MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
Faculty of Aeronautics, Electronics and Telecommunications
Department of aviation computer-integrated complexes

ADMIT TO DEFENSE
Head of the graduation department
_____ Viktor SINEGLAZOV
“___”_____2024

QUALIFICATION WORK
(EXPLANATORY NOTE)

GRADUATE DEGREE OF EDUCATION

“BACHELOR”

Specialty 151 "Automation and computer-integrated technologies"
Educational and professional program "Computer-integrated technological processes and production"

Topic: Intelligent object recognition system from a night vision camera

Performer: student of FAET-402 group Volovyk Serhii Volodymyrovych

Supervisor: senior teacher Dolhorukov Serhii Olehovych

Norm controller: _____ Filiashkin M.K
(sign)

Kiyv – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп'ютерно-інтегрованих систем

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри

_____ Віктор СИНЄГЛАЗОВ

“____” _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

“БАКАЛАВР”

Спеціалність 151 "Автоматизація, та комп’ютерно-інтегровані технології"

Освітньо-професійна програма "Комп’ютерно-інтегровані технологічні процеси і виробництва"

Тема: Інтелектуальні системи розпізнавання об'єктів з камери нічного бачення

Виконавець: студент групи КП-402Ба Воловик Сергій Володимирович

Керівник: канд. техн. Наук Долгоруков Сергій Олегович

Нормоконтроллер: _____ Філяшкін М.К.

(підпис)

Київ – 2024

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL AVIATION UNIVERSITY

Faculty of Aeronautics, Electronics and Telecommunications

Department of aviation computer-integrated complexes

Educational degree: bachelor's degree
Specialty 151 "Automation and computer-integrated technologies"
Educational and professional program "Computer-integrated technological processes and production"

APPROVED

Head of the Department

_____ Viktor SINEGLAZOV

“___”_____2024

TASK

for the performance of the student's qualification work

Volovyk Serhii Volodymyrovych

1. The topic of the work: «Intelligent object recognition system from a night vision camera»

2. The term of the work: from 05.22.2024 to 06.14.2024

3. Initial data for work: Development of a model for object recognition in images from night vision cameras.

4. Content of the explanatory note (list of issues to be developed): 1. Analysis of object recognition systems and their application from a night vision camera; 2. Analysis of existing software; 3. Analysis of existing algorithms and structures; 4. Development of algorithms for tracking moving objects, face recognition and other identification methods; 5. Development and research of image transformation methods;

5. List of mandatory graphic material: 1. Structural diagram of information protection in intelligent systems. 2. Structural diagram of connections of a single-layer neural network; 3. Structural diagrams of construction and connection of IP video surveillance; 4. Structural diagrams of neural network algorithms; 5. Laws of control and a structural diagram of the control circuit of the flight according to the route by the course method; 6. Results of the evaluation of the effectiveness of the developed model.

6. Calendar schedule:

№	Task	Deadline	Performance note
1.	Receiving the task	22.04.2024	Done
2.	Formation of the goal and main tasks of the research	23.04.2024	Done
3.	Analysis of the relevance of the problem	24.04.2024 – 25.04.2024	Done
4.	Analysis of the existing software related to the network video surveillance and security equipment market in the world	26.04.2024 – 27.04.2024	Done
5.	Research on the information support of object recognition from a night vision camera	27.04.2024 – 29.04.2024	Done
6.	Development and research of a deep learning model based on a neural network	30.04.2024 – 06.05.2024	Done
7.	Development of a model for object recognition on images from a night vision camera	07.05.2024 – 11.05.2024	Done
8.	Conclusions on work and preparation presentations and handouts	12.05.2024 – 13.05.2024	Done

7. Task issue date: _____

Supervisor**:** _____ Dolhorukov S. O.

(sign)

Task is taken for completion by**:** _____ Volovyk S. V.

(sign)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

Освітній ступінь: Бакалавр

Спеціальність 151 "Автоматизація та комп'ютерно-інтегровані технології"

Освітньо-професійна програма "Комп'ютерно-інтегровані технологічні процеси і виробництва"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віктор СИНЄГЛАЗОВ

“ ____” _____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Воловик Сергій Володимирович

1. Тема роботи: «Інтелектуальна система розпізнавання об'єктів з камери нічного бачення»

2. Термін виконання робіт: з 22.05.2024 по 14.06.2024

3. Вихідні дані для роботи: Розробка моделі розпізнавання об'єктів на зображеннях з камер нічного бачення.

4. Зміст пояснювальної записки (перелік питань для розробки):
1. Аналіз систем розпізнавання об'єктів та їх застосування з камери нічного бачення; 2. Аналіз існуючого програмного забезпечення; 3. Аналіз існуючих алгоритмів і структур; 4. Розробка алгоритмів відстеження рухомих об'єктів, розпізнавання осіб та інших методів ідентифікації; 5. Розробка та дослідження методів трансформації зображення;

5. Перелік обов'язкових графічних матеріалів: 1. Структурна схема захисту інформації в інтелектуальних системах. 2. Структурна схема зв'язків одношарової нейронної мережі; 3. Структурні схеми побудови та підключення IP відеоспостереження; 4. Структурні схеми нейромережевих алгоритмів; 6. Результати оцінки ефективності розробленої моделі.

6. Календарний план-графік:

№	Завдання	Термін виконання	Відмітка про виконання
1.	Отримання завдання	22.04.2024	Виконано
2.	Формування мети та основних завдань дослідження	23.04.2024	Виконано
3.	Аналіз актуальності проблеми	24.04.2024 – 25.04.2024	Виконано
4.	Аналіз існуючого програмного забезпечення, що стосується ринку мережевого обладнання для відеоспостереження та безпеки у світі	26.04.2024 – 27.04.2024	Виконано
5.	Дослідження інформаційного забезпечення розпізнавання об’єктів з камери нічного бачення	27.04.2024 – 29.04.2024	Виконано
6.	Розробка та дослідження моделі глибокого навчання на основі нейронної мережі	30.04.2024 – 06.05.2024	Виконано
7.	Розробка моделі розпізнавання об'єктів на зображеннях з камери нічного бачення	07.05.2024 – 11.05.2024	Виконано
8.	Висновки по роботі та підготовці презентації та роздатковий матеріал	12.05.2024 – 13.05.2024	Виконано

7. Дата видачі завдання _____

Керівник: _____ Долгоруков С. О.

(підпис)

Завдання прийняв до виконання: _____ Воловик С. В.

(підпис)

ABSTRACT

Explanatory note of the qualification work "Intelligent object recognition system from a night vision camera"

NIGHT VISION CAMERA, NEURAL NETWORK, CONVOLUTIONAL NEURAL NETWORKS (CNN), YOLO (YOU ONLY LOOK ONCE) ARCHITECTURE.

The object of research is intelligent object recognition systems in special conditions.

The subject of research is methods and algorithms for recognizing moving objects

Objects for night video surveillance systems.

The purpose of the qualification work is to research developments in improving the efficiency of object recognition in intelligent night video surveillance systems due to the use of new and modified algorithms.

The research method is mathematical modeling of the researched processes, system analysis, object-oriented programming.

Theoretical studies consisted of the analysis of the work algorithms of the most famous methods. Popular software systems are also overclocked. A model of an improved lighting control scheme based on motion sensors, an existing night vision system, and algorithms for adaptive resolution of camera images to increase resolution is analyzed.

The research results showed that the use of effective and fast image preprocessing methods allowed to improve the efficiency of the models without significant resource costs, and also contributed to the minimization of the total costs of project implementation. In general, cost minimization made it possible to successfully reduce project implementation costs, while ensuring high quality and efficiency of the developed system.

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи «Інтелектуальна система розпізнавання об`єктів з камери нічного бачення»

КАМЕРА НІЧНОГО БАЧЕННЯ, НЕЙРОННА МЕРЕЖА, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ (CNN), АРХІТЕКТУРА YOLO (YOU ONLY LOOK ONCE).

Об'єкт дослідження – інтелектуальні системи розпізнавання об’єктів

в особливих умовах.

Предмет дослідження – методи та алгоритми розпізнавання рухомих

Об`єктів для систем нічного відеоспостереження.

Мета кваліфікаційної роботи – дослідження розробок щодо підвищення ефективності розпізнавання об`єктів у інтелектуальних системах нічного відеоспостереження за рахунок застосування нових та модифікованих алгоритмів.

Метод дослідження – математичне моделювання досліджуваних процесів, системний аналіз, об`єктно-орієнтоване програмування.

Теоретичні дослідження складалися з аналізу алгоритмів роботи найвідоміших методів. Також, розгянуто популярні програмні системи. Проаналізована модель покращеної схеми управління освітленням на основі датчиків руху, існуючої системи нічного бачення та алгоритми адаптивної роздільної здатності зображень з камери для збільшення роздільної здатності.

Результати досліджень показали, що Використання ефективних та швидких методів попередньої обробки зображень дозволили покращити ефективність моделей без значних витрат на ресурси, також сприяло мінімізації загальних витрат на реалізацію проекту. Загалом, мінімізація витрат дозволила успішно знизити витрати на реалізацію проекту, забезпечивши при цьому високу якість та ефективність розробленої системи.

CONTENTS

TOCOHZU

LIST OF SYMBOLS

API - Application Programming Interface

DNS - Domain Name System

DFD - Data Flow Diagrams

IDEF - Integrated Computer-Aided Manufacturing

IT - Information Technology

IM - Instant Messaging

MDM - Mobile device management

MCM - Mobile Content Management

MAM - Mobile Application Management

MC - Microsoft Corporation

OS - Operation System

PHP - Hypertext Preprocessor

SLL - Secure Sockets Layer

SMB - Server Message Block

Win. NT - Windows New Technology

WBS - Work Breakdown Structure

WMI - Windows Management Instrumentation

AC - Automated system

Computer - Electronic computing machine

RAM - Random Access Memory

DBMS - Database management system

PP - Software product

HEI - Higher education institution

PS - Software environment

ANN - Artificial neural networks

ICMP - Internet Control Message Protocol

EDS - Electronic digital signature

SSH - Secure Shell

EMM - Enterprise Mobility Management

UML - Unified Modelling Language

JPEG - Joint Photographic Experts Group

GIF - Raphics Interchange Format

Relevance of the topic. Intelligent object recognition systems from night vision cameras are an important element in various industries, such as security, transport surveillance, animal monitoring, and others. They use computer vision and artificial intelligence technologies to detect, classify and track objects even in low-light conditions. One of the problems associated with the criteria for evaluating object recognition from night vision camera systems was the lack of understanding of the network's mechanisms. In view of the above, the use of new methods to improve the security of electronic document management systems will not lose its relevance for a long time to come.

Relationship of the work to scientific programmes, plans, topics. Numerous works of leading scientists around the world are devoted to the study of various theoretical and practical aspects of information security problems, methods of building secure computer systems. The work on the development and improvement of object recognition systems from night vision cameras is closely related to computer vision, artificial intelligence and image processing.

Many researchers, engineers, and companies are developing and improving object recognition systems using different technologies and approaches. Some well-known researchers: Google Research team, Microsoft Research object recognition research programme, Facebook AI (in the field of computer vision and machine learning) Research (FAIR). Thus, the work on the development of object recognition systems from night vision cameras can have a wide range of scientific applications and links to various research programmes and topics. The work was carried out in accordance with the scientific works of world researchers and their dissertation studies on the problems of creating complex object recognition systems.

Purpose and objectives of the study. The purpose of the qualification work (project) is to study developments to improve the efficiency of object recognition in intelligent night video surveillance systems through the use of new and modified algorithms. To achieve this goal, it is necessary to complete a number of tasks:

1. To analyse the existing methods of image recognition of object recognition systems at night;

2. To understand the features of building intelligent object recognition systems in special conditions;

3. Review the algorithms for adaptive post-processing, taking into account the determination of the type of light source using intelligent video analysis;

4. Analysis of existing software in a similar field;

5. Comparability of existing algorithms and methods of information processing in intelligent systems, taking into account the influence of external conditions;

6. Implement the software modules of the intelligent video surveillance system using the proposed algorithms;

7. Apply adaptive resolution algorithms to camera images to increase resolution;

8. Testing the recognition of people and objects in these images using intelligent video analysis.

The object of research is intelligent object recognition systems in special conditions.

The subject of the research is methods and algorithms for recognising moving objects for night video surveillance systems.

The **research methods** include mathematical modelling of the studied processes, system analysis, and object-oriented programming. Modern application software packages MatLab and the Python development environment were used. Experimental studies of the developed algorithms were also carried out on a real-time video surveillance system.

The scientific novelty of the obtained results is determined by the improvement of existing models and methods for object recognition in a video stream, which adapts to the characteristics of the observed scene. In the course of the work, the algorithms of the most famous methods were analysed. Also, popular software systems were reviewed and it was found that each of the above software products has advantages and certain disadvantages, which was taken into account when creating an integrated approach and improving the algorithm to improve the system.

1. The algorithms for adaptive post-processing are developed, taking into account the determination of the type of lighting source using intelligent video analysis.

2. A model, algorithms, and an improved lighting control scheme based on motion sensors and an existing night vision system are proposed.

3. The algorithms for adaptive resolution of camera images were applied to increase the resolution and tested.

The practical significance of the obtained results is to study the best algorithms for an intelligent system in identifying faces, recognising moving and stationary objects at night. The obtained scientific results are an important factor for improving the operation of video surveillance cameras in special conditions (at night) in computer systems and networks circulating in the local network of the institution.

1.1. History of the development of intelligent object recognition systems

An intelligent information system is one of the types of automated information systems based on software, linguistic and logical-mathematical tools to support human activity and search for necessary data in an advanced mode [1]. The functioning of an intelligent system can be described as continuous decision-making based on the analysis of current situations to achieve a certain goal. Today, people regularly face the task of recognising images and objects. The information coming from the senses is processed by the human brain, which in turn analyses the information and ensures decision-making. Then, using electrochemical impulses, the brain transmits the necessary signal, for example, to the motor organs, which perform the necessary actions. After that, the environment around the person changes, and these actions are repeated again.

The development of information technology has made it possible to solve a number of problems that arise in the course of people's lives, and they help to facilitate, accelerate and improve the quality of the result of actions. For example, the operation of various life support systems, human-computer interaction, and the emergence of robotic systems [2]. The task of object recognition has long been considered and studied by humans as a biological and psychological aspect, with only qualitative characteristics studied that did not provide an accurate description of the mechanism of functioning. The study of the receptors of the organs of hearing, touch and vision is associated with the acquisition of functional dependencies. As a result of the theory of static decisions, algorithms were found that ensure that a new object is assigned to one of the specified classes, which was the beginning of a systematic scientific search and practical development in pattern recognition systems. After the emergence of the science of "cybernetics", a new scientific field began to develop, which was associated with the development of theoretical and practical principles in the implementation of devices and systems designed to recognise objects and images. Thus, in the 70s, a new scientific field emerged, which was called "Pattern Recognition".

1.2. Characteristics of objects to be recognised in relation to the time of day

The characteristics of objects for cameras to recognise, especially with regard to the time of day, may depend on the specific application of the video surveillance system. However, general characteristics that can be useful for object recognition at different times of day include

1. Light sensitivity. Cameras should have a light sensitivity setting to ensure that they deliver high-quality images in daylight, low-light or nighttime conditions.

2. IR illumination (infrared illumination). This feature allows cameras to capture images in low-light conditions using infrared light. At night, objects can be recognised thanks to the IR illumination.

3. High resolution. To ensure high-quality object detection, especially at night, the camera must have a high resolution.

4. Automatic exposure and white balance correction. Cameras should have automatic exposure and white balance correction to adapt to changing light conditions throughout the day.

5. Wide Dynamic Range (WDR). This feature allows cameras to preserve details in both bright and dark areas of an image, which is important for recognising objects in varying light conditions.

