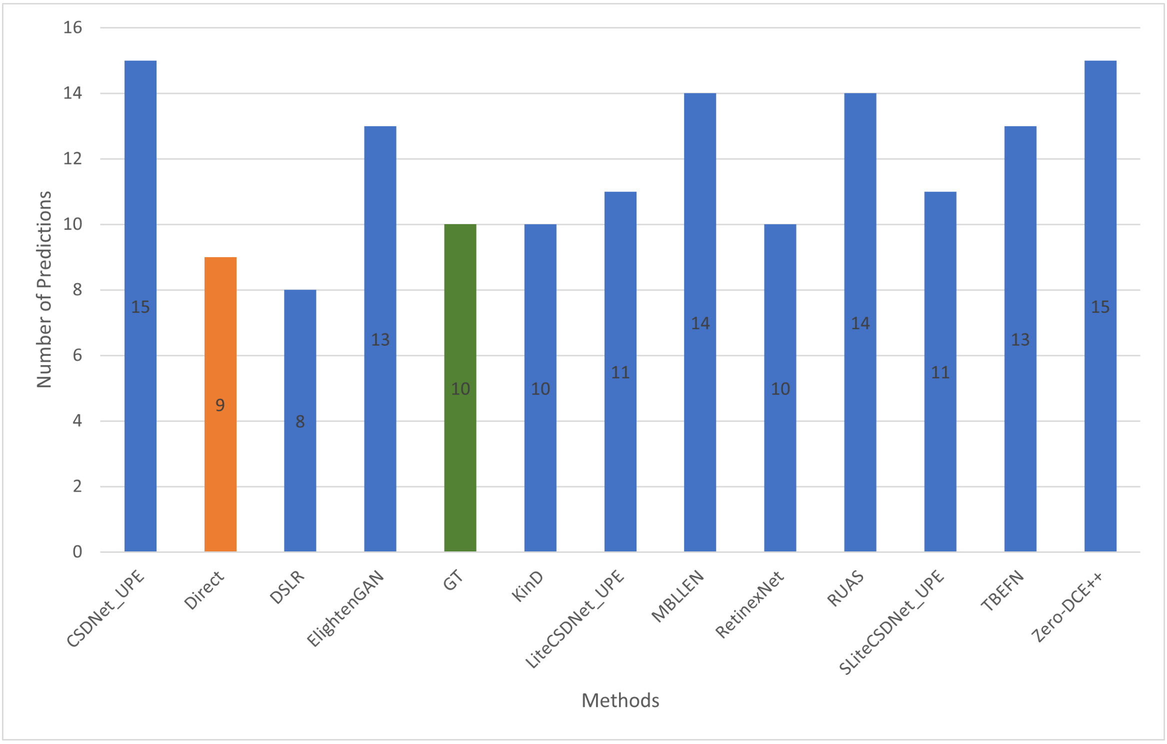


**Figure A.7** Ground Truth vs Direct vs After Enhancement for sample *"2015_06339.jpg"*.

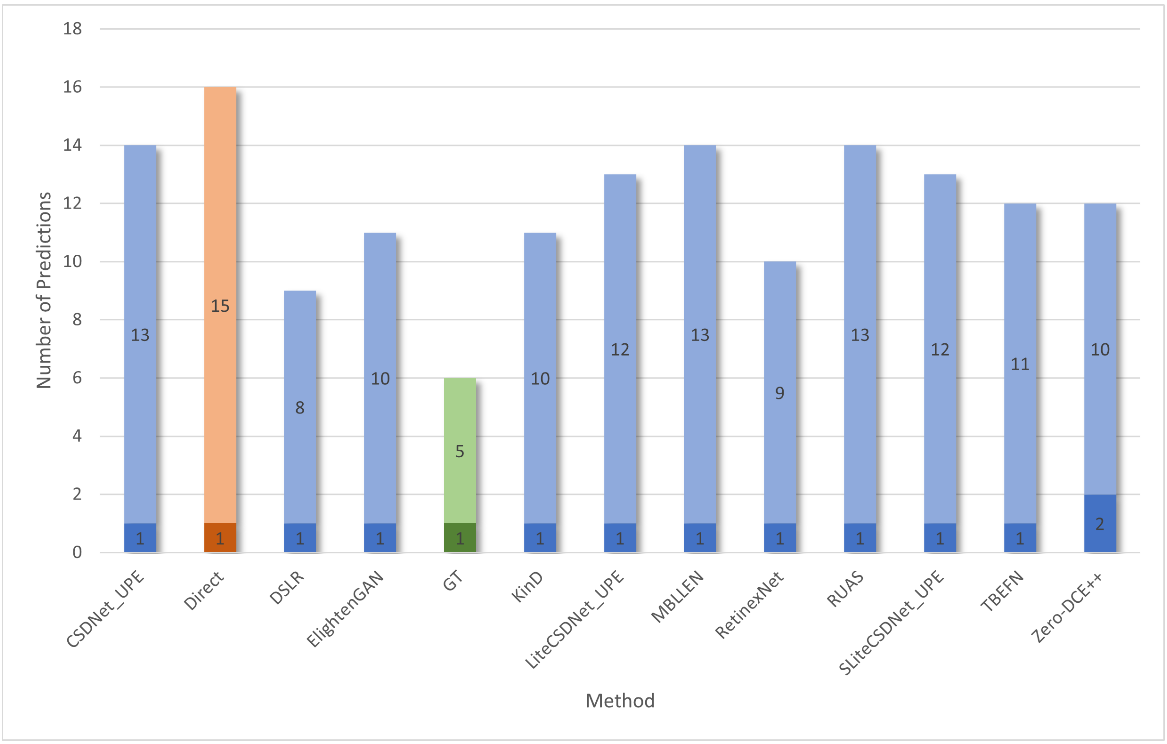


**Figure A.8** Ground Truth vs Direct vs After Enhancement for sample *"2015_06574.jpg"*.