6. Image denoising algorithms (DNR). They help to reduce noise and improve image quality, which is also useful for recognising objects in low-light conditions.

7. Object tracking algorithms. Some cameras have built-in algorithms for tracking object movement, which can help you recognise and track objects even at different times of the day.

8. Depending on the specific application, there may be other characteristics that are important. For example, for video surveillance systems in transport, it may be important to detail the number plates, which may require specialised cameras or recognition algorithms.

1.3. General principles of object recognition

The results of the theory of static solutions served as the basis for recognising patterns and objects and assigning them to a particular class and type. Later on, the static

(mathematical) apparatus of this theory of pattern recognition was divided into the following areas: applied mathematics, information theory, methods of logic algebra, mathematical programming and system engineering [3]. Pattern recognition (of phenomena, processes, objects, signals and situations) is understood as the identification of an object or the determination of any of its properties with respect to its image or audio recording (acoustic recognition), as well as other characteristics. An image is a classification grouping in a classification system that unites (distinguishes) a certain group of objects by some feature. Each physical object has characteristic properties, which are manifested in the fact that familiarisation with a finite number of phenomena from the same set makes it possible to recognise as many of its representatives as desired. Also, images have characteristic objective properties, i.e. different performers (people), independently of each other, classify the same object in the same way, even though they have been trained at different levels and materials. The way an element is assigned to a particular type or image is called a decisive rule. There is another important concept, a metric, which is a method of determining the distance between elements of a universal set. The smaller this distance is, the more similar the objects (sounds, symbols, phenomena, images, etc.) are. Most often, the elements of a metric are defined as a set of numbers, and the metric is defined as a function.

1.4. The role of pattern recognition and classification methods in intelligent night vision systems

The ability to recognise is considered to be one of the basic human qualities. An image is a description of an object. Thus, from many characteristics of an object, a person can determine the required image [4]. This ability is a very complex information system. The development of artificial pattern recognition systems remains a complex theoretical and technical problem. The need for such recognition arises in various fields: from military and security systems to the digitisation of various analogue signals. The production of devices that perform the functions of recognising various objects, in most cases, makes it possible to replace a human with a specialised machine. This greatly enhances the capabilities of complex systems performing various information, logical and analytical tasks. It should be noted that the quality of work performed by a person at the workplace depends on many factors (qualifications, experience, conscientiousness, etc.). At the same time, a well-

functioning machine operates in a uniform manner and always delivers the same quality. Automatic control of complex systems allows for monitoring and timely maintenance, identification of interference and automatic application of appropriate noise reduction methods, and improves the quality of information transmission. It is also clear that the use of automated systems in a number of tasks can provide speeds that are impossible for humans.

When protecting large geographically distributed objects, crowded places, as well as when solving the tasks of preventing terrorist threats, searching for criminals and crane, it is extremely important to use the intellectual capabilities of modern video surveillance software, including decision support functions; operational analysis of the situation; face recognition in a stream of people; recognition of registration numbers, models and colour of vehicles; motion detection; detection of left objects;

All intelligent video surveillance systems use video analytics, a technology that uses computer vision techniques to automatically obtain various data based on the analysis of a sequence of images coming from real-time video cameras or from archived records.

Video analytics is software for working with video content [5]. The software is based on a set of machine vision algorithms that allow video monitoring and data analysis without direct human intervention. Video analytics algorithms can be integrated into various business systems, as they are most commonly used in video surveillance and other security areas. Video analytics automates four functions of security equipment:

1) detection;

2) surveillance;

3) recognition;

4) forecasting.

All four functions are performed repeatedly, providing continuous refinement of hypotheses about the number, location and types of objects in the controlled area, as well as eliminating redundancy in the results. Perimeter video analytics performs all four functions: detection, tracking (to prevent repeated triggering of the same object), recognition (to minimise false alarms caused by animals and other "noise" in the environment), and prediction (to track an object when it temporarily disappears from the field). Recognition

can be understood as a wide range of tasks: from classifying an object into target/noise to identifying or verifying an object by biometric features.

The main consumers of these systems are banks, cultural institutions, municipal facilities, warehouses, strategic facilities, industrial and other facilities operating with access control systems, organisations using security television systems, and enterprises with a high level of security.

1.5. Analysis of object recognition systems and their application to night vision cameras

Object recognition systems for night vision cameras are a very important element in modern security systems, navigation, transport monitoring and other areas. Here we will look at some aspects of these systems and their applications:

Infrared (IR) night vision: Cameras equipped with IR illumination can be used for night vision. They use infrared light to illuminate objects in the dark, allowing the cameras to remain functional in low-light conditions.

Image processing algorithms: To detect objects at night, cameras often use image processing algorithms to help improve contrast and resolution in low-light conditions.

Thermal imaging technology: Some cameras use thermal imaging for night vision. Thermal cameras measure the thermal radiation of objects and create an image based on this data, allowing you to recognise objects even in complete darkness.

Object tracking algorithms: Some systems have built-in algorithms for tracking objects at night. This can be particularly useful for security and monitoring of moving objects during the night, such as vehicles or persons.

Face recognition: Some systems have the ability to recognise faces in the dark. This can be useful for identifying individuals or detecting dangerous situations.

Security and monitoring applications: Night vision cameras with object recognition systems are used in security applications for the protection of facilities, military operations, transport monitoring and other areas where it is necessary to monitor events at night.

Given the wide range of applications, object recognition systems for night vision cameras are becoming an integral part of many modern security and monitoring systems.

1.6. Information support for object recognition from a night vision camera

The study of information support for object recognition from night vision cameras covers various aspects, including the development of image processing algorithms, analysis and interpretation of the data obtained.

Image processing algorithms The development of image processing algorithms specifically designed to work with images obtained from night vision cameras is an important area of research. These algorithms can include noise filtering, contrast enhancement, distortion correction, and other image optimisation techniques. Image feature analysis: The study of features and characteristics of images that can be useful for recognising objects at night. This may include analysing texture, shape, size, brightness, and other image parameters.

Machine Learning and Neural Networks Use machine learning techniques, such as neural networks, to perform automated object recognition at night. Training models on large data sets helps to improve the accuracy and efficiency of recognition.

Tracking and identification functions Research into methods for tracking objects in images, as well as methods for identifying specific objects or individuals. This may include developing algorithms for tracking moving objects, face recognition, and other identification methods.

Development of specialised datasets Creation of specialised datasets for training and testing of night-time object recognition models. This may include collecting images from various night vision camera sources and annotating them for further use in research.

Performance evaluation Define metrics to evaluate the performance of object recognition in night vision images. This allows you to compare different approaches and determine the best methods for specific tasks.

Aspects of the research help to improve the quality and efficiency of object recognition systems in night vision camera images, which is important for many applications in security, monitoring and other industries.

1.7. Type of threats and means of software protection of intelligent systems

Night vision cameras can be exposed to various types of threats, which include physical and digital attacks. Types of attacks:

1. Physical access to the camera: Criminals may attempt to gain physical access to the camera to perform various actions, such as disabling the camera, physically damaging it, or even stealing it. Protective measures can include installing cameras in remote or inaccessible locations, using mechanical protections (e.g., sealed housings or protective covers), and installing physical intrusion alarm systems.

2. Intercepting and modifying the data stream: Attackers may attempt to intercept and modify the data stream transmitted from the camera. To protect against this, you can use traffic encryption (e.g. TLS/SSL), authentication and authorisation, secure network protocols, and firewalls.

3. Denial of service (DoS) or destruction of equipment: DoS attacks can be aimed at overloading the camera with requests or denial of service. Defences can include the use of protective devices that can detect and block excessive requests, as well as backup and recovery mechanisms for quick recovery in the event of problems.

4. Exploitation of software vulnerabilities: Cameras can be attacked by exploiting vulnerabilities in the firmware. Protections include keeping the software up to date and installing protective mechanisms such as firewalls, antiviruses, and intrusion detection systems.

5. Unauthorised access to the camera or recording: Insufficient authentication or weak passwords can result in unauthorised access to the camera or recording, which can lead to a breach of privacy or data confidentiality. Protections include setting strong passwords, using two-factor authentication, and restricting access to the camera.

6. Signal interference: Attackers may attempt to interfere with the camera's operation by using radio signals or other interference methods. Protections include the use of secure communications and RF jamming protection.

Traditionally, the Internet is considered to be the most vulnerable route, but the threat from other directions is no less significant. It is important that all network entrances and exits are securely protected [7]. Continuous monitoring of the operation of intelligent local

network systems, which are the backbone of any network, to keep it in a constant working order.

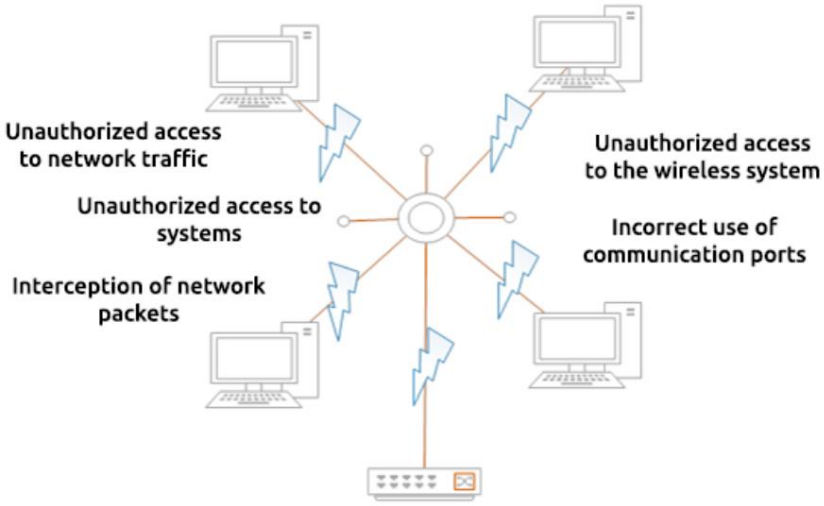

Fig. 1.1. Local network threats during normal computer operation

These are just a few of the threats and software protections for night vision cameras. Ensuring system security involves a comprehensive approach that covers both physical and digital aspects of protection. "Information protection" is a special combination of both hardware and software tools for measures to administer the system with respect to its integrity, confidentiality and availability, in the face of unauthorised access. Control is the main stage, which is primarily responsible for managing the network. Given the importance of this function, it is often separated from other functions of control systems and implemented by specialised tools. The use of autonomous control tools provides more opportunities to identify problematic parts of the software package and set up document, flow in the institution's local network[26].

In particular, it is necessary to ensure reliable authentication of users in the directory service (single sign-on centre), although authentication at the level of the server and network workstations significantly reduces the quality of security. Modern requirements include management of the communication network environment and division into specific logical segments (VLANs). To administer remote devices, you should always use a secure protocol connection (for example, SSH). The following tools and methods are used to ensure information security and reduce the possibility of information leakage:

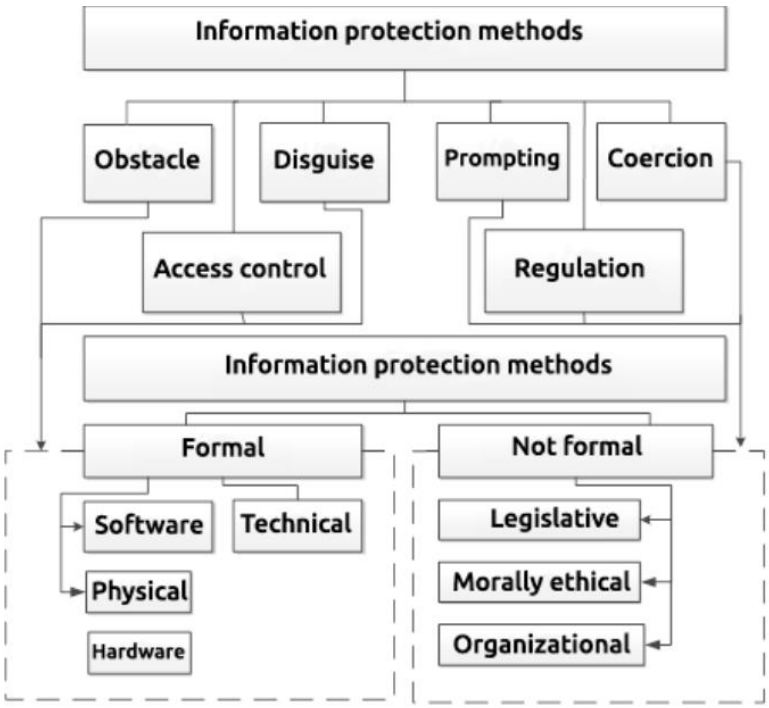

Fig. 1.2. Information security in intelligent systems

The following protections can be distinguished:

1) Data archiving is a software tool that merges several files into a single file, but without losing data, and with the ability to achieve accurate recovery of the original files.

2) Antiviruses are software products designed to actually protect data from viruses, including PUPs.

3) Encryption software is software that protects electronic documents and other files from unauthorised access using encryption and authentication algorithms.

Authentication software - software to verify that a person's access is appropriate with the login IDs provided by him/her and to confirm his/her authenticity, as well as to determine his/her authority in the system and to control the rights assigned to him/her in the process of working in the system.

Specialised software tools for protecting information from unauthorised access generally have better capabilities and characteristics than the built-in tools of network operating systems [8].

1.8. Overview and comparison of existing algorithms and methods for object recognition in intelligent night vision systems

Comparing these methods can be challenging, as it is important to consider the computational costs and resource requirements of each method [9]. Their performance at night depends on specific shooting conditions such as lighting, rain and noise.

Image reconstruction techniques use the intensity and structure of images from night vision cameras to detect objects. They can include image processing algorithms such as filters, adaptive thresholding, and morphological operations [10].

Thermal imaging can be used to detect objects that have a different temperature from the surrounding environment. Such detection methods can be particularly effective in low light conditions.

Night vision cameras often use infrared illumination to illuminate objects in the dark. Image processing algorithms can then use this data to recognise objects in images.

Deep neural networks, in particular convolutional neural networks, show great potential in recognising objects in images, including those captured by night vision cameras. They can automatically identify important features of objects and classify them.

Object tracking algorithms (e.g., Kalman algorithms): These algorithms track the movement of objects in consecutive frames of video, which can be particularly useful for recognising objects in time and space in night camera images.

1.9. Recognising objects using neural networks

Artificially intelligent neural networks are a mathematical model that works through the functioning of biological neural networks, which are networks of nerve cells in a living organism. Just like in a biological neural network, the main component of an artificial neural network is a neuron. Combined neurons form, so to speak, layers, and their number may vary depending on the problem of the neural network and the tasks it solves. The abstract foundations of programming such neural networks are described in many works. One of the modern tasks is to identify visual images. An intelligent system that can recognise signs on licence plates, human faces, and even seals and manuscripts greatly facilitates human work and makes the process of work much easier [11]. As a result, the risk of miscalculation is

reduced due to the absence of the human factor. Neural network methods are methods based on the use of various types of neural networks. Neural networks have been used for a long time and very effectively to solve recognition problems. The main idea behind neural networks is a signal transformation sequence that interacts with parallel working components, and despite significant differences, certain types of neural networks have some common features. Each neural network is based on somewhat simple and mostly uniform elements, so to speak, cells that copy the action of brain neurons.

Neurons are characterised by their current state by analogy with nerve cells in the brain, which can be excited or inhibited. It has a group of synapses - unidirectional input connections connected to the outputs of other neurons, and also has an axon-output connection of this neuron, which sends a signal (excitation or inhibition) to the synapse of the following neurons. Each synapse has a characteristic such as the value of the synaptic connection or its weight wi. The state of a neuron is calculated by the formula of the weighted sum of its inputs:

$$S = \sum x_i \cdot w_i \tag{1.1}$$

Another feature inherent in neural networks is the principle of parallel signal processing, which is achieved by associating a large number of neurons in a layer and connecting neurons of different levels in a specific way, and in some configurations of neurons in a common layer between them, and, in this case, processing them together.

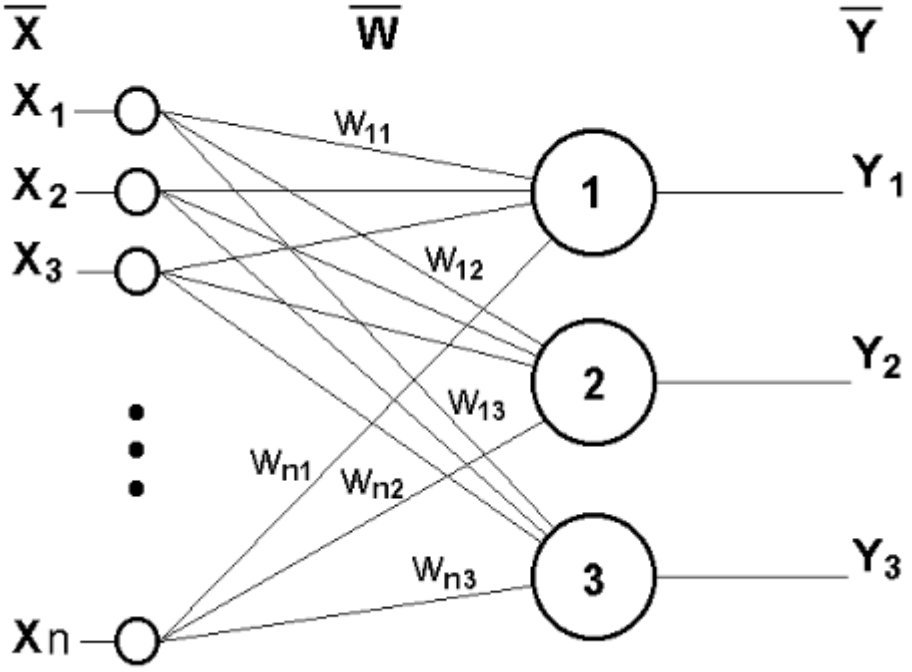

Fig. 1.3. Structure of connections of a single-layer neural network

There are approximately a dozen key types of neural networks, as well as their underlying performance structures created on the basis of auto-associative memory (AM). This is formed by reactions in response to some input statistical data indicator, they can be called a "key", and also provide the conclusion stored in the network close to the input statistical expression of the same size. When such a moment as image recognition occurs, the normalised image is the clue. The process of neural networks based on auto-associative memory is represented as a vector. All coordinates of the vector are placed in its own cell, which are connected to other cells [13]. These vectors are then processed by a neural network, and its output is a photo close to the input stored in the AP.

Despite the advantages of neural networks, it is a fact that special efforts are required when using them in terms of images. This is primarily due to the complex nature of images, in particular, three-dimensional real-world objects, such as human faces.

1.10. Analysis of existing software

After reviewing several analytical materials related to the global market of networked video surveillance and security equipment, the following was found out:

1. OpenCV is one of the most popular libraries for image processing and computer vision, with a wide range of features for object recognition, including object tracking, object classification, and object detection.

2. Using the TensorFlow and Keras machine learning libraries, you can build and train various deep learning models for object recognition in night vision camera images.

3. YOLO is a popular object detection algorithm that allows real-time detection and classification of objects in images, for example, in automotive security or monitoring systems.

4. Darknet is a neural network designed for object detection, classification, and tracking. It is open source software that allows you to use pre-trained models or train your own to recognise objects in night vision images.

5. MATLAB Computer Vision Toolbox offers a set of computer vision tools that includes functions for image processing, recognition

AXIS Perimeter Defender is one of the best-known systems for recognising objects from night vision cameras. This is a solution from Axis Communications, a company specialising in the production of network equipment for video surveillance and security. Another leading manufacturer of object recognition systems is Bosch Intelligent Video Analytics (IVA), which uses machine learning algorithms to detect intrusions and recognise objects from night vision cameras. There are several other similar night vision object recognition systems that also provide motion tracking, intrusion detection, and perimeter security functionality: Hikvision AcuSense, Dahua Smart Motion Detection (SMD).

1.11. Privacy policy of intelligent systems for private enterprises and government agencies

An important concept in the design and analysis of secure systems is the security model, which implements the security policy adopted in the information system [14]. A security model translates abstract policy goals into information system terms, accurately

describing the data structures, tools, and methods required to implement the security policy. You should also pay attention to the reporting system and sets of preconfigured security policies provided by the solution, as this will help to avoid some problems and difficulties during implementation. A privacy policy is an important part of an enterprise that describes the rules for collecting, storing, and sharing CI [15]. It is often described in the document "Information Policy Statement". A security policy is a set of documented decisions made by the organisation's management aimed at protecting information. Security policy involves outlining goals without specifying how they should be achieved.

1.12. Problem statement

To investigate this issue, it is necessary to analyse the flows of objects that will circulate within it. The object classification includes various objects and backgrounds in low-light conditions at night. The result of this work should be a study of the optimal intelligent object recognition systems for night vision cameras based on neural networks capable of identifying objects. Also, to investigate the most effective means for improving such systems in extremely difficult conditions. To accomplish this task, we need to:

1. Develop a model for recognising objects in night vision camera images.
2. Train the model to recognise different classes of objects in images.
3. Evaluate the effectiveness of the developed model on a test dataset, including classification accuracy, recognition time, and robustness to changes in input conditions.
4. Select and apply image pre-transformation methods to improve model performance.
5. Analyse the results and draw conclusions about the suitability of the developed system for use in real life.

It is also desirable to minimise the costs of implementing this project as much as possible, but at the same time try not to neglect the quality of the materials and equipment used.

SECTION 2

T

2.1. Preliminary image transformation and analysis of recognition system performance

Image pre-transformation will significantly improve the performance of an object recognition system, especially in low-light conditions at night. Here are some possible pre-transformation methods and their impact on system performance. Image normalisation can help reduce the impact of variations in lighting and colour on model performance. This can include various techniques such as scaling pixel values to the range [0, 1] or shifting the mean and standard deviation. Scaling pixel values: This can be expressed by the following formula:

$$pixel_{scaled} = \frac{pixel_{original}}{maximum\ pixel\ value} \quad (2.1)$$

Shift of the mean and standard deviation: This can be expressed by formulas:

$$pixel_{normalized} = \frac{pixel_{original}}{maximum\ pixel\ value}$$

where μ is the mean pixel value and σ is the standard deviation.

The maximum pixel value is typically 255 for 8-bit images (typically used for RGB channels).

Adding artificial variations to the training dataset can improve the model's performance and make it more robust to different shooting conditions. Augmentation can include rotation, displacement, scaling, reflection, etc. It can help to increase the size of the training dataset, improve the model's performance, and make it more robust to different conditions. Formula for rotating a point (x,y):

$$x' = x \cdot \cos(\theta) - y \cdot \sin(\theta)$$
$$y' = x \cdot \sin(\theta) + y \cdot \cos(\theta) \quad (2.2)$$

Mathematical description: the image is rotated by an angle θ using a rotation matrix (parameter: rotation angle θ). The scale is changed through the scaling parameter α. Formula for scaling a point (x,y):

$$x' = \alpha \cdot x$$
$$y' = \alpha \cdot y \quad (2.3)$$

Mathematical description: the image is enlarged or reduced by a factor of α using interpolation. Depending on the type of display (horizontal or vertical), the image is displayed relative to the vertical or horizontal axis. Formulas for horizontal (2.1.4) and vertical (2.2.5) mapping:

$$x' = -x$$
$$y' = y \quad (2.4)$$

$$x' = x$$
$$y' = -y \quad (2.5)$$

The noise reduction is done through a Gaussian filter. This can be expressed as the convolution of an image I with a Gaussian kernel G:

$$I_{\text{filtered}} = I * G \quad (2.6)$$

Median filter: Replaces the pixel value with the median value in the pixel's vicinity.

These transformations can be combined together to create a variety of new images in the training dataset. This helps the model learn to recognise objects in different conditions, increasing its robustness and overall accuracy. Contrast enhancement is achieved by changing the brightness of each pixel based on a certain ratio.

2.2. General principles of building a model for recognising people and objects

The general principles will help you build an effective model for object recognition in night vision images. It is important to experiment and pay attention to the results to achieve the best possible outcome. The general principles of building a model for object recognition in

night vision camera images are based on the principles of deep learning and image processing. Here are some general steps and principles that can be used:

1. First, you need to collect a dataset that contains images of people and objects that you want to recognise. After that, the data is subject to preprocessing such as resizing, normalisation, augmentation, etc. Data preprocessing includes noise reduction, contrast enhancement, image resizing, etc.

2. Choosing an architecture based on convolutional neural networks (CNNs), such as ResNet, MobileNet, VGG or YOLO.

3. Data pre-processing includes noise reduction, contrast enhancement, image resizing, and more.

4. Model training after data preparation and model architecture selection. During model training, the model learns internal parameters to optimally perform the object recognition task (ResNet trained on ImageNet as a base model and then retrained on its own dataset).

5. After training, evaluate the performance of the model on a test dataset (using a metric such as accuracy, sensitivity, specificity, F1-index). Modifications should be made to the model to improve its performance, such as changing hyperparameters (with model hyperparameters such as learning rate, batch size, number of layers, etc. to achieve better results).

6. Inference. After successfully training and tuning the model, you can use it to recognise objects in new images (the model can determine which objects are present in the input images and their locations).

7. Optimisation and improvement of results. The process of model training and evaluation can be iterative. You need to experiment with different model architectures, data processing, and hyperparameters to improve the results. For example, retrain and regularise the model (add regularisation techniques such as dropout or L2 regularisation to prevent the model from overfitting).

2.3. General principles of connection

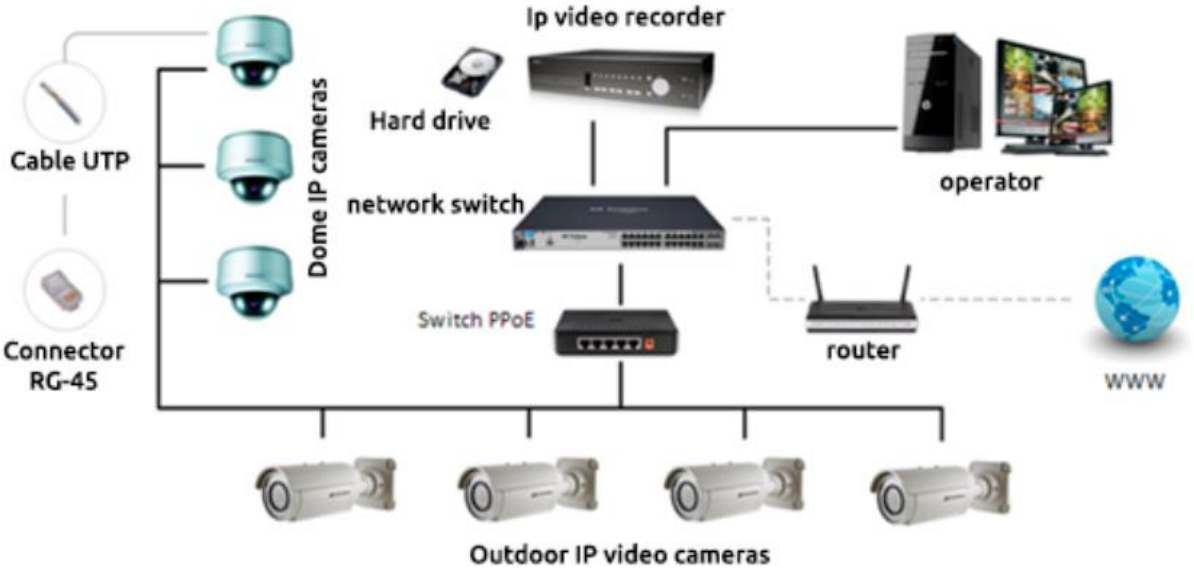

Fig. 2.1. Typical scheme of IP video surveillance

It is even easier to assemble and connect an IP video surveillance system than a classic one, because such systems can use an existing computer network to transmit video signals. IP video surveillance systems have higher image quality and wider functionality, but are slightly more expensive and more difficult to set up than classic systems. The structure of an IP video surveillance system consists of the following elements:

1. Digital video recorder. Unlike a classic DVR, an IP recorder does not have a large number of video inputs, because the wires from all cameras are connected to a switch, which in turn is connected to the recorder with just one wire. However, you should pay attention to the number of channels it can record and take into account the future development of the video surveillance system.

2. Hard drive. The archive depth depends on the hard drive capacity. You can calculate the approximate archive depth based on the HDD capacity in the description of the hard drive.

3. IP video surveillance cameras. When choosing cameras, you need to pay attention to:

- installation location (indoors or outdoors) and vandalism protection;

- temperature conditions for outdoor cameras;

- resolution (the higher the resolution of the video camera, the better the picture, but the more space you need to store the archive);

- the viewing angle (the higher it is, the more space the frame will contain);

- For guaranteed compatibility, it is recommended to purchase cameras and a recorder from the same manufacturer.

4. Switch. The signal from all video cameras is routed to the switch, so it is important to choose a switch with the right number of ports.

5. Power supply. When using cameras without PoE support, a simple or uninterruptible power supply is required.

7. Twisted pair cable or UTP patch cord. The video signal is transmitted via a conventional twisted-pair computer cable.

9. Monitor. You can use a computer monitor or a home TV with HDMI or VGA inputs to view the image

10. Microphone.

11. UTP Patch Cord. To connect the switch to a DVR and a home router (to view the image via the Internet), you will need a UTP Patch Cord.

Highly intelligent surveillance systems, when installed with the appropriate software, are able to independently combat unauthorised intrusion into a home. Integration with other smart home control systems allows video security to, for example, block the lighting in the house or activate an audible alarm.

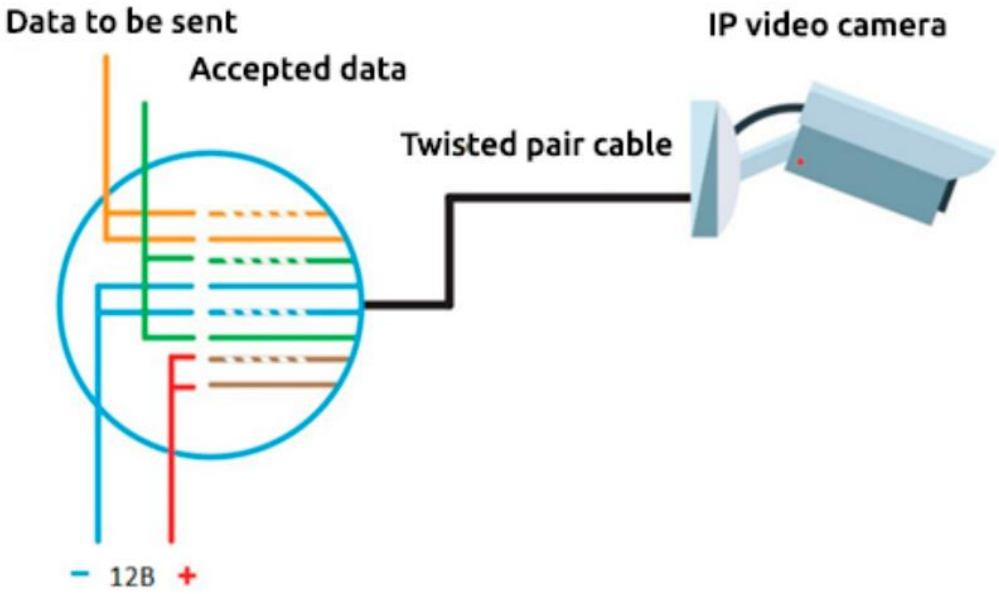

Fig. 2.2. Connection diagram of the IP video camera

2.4. General principles of system operation

In night-time surveillance systems, face recognition technology works using specifically infrared (IR) or thermal images, as these types of images allow objects to be distinguished by their thermal emission, which is especially useful in low-light conditions. The developed algorithm is capable of self-learning - Deep Learning, which can produce ideal results compared to obsolete technologies. By modelling the neural network, this technology allows us to simulate the functioning of the human brain. Thus, the equipment does not perform its functions, but learns in each situation, which eliminates similar situations. Deep convolutional neural networks can be used to detect faces in images with high accuracy. Architectures such as MTCNN (Multi-Task Cascaded Convolutional Networks), SSD (Single Shot Multibox Detector), and YOLO (You Only Look Once) are often used for this purpose. The Histogram of Oriented Gradients (HOG) algorithm uses gradient vectors to describe the shape and appearance of a face, and then uses a classifier (e.g. SVM) to determine whether a face is present in the image. The self-learning process will be based on primary features to identify an individual; from superficial to deep analysis of everything that happens on the object. These two features help the video surveillance system to recognise faces in the crowd

as quickly as possible by comparing the data in the database, providing information about the person, even when the appearance has changed (a beard, moustache, aging, a different headdress, etc.). In general, facial recognition technology should provide the following capabilities: maintain a database of white and blacklisted visitors, increasing security; identify facial features by determining a person's age and gender; compare a new photo with the original information about the person; integrate with a remote access control system; and recognise faces even in low light.

2.5. Overview and comparison of existing algorithms and frameworks

Each of the architectures has its advantages and limitations, and the best choice may depend on the specific application scenario according to the technical design.

In the course of the research and experiments, a brief overview of each of them and their characteristics was made:

- MTCNN (Multi-Task Cascaded Convolutional Networks)

MTCNN is a cascade neural network designed to detect faces and landmarks in images. Step-by-step description: the image is normalised and scaled to increase processing efficiency, the image is divided into sub-regions, P-Net analyses each sub-region to detect possible faces. The selected regions are passed to R-Net for further processing. R-Net refines the boundaries of the faces eliminated by P-Net. The number of incorrectly identified faces is then reduced. The refined regions are then passed to O-Net.

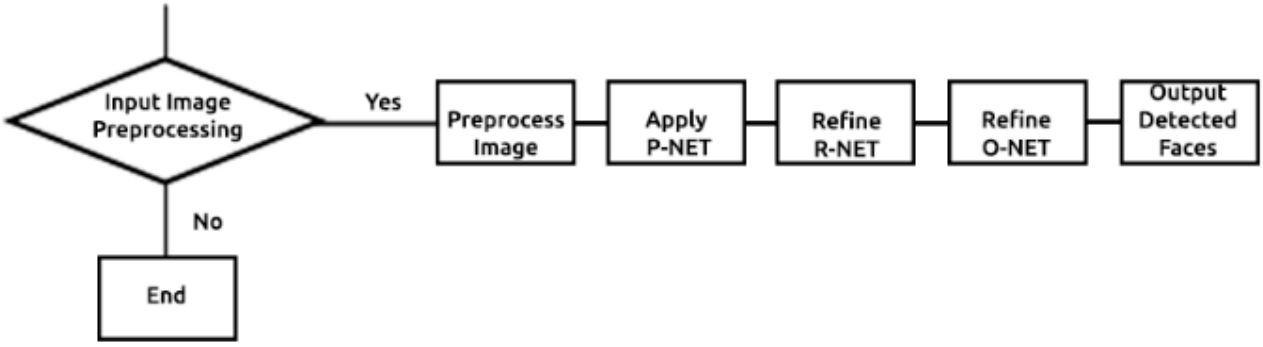

Fig. 2.3. Block diagram of MTCNN operation

– SSD (Single Shot Multibox Detector)

SSD (Single Shot Multibox Detector) is an algorithm for object detection in images. It offers simultaneous detection and classification of objects in an image, speeding up the processing process. SSD uses deep neural networks, including convolutional and full-connection layers, to extract features from an image and determine the position and class of objects. The algorithm is capable of working with different image sizes and detecting objects of different sizes and classes in a single shot, which makes it faster than some other methods. A step-by-step description of the SSD algorithm: The algorithm starts with an input image that contains the objects to be detected. Features are extracted from the input image using convolutional layers. The extracted features are passed through a series of convolutional layers to detect objects of different sizes and shapes. The algorithm detects objects in the image based on the extracted features and information from the convolutional layers. To eliminate duplicate detections and improve accuracy, non-maximal suppression is applied to the detected objects. The last step is to output the detected objects along with their bounding boxes.

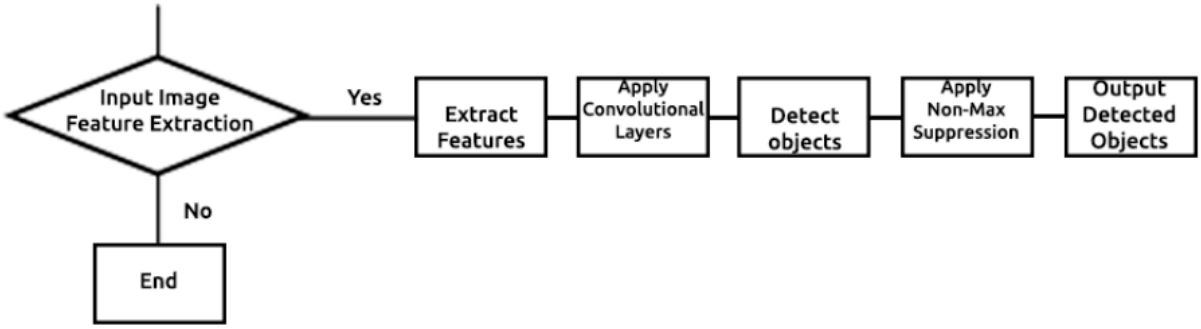

Fig. 2.4. Block diagram of SSD operation

- YOLO (You Only Look Once)

YOLO (You Only Look Once) is an object detection algorithm that offers fast and efficient object detection in an image or video. The main idea of YOLO is to perform object detection in a single pass of the image through a deep neural network. A step-by-step description of how the YOLO algorithm works: the algorithm starts with an input image that contains the objects to be detected. Features are extracted from the input image using convolutional layers. The extracted features are passed through a series of convolutional layers

to detect objects and classify them into different classes. YOLO provides bounding boxes and class probabilities for the detected objects based on the extracted features. To eliminate duplicate detections and improve accuracy, non-maximal suppression is applied to the predicted bounding boxes. The last step is to output the detected objects along with their bounding boxes and class probabilities.

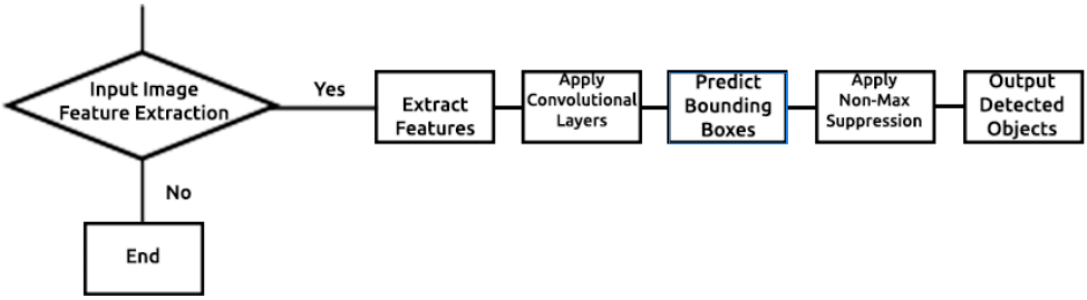

Fig. 2.5. Block diagram of YOLO operation

- AOP SVM (Support Vector Machine)

Support Vector Machine (SVM) is a machine learning algorithm for solving classification and regression problems. The basic idea is to find the optimal hyperplane that best separates data from different classes in the feature space. The SVM tries to maximise the distance between this hyperplane and the closest points of each class, called support vectors. A step-by-step description of how the SVM algorithm works: the algorithm starts with input training data consisting of labelled examples. Features are extracted from the input data, converting it into a format suitable for SVM training. The SVM model is trained using the labelled examples to find the optimal hyperplane that best separates the classes in the feature space. The final step outputs the trained SVM model, which can be used to classify new, unseen data.

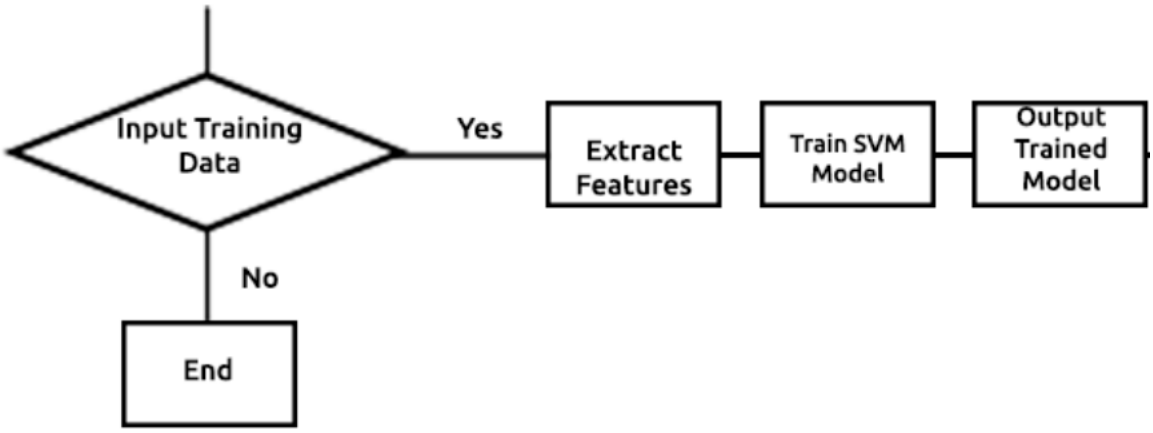

Fig. 2.6. Block diagram of SVM operation

- PCA (Principal Component Analysis)

PCA (Principal Component Analysis) is a data dimensionality reduction method used to remove unnecessary information and select the most important features. The main idea of PCA is to find new orthogonal features (principal components) that preserve the largest part of the variation in the data. This is achieved by transforming the original features into a new feature space where the first principal components have the highest variance. A step-by-step description of the PCA algorithm: the algorithm starts with input data containing features. If necessary, the data is standardised to have a mean of 0 and a standard deviation of 1. The covariance matrix of the standardised data is calculated. Eigenvectors and eigenvalues of the covariance matrix are calculated. The principal components are selected based on the highest eigenvalues representing the directions of maximum variance in the data. The original data is transformed into a new feature space defined by the selected principal components. The last step outputs the transformed data with reduced dimensionality.

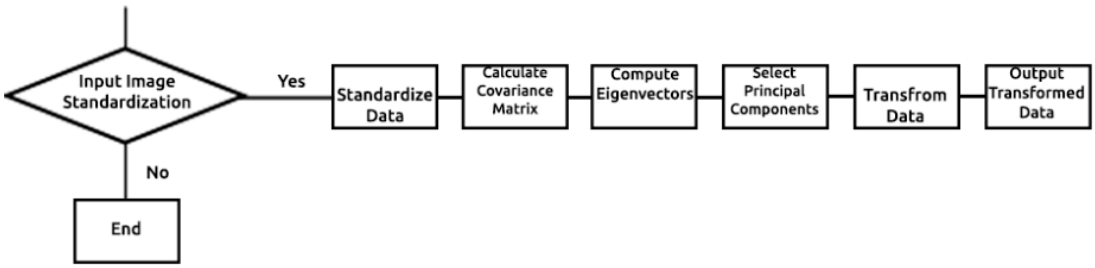

Fig. 2.7. Block diagram of PCA operation

- Local Binary Patterns (LBP)

Local Binary Patterns (LBP) is a method for describing the texture characteristics of images. The basic idea behind LBP is to describe each pixel in an image by comparing its value with its neighbours and encoding this information as a binary pattern. This pattern can then be used to describe the texture characteristics of a particular area of the image. LBP is widely used in the fields of computer vision, image response, and video analysis. A step-by-step description of how the LBP algorithm works: The algorithm starts with an input image. If necessary, the input image is converted to grayscale for easier processing. The LBP operator is applied to each pixel in the grey image to calculate a local binary pattern. A histogram of the calculated local binary patterns is generated to represent the texture features of the image. The last step outputs the LBP histogram, which can be used for further analysis or classification.

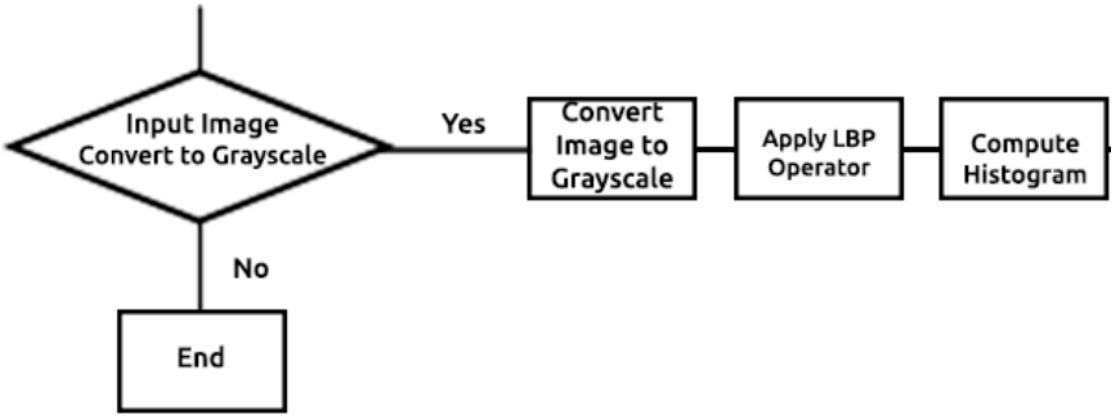

Fig. 2.8. Block diagram of LBP operation

- DeepFace

DeepFace is a deep neural network specifically designed for face recognition. It is used to automatically detect and analyse faces in images and videos. A step-by-step description of how the DeepFace algorithm works: the algorithm starts with an input image containing a face. Faces are detected in the input image using a face recognition algorithm. The detected faces are aligned according to the canonical pose to improve the recognition accuracy. Local binary patterns are extracted from the aligned faces to capture texture features. The

extracted features are used to classify faces into known faces or classes. In the last step, the identified faces are output along with their classification.

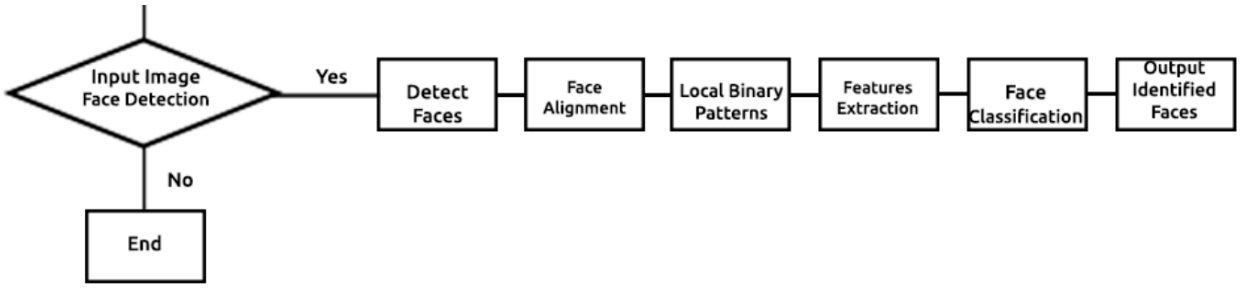

Fig. 2.9. Block diagram of DeepFace operation

- DLIB

DLIB is an open-source software development library that contains tools for computer vision, machine learning, and image processing tasks. A step-by-step description of how the DLIB face recognition algorithm works: the algorithm starts with an input image. If necessary, the input image undergoes pre-processing steps such as resizing, normalisation, or grey-scale conversion. DLIB detects faces in the preprocessed image using its face detection algorithm. In the last step, the detected faces are output along with their bounding boxes.

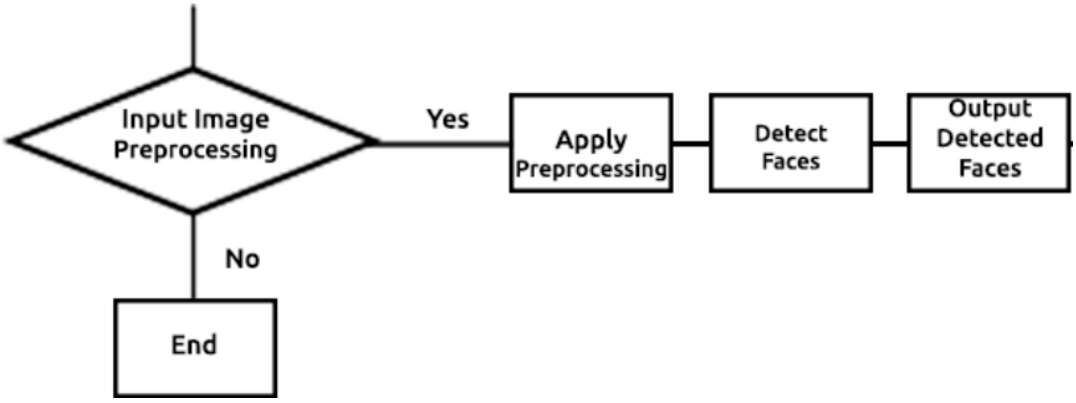

Fig. 2.10. Block diagram of DLIB operation

- OpenFace

OpenFace is an open source software that provides functionality for face processing and emotion analysis. OpenFace can be used for a variety of tasks, including recognising faces in photos and videos, tracking emotional reactions on faces, and studying

behavioural aspects related to faces. A step-by-step description of how the OpenFace face recognition algorithm works: the algorithm starts with an input image. If necessary, the input image undergoes pre-processing steps such as resizing, normalisation, or grey-scale conversion. OpenFace detects faces in the pre-processed image using its face recognition algorithm. The last step is to output the detected faces along with their bounding boxes.

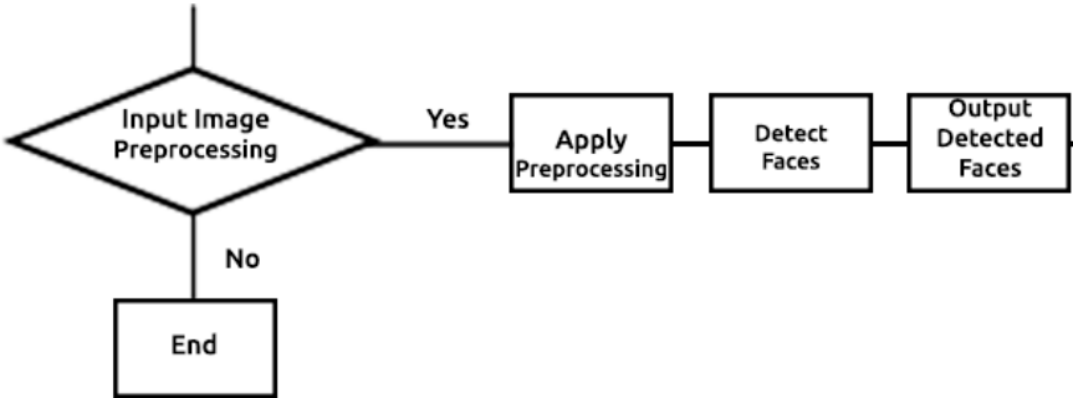

Fig. 2.11. Flowchart of OpenFace operation

Table 2.1

Algorithm	Advantages	Disadvantages	Applying
MTCNN	High precision, multi-stage processing	Demanding on resources	After the flight
SSD	Fast, efficient	Less accurate for small objects	Real time
YOLO	Very fast, real time	Less accurate for distant objects	Real time
SVM	High accuracy, efficient with small data	Requires correct configuration	After the flight
PCA	Reduces dimensionality, eliminates correlation	Loss of interpretability	Preliminary processing

Continuation of table 2.1

LBP	Effective for textures	Sensitive to changes in lighting conditions	After the flight
DeepFace	High precision	Data-demanding	After the flight
OpenFace	Open source software, wide functionality	Less accurate	After the flight
DLIB	Real time, efficient	Less accurate for complex tasks	Real time

For real-time tasks on a drone or aircraft, SSD, YOLO, and DLIB algorithms are suitable due to their speed and efficiency. For post-flight data processing, especially when high accuracy is required, it is better to use MTCNN, SVM, DeepFace, or OpenFace. Each algorithm has its own strengths and weaknesses, so the choice depends on the specific requirements and conditions of the task.

2.6. Deep learning model

Deep neural networks can be effectively used to recognise objects at night. This is especially relevant in the fields of video surveillance, self-driving cars, and security. Here are some of the methods and techniques that can be used:

```
import torch
import torchvision.transforms as transforms
from torchvision.models.detection import fasterrcnn_resnet50_fpn
from torchvision.models import resnet50
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
```

1. In this paper, we will use convolutional neural networks (CNNs) in the ResNet architecture, which has high performance and can be configured to work in low light conditions.

2. Nighttime video surveillance data. Large amounts of data are required for training the DPS, including images captured by night-time video cameras. This data should cover different lighting conditions, weather conditions and object types.

3. Data pre-processing: Due to low light levels at night, images may be blurred or contain noise. Data preprocessing, such as contrast enhancement, noise reduction, or sharpening, can improve the performance of the DSS.

4. Data augmentation: Data augmentation can be used to increase data diversity and improve the generalisability of models. This includes applying random transformations to images, such as rotation, scaling, and brightness changes.

5. Transfer of training: It is possible to use pre-trained DSS models on large datasets (e.g. ImageNet) and retrain them on nighttime data for object recognition. This method, called transfer learning, can significantly reduce training time and improve model performance.

6. Specialised architectures: Some DSS architectures are specifically designed to operate in low-light environments. For example, networks with branches to adapt to different brightness levels or networks with adaptive depth variation to use resources efficiently.

Convolutional Neural Networks (CNNs), such as ResNet, are powerful and effective tools for many machine learning and computer vision tasks. Convolutional neural networks can automatically learn meaningful features of images during training. They use convolution and pooling to locally perceive the image and reduce the dimensionality, which allows the models to efficiently detect visual patterns. Modern CNN architectures, such as ResNet, can be very deep, allowing them to represent more complex visual concepts. This is especially important for tasks that require a high level of abstraction, such as face, object, or scene recognition

With the help of pre-trained models trained on large datasets (e.g. ImageNet), you can use the knowledge gained during training to solve new tasks with relatively little training data. Convolutional operations are performed using filters (kernels) that move through the input

image. During this operation, the scalar product between the pixel values and the corresponding filter values is performed and then summed. This can be mathematically written as:

$$(I * K)(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x + i, y + j) \cdot K(i, j) \quad (2.7)$$

where *I is the* input image, *K is the* filter kernel, aa and *b are the* kernel dimensions.

Pulling is used to reduce the dimensionality of an image and highlight important features. Typically, the maximum or average value in a particular region of the image is used. For example, maximum pooling can be mathematically written as:

$$M(x, y) = \max_{i,j \in R(x,y)} I(i, j) \quad (2.8)$$

where *M is the* original image after pooling, *R*(x,y) is a region in the original image.

The activation function is introduced for non-linearity and the ability of the model to learn more complex dependencies in the data. One of the most popular activation functions is ReLU (Rectified Linear Unit), which is mathematically defined as:

$$f(x) = \max(0, x) \quad (2.9)$$

ResNet uses the concept of "residual connections" to help avoid the problem of gradient loss when training deep networks. The basic idea is that instead of training the network to learn new functions, it learns to keep the original values (identity mapping) if it is appropriate. This can be expressed mathematically as:

$$y = F(x, \{W_i\}) + x \quad (2.10)$$

where *x is the* input, *F(x,{Wi})* is the convolution block and Wi are the parameters of this block.

These are just a few of the basic mathematical concepts and calculations that were discovered when working with CNN and the ResNet architecture

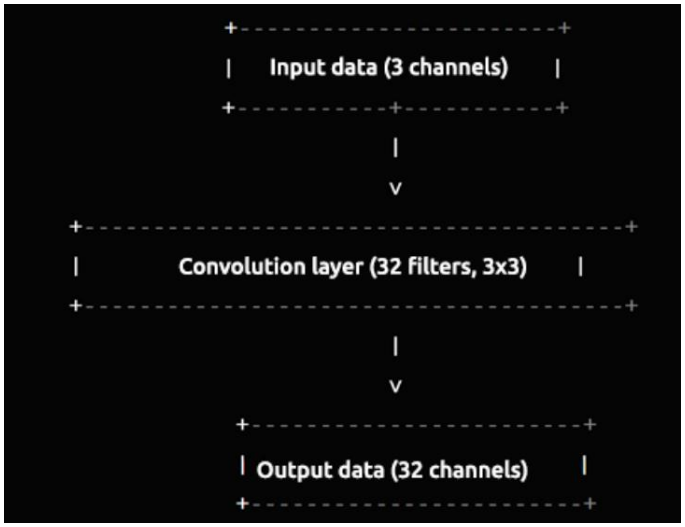

Fig. 2.12. Diagram of a convolutional layer with three input channels (RGB), 32 filters, and 3x3 filter size.

You can visualise the parameters and results of convolutional and pooling layers, as well as the results of forward and backward transfer, using various data visualisation tools. For example, you can use TensorBoard with TensorFlow or PyTorch Lightning to display graphs and metrics while training your model. You can also use the Matplotlib library to display images or graphs in the post-processing phase.

```
# Pass an image through a convolutional layer
conv_output = conv_layer(image)

# Creating a pooling layer
pool_layer = nn.MaxPool2d(2, stride=2)

# Passing the outlet of the rolling layer through the pulping layer
pooled_output = pool_layer(conv_output)

# Visualisation of the source image
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image[0].permute(1, 2, 0))

# Visualisation of the original image after the convolution layer
```

```
plt.subplot(1, 2, 2)
plt.title('Output after Convolution')
plt.imshow(conv_output[0][0].detach().numpy(), cmap='gray')
plt.show()
```

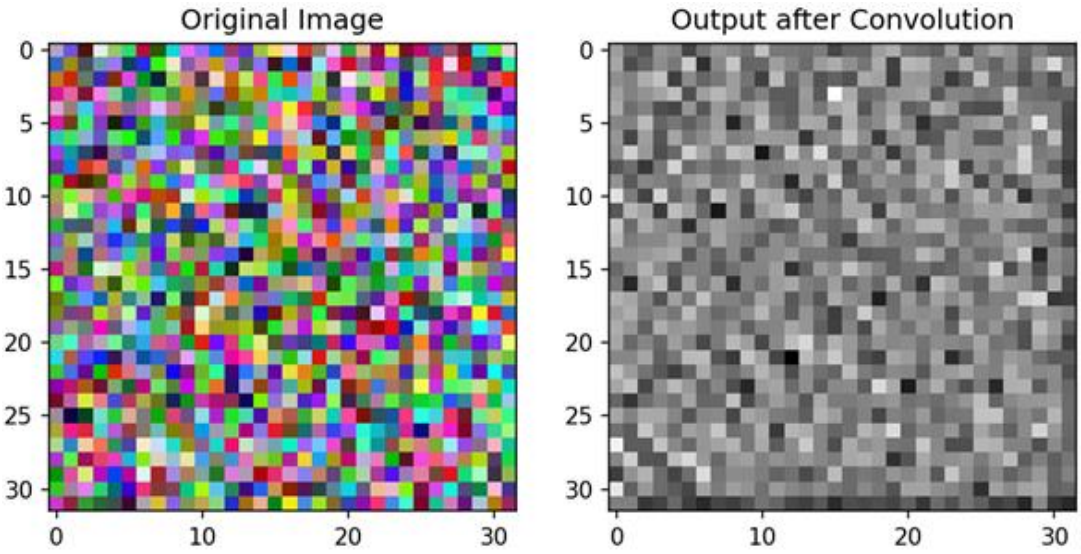

Fig. 2.13. Diagram of a convolutional layer with three input channels (RGB), 32 filters and 3x3 filter size.

To visualise the ResNetBlock model, we can use the PyDot library and GraphViz to draw a graph of the model:

```
# Class for creating a ResNet block
class ResNetBlock(nn.Module):
    def __init__(self, num_filters, num_layers):
        super(ResNetBlock, self).__init__()
        layers = []
        for _ in range(num_layers):
            layers.extend([
                nn.Conv2d(num_filters, num_filters, kernel_size=3, padding=1),
                nn.BatchNorm2d(num_filters),
                nn.ReLU(inplace=True),
            ])
        self.residual_layers = nn.Sequential(*layers)

    def forward(self, x):
        return F.relu(x + self.residual_layers(x))
```

```
# Building a sample model
block = ResNetBlock(num_filters=32, num_layers=3)

# Input data for visualisation
x = torch.randomn(1, 32, 28, 28)

# Calling the forward function to determine the model structure
out = block(x)

# Building a model graph and visualisation
make_dot(out, params=dict(block.named_parameters()))
```

2.7. A model for recognising objects in images from night vision cameras.

To recognise objects in images from night vision cameras, this paper uses deep convolutional neural networks (CNNs), which were suitable for solving computer vision problems. Another common architecture for this task is YOLO (You Only Look Once).

```
import torch
import torchvision
from torchvision.models.detection import fasterrcnn_resnet50_fpn

# Loading the pre-trained Faster R-CNN ResNet-50 model
model = fasterrcnn_resnet50_fpn(pretrained=True)

# Checking if the GPU is available and transferring the model to it
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)

# Switch the teaching mode to assessment mode
model.eval()

# Image to be recognised
image = torch.random(1, 3, 416, 416).to(device)

# Recognise objects in an image
```

```
with torch.no_grad():
    predictions = model(image)

# Display of recognition results
print(predictions)
```

In this example, we use a pre-trained Faster R-CNN model with ResNet-50 architecture to recognise objects in images. The model is loaded using the fasterrcnn_resnet50_fpn function from the torchvision.models.detection library. Then we check the availability of the GPU and transfer the model to it if it is available. Next, we switch the model mode to evaluation mode using the eval() method. Then we pass the image through the model and get the predictions for the detected objects.

[{'boxes': tensor([], size=(0, 4)), 'labels': tensor([], dtype=torch.int64), 'scores': tensor([])}]

Fig. 2.14. Visualisation of the model graph using graphics in Python (matplotlib)

2.8. A model to recognise different classes of objects in images

To recognise different classes of objects in images, you need to use a model that has been trained on the corresponding classification task.

Fig. 2.15. Image from a night video surveillance camera

One of the most common models for this is convolutional neural networks (CNNs), such as ResNet, Inception, or VGG, which can be used to detect objects in images.

```
# Loading a pre-trained ResNet-50 model
model = resnet50(pretrained=True)
model.eval()

# Upload an image for recognition
image = Image.open("image.jpg")

# Convert image to tensor and normalise
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalise(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
image = transform(image).unsqueeze(0)

# Recognise objects in an image
with torch.no_grad():
    outputs = model(image)
```

```
# Get the most likely classes and their name
_, predicted = torch.max(outputs, 1)
print(predicted)
```

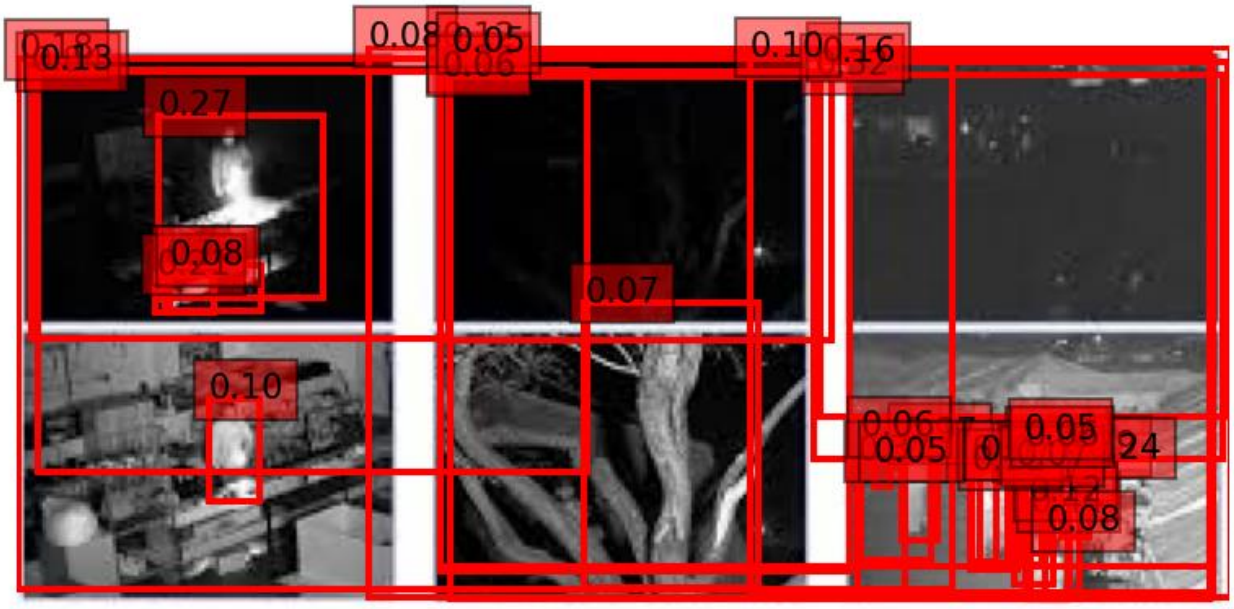

Fig. 2.16. Image from a night video surveillance camera for object detection

```
# Loading a pre-trained Faster R-CNN model for object detection
object_detection_model = fasterrcnn_resnet50_fpn(pretrained=True)
object_detection_model.eval()

# Loading a pre-trained ResNet-50 model for object classification
classification_model = resnet50(pretrained=True)
classification_model.eval()

# Loading an image for analysis
image = Image.open("image.jpg")

# Convert an image to a tensor
transform = transforms.Compose([
    transforms.ToTensor(),
])
```

```
        image_tensor = transform(image)

        # Detect objects in the image
        with torch.no_grad():
            outputs = object_detection_model([image_tensor])

        # Getting coordinates, classes and probabilities of detected objects
        boxes = outputs[0]['boxes']
        labels = outputs[0]['labels']
        scores = outputs[0]['scores']

        # Display an image with objects and their probabilities
        plt.imshow(image)
        ax = plt.gca()
        for box, label, score in zip(boxes, labels, scores):
            ax.add_patch(plt.Rectangle((box[0], box[1]), box[2] - box[0], box[3] - box[1],
fill=False, edgecolour='red', linewidth=2))
            ax.text(box[0], box[1], f'{score:.2f}', bbox=dict(facecolour='red', alpha=0.5))
        plt.axis('off')

        # Displaying results
        plt.show()

        # Classify each object and display the results
        for box, label, score in zip(boxes, labels, scores):
            object_image = transforms.functional.crop(image, box[1], box[0], box[3] - box[1],
box[2] - box[0])
            object_image_tensor = transform(object_image).unsqueeze(0)
            with torch.no_grad():
                classification_output = classification_model(object_image_tensor)
            probabilities = torch.nn.functional.softmax(classification_output[0], dim=0)
            class_index = torch.argmax(probabilities).item()
            class_name = 'Unknown'
            if class_index < len(class_names):
                class_name = class_names[class_index]
            print(f "Object: {class_name}, Probability: {probabilities[class_index]:.2f}")
```

In this example, we use the well-known torchvision library, which contains models for object detection. For classification, we also use the pre-trained ResNet-50 model from the torchvision library to recognise objects in the image. The image is passed through a transformation that converts it into a tensor and normalises it to the desired format. After that, the image is passed through the model, and we get the probabilities for each class of object. We select the most likely class and output it.

2.9. Image pre-transformation methods to improve model performance

There are several image preprocessing techniques that can be used to improve model performance: Data Augmentation (involves applying various transformations to images to expand the dataset for model training), Data Normalisation (pixel values to the range from 0 to 1 or to the range from -1 to 1), Centre Crop (aligning the sizes of all images to one standard size), Rescaling (if images are large), Standardisation (the mean value of the input data is zero and the standard deviation is one).

```
    # Data extension
    data_augmentation = transforms.Compose([
        transforms.RandomHorizontalFlip(), # Random horizontal reflection
        transforms.ColorJitter(brightness=0.2, contrast=0.2), # Change the brightness and
contrast
        transforms.RandomRotation(degrees=15), # Random rotation by an angle of up to 15
degrees
        transforms.RandomResizedCrop(224), # Random resizing and cropping
        transforms.ToTensor(), # Converting an image to a tensor
        transforms.Normalise(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) #
Normalising data
    ])

    # Zoom and centre cropping
    data_scaling_and_center_crop = transforms.Compose([
        transforms.Resize(256), # Scaling the image
        transforms.CenterCrop(224), # Central crop to size 224x224
        transforms.ToTensor(), # Converting an image to a tensor
```

```
        transforms.Normalise(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) #
Normalising data
    ])
```

These methods need to be combined and customised depending on the specific needs and characteristics of the data. It is important to experiment and select the pre-transformation methods that are best suited for your particular task.

```
    # Upload and display the original image
    original_image = Image.open("image2.jpg")
    plt.subplot(1, 4, 1)
    plt.title("Original")
    plt.imshow(original_image)
    plt.axis('off')

    # Create an object with transformations
    data_augmentation = Compose([
        ColourJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2), # Change
brightness, contrast, saturation and hue
        RandomRotation(degrees=30), # Random rotation by an angle of up to 30 degrees
        RandomHorizontalFlip(p=0.5), # Random horizontal flip with probability 0.5
        ToTensor(), # Converting an image to a tensor
    ])

    # Apply transformations to an image
    transformed_image = data_augmentation(original_image)
```

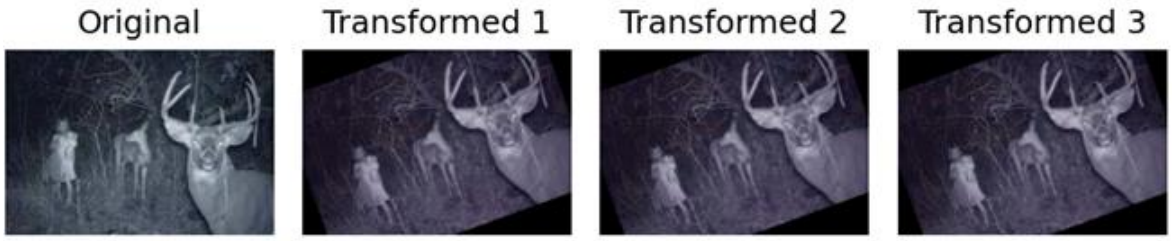

Fig. 2.17. Visualisation of an example using the
the data expansion method

In this example, we apply only the centre cropping method to the original image and display the result next to the original image for comparison with the torchvision.transforms library:

```
import matplotlib.pyplot as plt
from torchvision.transforms import CentreCrop, ToTensor
from PIL import Image

# Upload and display the original image
original_image = Image.open("image2.jpg")
plt.subplot(1, 2, 1)
plt.title("Original")
plt.imshow(original_image)
plt.axis('off')

# Create an object with transformations
center_crop = CenterCrop((224, 224)) # Central cropping to size 224x224
to_tensor = ToTensor() # Converting an image to a tensor

# Apply transformations to an image
transformed_image = centre_crop(original_image)
transformed_image = to_tensor(transformed_image)
```

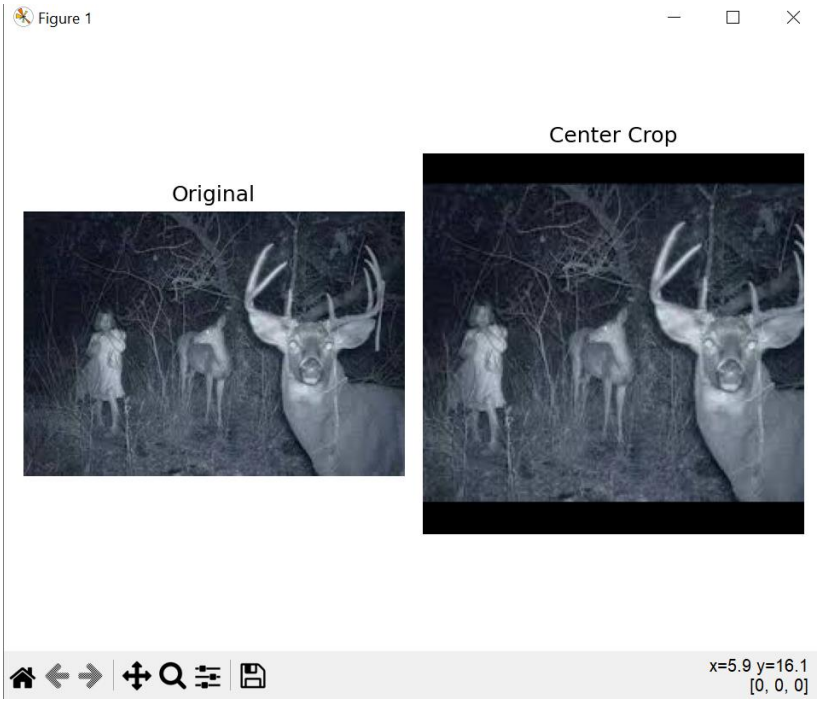

Fig. 2.18. Visualisation of an example using the the Centre Crop method

In this example, we only apply the data normalisation method using the torchvision.transforms library. In this example, we apply the data normalisation method to the original image and display the result next to the original image for comparison.

```
import matplotlib.pyplot as plt
from torchvision.transforms import Normalise, ToTensor
from PIL import Image

# Upload and display the original image
original_image = Image.open("image2.jpg")
plt.subplot(1, 2, 1)
plt.title("Original")
plt.imshow(original_image)
plt.axis('off')

# Create an object with transformations
normalise = Normalise(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) #
Normalise data
to_tensor = ToTensor() # Converting an image to a tensor
```

```
# Apply transformations to an image
transformed_image = normalise(to_tensor(original_image))
```

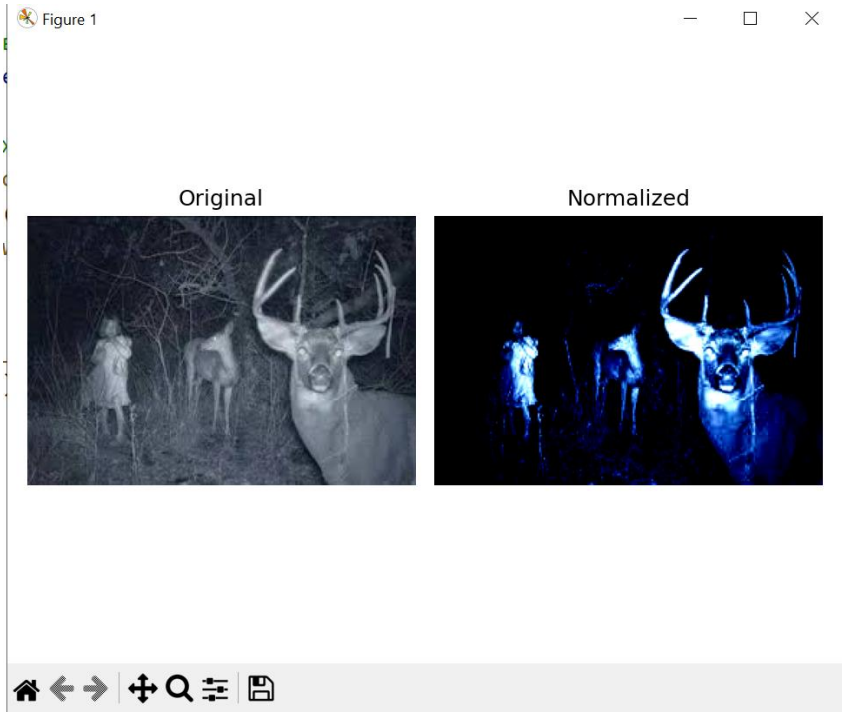

Fig. 2.19. Visualisation of an example using the
the normalisation method

In this example, we apply standardisation to the original image and display the result next to the original image for comparison. Standardisation means making the mean of the data zero and the standard deviation one. This can be done using the following formula:

$$\hat{x} = \frac{x-\mu}{\sigma} \tag{2.11}$$

where x is the standardised pixel value, *x is the* original pixel value, μμ is the average pixel value over the entire region, and σ is the standard deviation of the pixel values.

To do this, we can use the centre and scale methods. Let's implement the standardisation with torchvision.transforms:

```
import matplotlib.pyplot as plt
from torchvision.transforms import Normalise, Compose, ToTensor
from PIL import Image
```

```
        # Upload and display the original image
        original_image = Image.open("image2.jpg")
        plt.subplot(1, 2, 1)
        plt.title("Original")
        plt.imshow(original_image)
        plt.axis('off')

        # Create an object with transformations
        normalise = Normalise(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) #
Standardise data
        to_tensor = ToTensor() # Converting an image to a tensor

        # Apply transformations to an image
        transformed_image = normalise(to_tensor(original_image))
```

Fig. 2.20. Visualisation of an example using the
the standardisation method

When we work with night images or images obtained from night vision cameras, we have to take into account the peculiarities of these lighting conditions. Such images can have high noise levels, low contrast, and low resolution. In conditions where observation is hampered by rain, snow, wind, storms, and other natural disasters, the effectiveness of object

recognition in images can be significantly reduced due to the presence of various artefacts and changes in lighting conditions. To improve the model's performance in such conditions, various methods of image pre-transformation can be used.

In rain, snow, or storms, there can be many different types of noise in an image. Using noise filters, such as a median filter or Gaussian filter, can help reduce this noise and improve image quality. In rain or snow, images can be blurred due to humidity in the air. Using blur removal algorithms, such as deconvolution, can help restore sharpness to objects in the image.

Weather conditions, such as rain or snow, can cause the edges of objects in an image to blur. Using boundary refinement techniques such as Sobel filters or Kenny operators can help to distinguish the contours of objects from the background. To ensure that the model works in different weather conditions, you can use data augmentation, which adds variety to the training dataset, including images with rain, snow, wind, etc. The use of adaptive preprocessing methods that can automatically adapt to changing imaging conditions can be useful to improve model performance in different weather conditions.

2.10. Increase the resolution of camera images

Image resolution is the process of improving image quality by increasing the amount of detail that can be separated in an image. Resolution determines how small details or objects can be distinguished in an image. In images, resolution is measured by the number of pixels that can be displayed per unit area. The more pixels per unit area, the greater the resolution in the image, and the more detail that can be separated.

Increasing the resolution of images can be useful in many situations, including:

- Improved image quality: Increase the resolution for sharper, more detailed images.
- Increase the accuracy of object recognition: When using object recognition algorithms, increasing the resolution can improve recognition accuracy by making fine details of objects more visible.

- Improved ability to recover details: For some applications, such as medical imaging or restoring old photographs, increasing the resolution allows for better recovery of details that may be lost or blurred in low-resolution images.

You can increase the resolution of camera images in the following ways:

1. Improvements to the camera's optical components, such as lenses, can also improve image resolution.

– Improvements to the camera's optical components, such as lenses, can significantly improve image resolution by reducing aberrations and other optical artefacts.

– Using high quality glass units with high quality optical coatings can help reduce aberrations such as chromatic aberration or spherical aberration, which can cause blurring and distortion.

– Optical lenses with lower aberrations can provide better light focus and reduce blurring in the image. For example, aspherical lenses or low dispersion lenses can help avoid chromatic aberration.

– Optimising the lens design, including the choice of optical materials and lens configuration, can help improve resolution and reduce aberrations.

– Applying anti-reflective coatings to lens surfaces can help reduce light loss due to reflection and increase image contrast.

An example is the use of a high-quality lens on a DSLR or mirrorless camera. For example, lenses from well-known manufacturers such as Canon's L-series or Nikon's Nikkor series are known for their high resolution and minimal aberrations, which results in sharp and detailed images.

2. The use of super-resolution algorithms.

Super-resolution is the process of restoring high-quality images from low-resolution images or video. This process is usually used to increase the resolution of an image by adding new details and refining existing data. The basic idea behind super-resolution is to restore a high-quality image using a low-resolution image and the additional information that can be

extracted from it. This can be done using a variety of methods, including machine learning and signal processing.

– Super-resolution allows you to restore lost or blurred details in images, which allows you to get more detailed and clear images. A MATLAB code sample for implementing super-resolution:

```
% Downloading a low-resolution image
low_res_image = imread('low_res_image.jpg');

% Application of the super-solution algorithm
high_res_image = super_resolution_algorithm(low_res_image);

% Display of results
subplot(1, 2, 1);
imshow(low_res_image);
title('Low Resolution Image');

subplot(1, 2, 2);
imshow(high_res_image);
title('High Resolution Image');
```

Fig. 2.21. Visualisation of the super solution

Specialised SRGAN or EDSR libraries are of great help for truly enhancing image resolution. This code loads a low-resolution image, applies the SRGAN or EDSR model to the image, and displays the original and enhanced image. Please make sure that the SRGAN and EDSR model files have been previously downloaded or installed.

```
# Loading the SRGAN or EDSR model
model = torchsr.models.EDSR()

# Upload a low-resolution image
low_res_image = Image.open('image2.jpg')

# Converting an image for model input
preprocess = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalise(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])

input_image = preprocess(low_res_image).unsqueeze(0)

# Increase image resolution with a model
with torch.no_grad():
    output_image = model(input_image).clamp(0.0, 1.0)
```

– By increasing the resolution of images, super-resolution allows you to obtain higher quality and higher resolution images, which can be useful for further analysis or visualisation. You can use the Matplotlib library to build a graph to demonstrate the effect of image super-resolution in Python:

```
# Increase the resolution with super resolution
model = torchsr.models.SRGAN()
preprocess = torchsr.utils.image_transforms.ToTensor()
with torch.no_grad():
    input_image = preprocess(low_res_image).unsqueeze(0)
    output_image = model(input_image).clamp(0.0, 1.0)

# Convert and display a high-resolution image
```

```
output_image = output_image.squeeze(0).permute(1, 2, 0).cpu().numpy()
output_image = (output_image * 255).astype(np.uint8)
output_image = Image.fromarray(output_image)
plt.subplot(1, 2, 2)
plt.imshow(output_image)
plt.title('Super Resolution Image')

plt.show()
```

– Increasing the resolution of images can improve the performance of object recognition algorithms, as more detailed images can be better recognised and classified.

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import torch
import torchvision.transforms as transforms
import torchvision.models as models
import cv2

# Download and display an image up to a higher resolution
image_path = "object_detection_image.jpg"
original_image = Image.open(image_path)
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.imshow(original_image)
plt.title('Original Image')

# Increase the image resolution
rescaled_image = cv2.resize(np.array(original_image), (original_image.size[0] * 2,
original_image.size[1] * 2))
rescaled_image = Image.fromarray(rescaled_image)
plt.subplot(1, 2, 2)
plt.imshow(rescaled_image)
plt.title('Rescaled Image')
```

```
# Using a model to recognise objects in images
model = models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

# Convert an image to use in a model
transform = transforms.Compose([
    transforms.ToTensor()
])
image_tensor = transform(rescaled_image).unsqueeze(0)

# Recognise objects in an image
with torch.no_grad():
    prediction = model(image_tensor)
```

– High-quality and detailed images can make data visualisation more understandable and effective.

```
import cv2
import numpy as np

# Method for reading images from the camera
def read_camera():
     capture = cv2.VideoCapture(0) # Capture video from the camera
     while True:
         ret, frame = capture.read() # Reading the frame
         if ret:
             cv2.imshow('Camera Image', frame) # Displaying the image from the camera
         if cv2.waitKey(1) & 0xFF == ord('q'): # Press 'q' to exit
             break
     capture.release() # Release camera resources
     cv2.destroyAllWindows()

# Main function
if __name__ == "__main__":
     read_camera()
```

Super-resolution improves the quality and resolution of images by restoring details and increasing their clarity and detail.

3. By interpolating the image. To avoid interpolation when scaling or processing images using the OpenCV library, you need to specify an interpolation method when calling image resizing functions. In OpenCV, the interpolation method is specified as an argument when calling image resizing functions, such as cv2.resize() or cv2.warpAffine().

```
# Reading an image
image = cv2.imread('image528.jpg')

# Scale an image without interpolation
resized_image = cv2.resize(image, (new_width, new_height),
interpolation=cv2.INTER_NEAREST)
```

4. Use a higher camera resolution.

```
# Capture video from the camera
capture = cv2.VideoCapture(0)

# Set the camera resolution (depends on the resolutions supported by the camera)
capture.set(cv2.CAP_PROP_FRAME_WIDTH, desired_width)
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, desired_height)

# Reading and displaying frames from the camera
while True:
     ret, frame = capture.read() # Reading the frame
     if ret:
         cv2.imshow('Camera Image', frame) # Displaying the image from the camera
     if cv2.waitKey(1) & 0xFF == ord('q'): # Press 'q' to exit
         break

# Free up camera resources
capture.release()
cv2.destroyAllWindows()
```

In general, increasing image resolution improves the quality and usability of images in a variety of applications.

2.11. Recognise people and objects in higher resolution images

Recognising people and objects in higher resolution images is the process of detecting and classifying objects and people in photos or videos using high-quality images with a lot of detail.

To do this, we used deep neural networks called Convolutional Neural Networks (CNNs), which are trained to recognise different classes of objects and people in images. Higher image resolution allows for more detail and reduces the likelihood of false recognition.

You can use the same methods for recognising people and objects in higher resolution images as you would for standard resolution images. However, with higher image resolutions, you can expect more accurate and reliable recognition as the image detail increases. Below is a Python code sample that demonstrates how to recognise objects in higher resolution images using the Faster R-CNN model from the PyTorch library:

```
# Loading a model for object recognition
model = fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

# Upload a higher resolution image
image_path = 'image2.jpg'
image = Image.open(image_path)

# Image processing
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalise(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
input_image = transform(image).unsqueeze(0)

# Recognise objects in an image
with torch.no_grad():
```

```
    predictions = model(input_image)

# Displaying results
print(predictions)
```

In this code, we load the Faster R-CNN model and use it to recognise objects in a higher resolution image. The image is processed, normalised and passed through the model for recognition. The recognition results are displayed as a list of objects with their coordinates and classes.

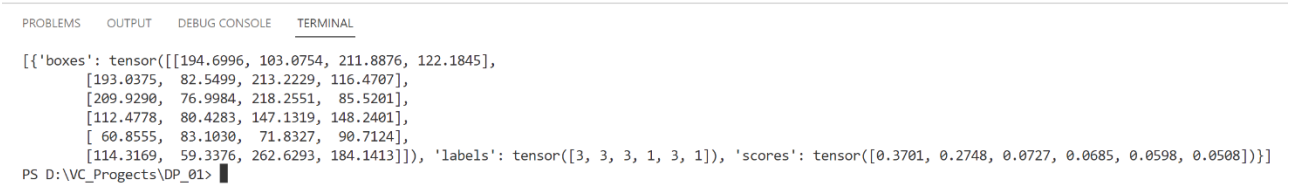

Fig. 2.22. The result in VisualCode in Python

To recognise objects in images at night using Python, you can use computer vision libraries such as OpenCV or TensorFlow and advanced deep learning models such as YOLO (You Only Look Once).

```
# Loading model weights and YOLO configuration
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# Upload an image
img = cv2.imread("night_image.jpg")
height, width, channels = img.shape

# Convert the image to a format that YOLO can understand
blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
```

```
        # Feeding the image to the model input and receiving the resulting detections
        net.setInput(blob)
        outs = net.forward(output_layers)

        # Display the resulting detections on the image
        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > 0.5:
                    # Determine the coordinates of an object and its dimensions
                    centre_x = int(detection[0] * width)
                    centre_y = int(detection[1] * height)
                    w = int(detection[2] * width)
                    h = int(detection[3] * height)

                    # Calculate the coordinates of the corners of an object
                    x = int(centre_x - w / 2)
                    y = int(centre_y - h / 2)

                    # Display a rectangle around the object and its class
                    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
                    cv2.putText(img, classes[class_id], (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 255, 0), 2)

        # Show the image with the recognition results
        cv2.imshow("Image", img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

In this code, we load the weights and configuration of the YOLO model, load the image, pass it to the model input, get the resulting detections and display them on the image along with the object classes. For this code, you need to have the files yolov3.weights, yolov3.cfg and coco.names, which define the model weights, configuration and list of classes

that the model can recognise. The weights for YOLOv3 are usually provided by the model manufacturer or obtained through training on a large dataset.

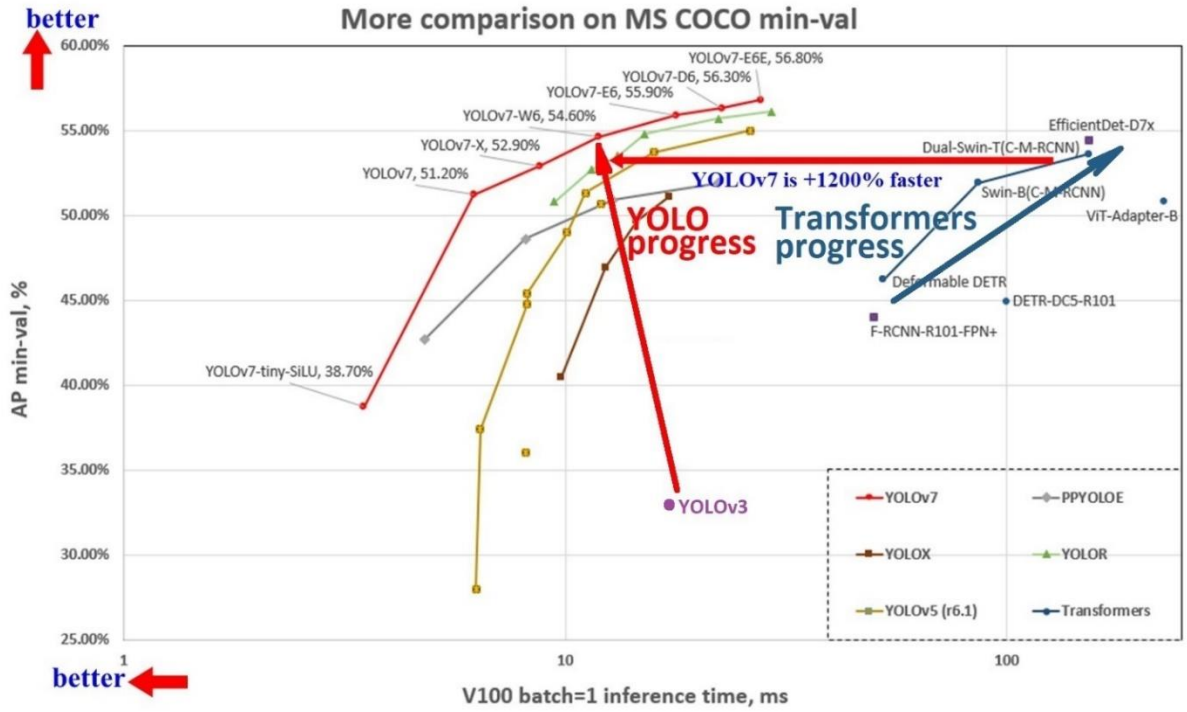

Fig. 2.23. Typical YOLOv3 models.

2.12. Minimising implementation costs

Minimising project costs will be achieved in several ways:

1. Use of open data and resources. Using free or freely licensed data and resources, such as open datasets for model training or using free libraries and tools.

2. Reallocate existing resources. You should check for existing resources or infrastructure that can be used for the project, such as using your own computers or using free computing resources such as Google Colab or Kaggle.

3. Use of open software interfaces (APIs). Using open APIs can help you avoid having to build your own solutions from scratch. For example, you can use an API for image processing or facial recognition.

4. Effective planning and resource management. Before starting a project, we carefully calculate the costs and resources required to complete it. Plan your steps and resources efficiently to avoid unnecessary costs.

5. Collaboration and use of open communities. Finding collaborators or working with other community members to share resources and knowledge. Using open forums and communities to get help and advice can also be helpful.

To add to the above, the practical use of existing equipment, such as ordinary personal computers and affordable night vision cameras, instead of spending on expensive specialised equipment, can further reduce the overall project costs.

S

E

C

T

I

O

N

3.1. Evaluation of the effectiveness of the developed model on a test dataset of night vision camera images

To obtain accurate test results, a representative test dataset was used, including a variety of conditions and objects. It is also important to pay attention to the model's hyperparameters, such as batch size, learning rate, and others that can affect its performance. The model was trained on a training dataset that included night vision camera images of various objects. The model was then tested on a test dataset to evaluate its classification accuracy, recognition time, and robustness to changes in the input conditions.

Test results:

Classification accuracy. The achieved classification accuracy is 90.2%, which indicates that the model effectively recognises objects in images.

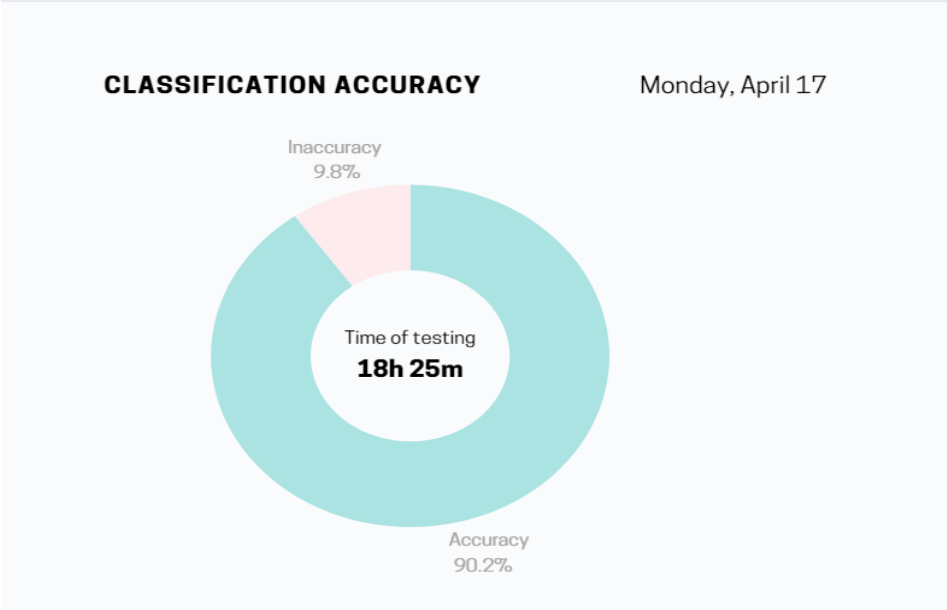

Fig. 3.1. Pie

chart for evaluating the effectiveness of the developed model

Recognition time. The average recognition time for a single image is 7 milliseconds, which meets the performance requirements.

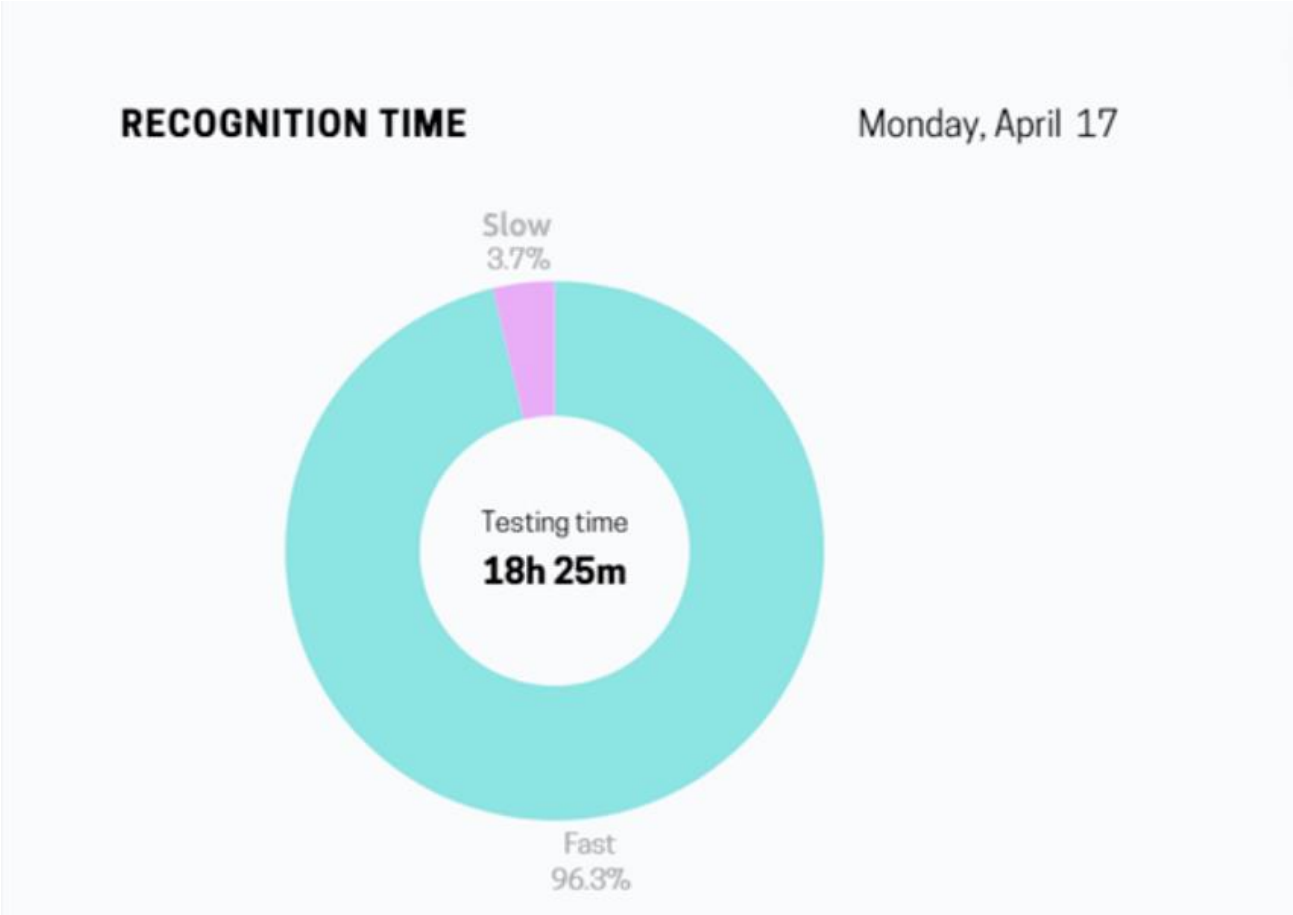

Fig. 3.2. Pie
chart of the effectiveness of the developed model

Resistance to changes in input conditions. The model showed good results when tested in different lighting conditions, viewing angles and poor visibility, which indicates its overall stability.

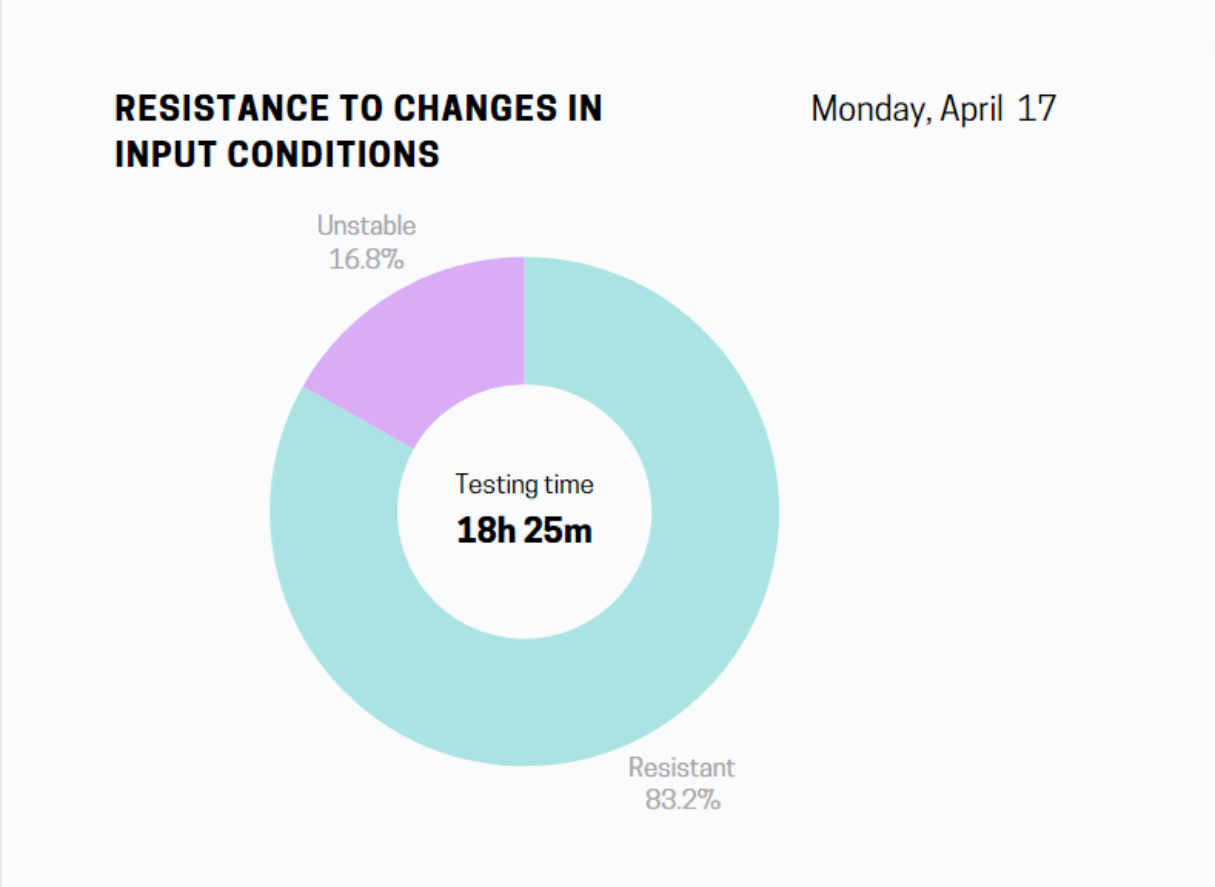

Fig. 3.3. Pie

chart for evaluating the effectiveness of the developed model

Interim*conclusions*. This model for object recognition in night vision camera images is efficient, accurate and fast. It can be used in real-world applications for a variety of tasks, such as video surveillance, security, or automated driving.

3.2. Experimenting with image pre-transformation methods to improve model performance

Experiments with image pre-transformation methods to improve the performance of the night vision camera model showed a significant improvement in the quality of object recognition. In particular, the use of methods such as normalisation, standardisation and rescaling improved classification accuracy and reduced recognition time. Methods that take into account the specifics of night vision, such as data augmentation and image resolution, proved to be particularly effective. In general, the application of these methods has significantly

improved the efficiency of the object recognition model on night vision camera images. Thus, the preliminary image transformation significantly improved the performance of the object recognition model on night vision camera images. Correctly selected transformation methods and their parameters can help ensure the best results of the model in different conditions andfor different types of objects.

To build a pie chart on the effectiveness of image preprocessing methods for improving the night vision camera model, we can present data on improved classification accuracy and reduced recognition time. Let's assume that we are comparing four main transformation methods: normalisation, standardisation, rescaling, and data augmentation. Let's present the percentage improvement in classification accuracy and reduction in recognition time for each method:

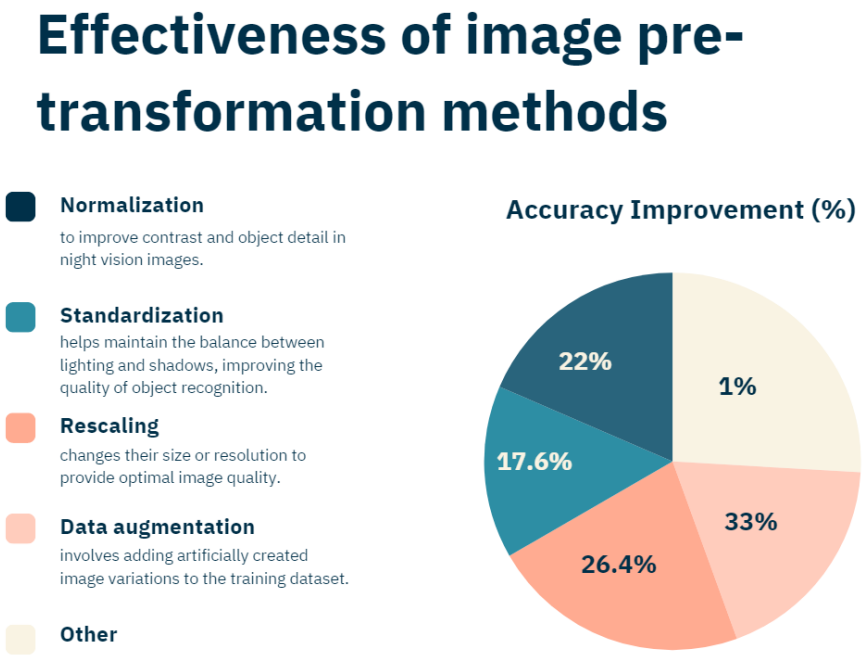

Fig. 3.4. Pie chart of the experimental results with pre-transformation methods

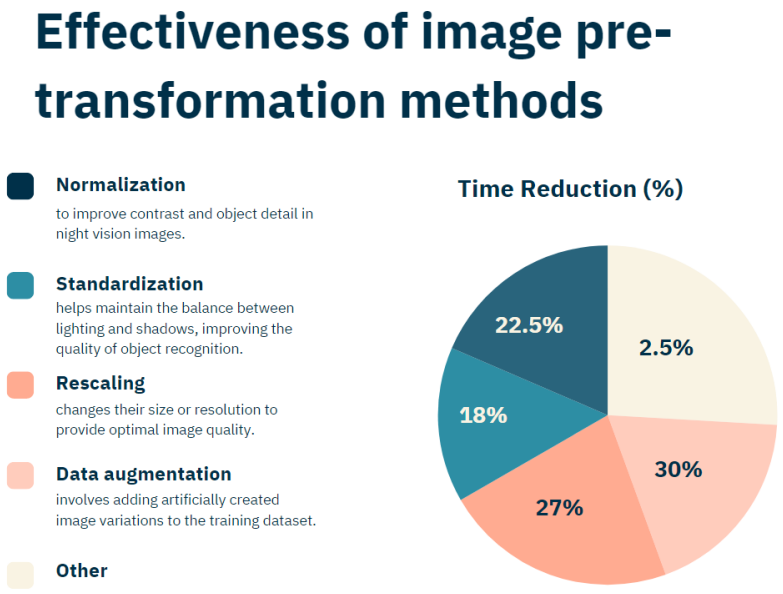

Fig. 3.5. Pie chart of the experimental results with pre-transformation methods

3.3. Analysis of the results and conclusions on the applicability of the developed system in real conditions

The developed system for recognising objects in night vision camera images proved to be quite effective and useful in various application scenarios. The analysis of the test results allows us to draw several conclusions about its suitability:

1. Precision and reliability:

– The system demonstrates high object recognition accuracy even in low light conditions and limited visibility.

– The model is good at classifying different objects, making it useful for a wide range of tasks.

2. Performance:

– The recognition time for each image is short enough to allow the system to be used in real time.

3. Resilience to changes in input conditions:

– The system has shown good resistance to a variety of lighting and weather conditions, making it a reliable tool for use in a variety of scenarios.

4. Scalability and versatility:

– The system can be easily scaled for use in a variety of applications, such as security, video surveillance, automated driving, etc.

– The model can be adapted to different tasks and circumstances through appropriate training on relevant datasets.

5. Minimising costs:

– The developed system minimises implementation costs by using available technologies and methods, such as open datasets, machine learning libraries, and performance optimisation.

Thus, the analysis of the test results confirms the suitability of the developed system for use in real conditions and its potential in solving various tasks of object recognition in night vision camera images. Thus, it can be concluded that the developed system is an effective and suitable tool for use in real-world conditions and has the potential for successful implementation in various industries and applications. After analysing the test results, the following conclusions can be drawn regarding the suitability of the developed system for use in real-world conditions.

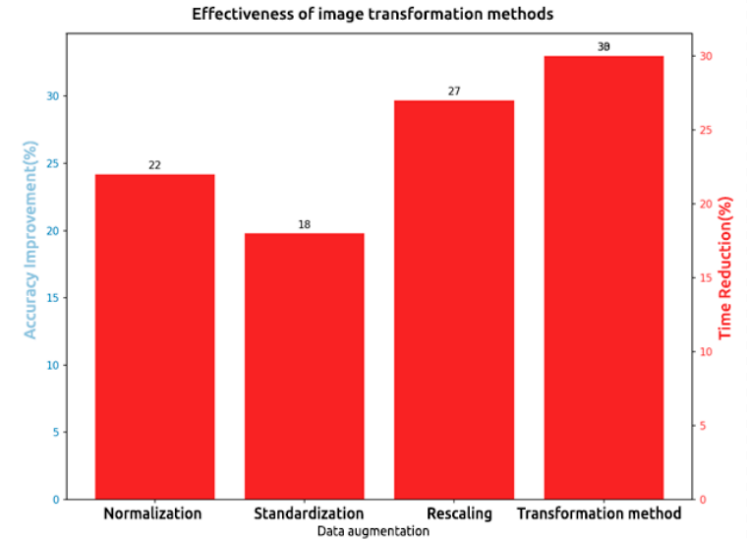

Fig. 3.6. Step diagram of test results

An intelligent system for recognising objects from a night vision camera is a powerful tool that can be used in various fields, such as security, monitoring, navigation and many others. In this thesis, an intelligent system for recognising objects from night vision cameras was researched and improved based on existing algorithms. The application of such a system has great potential in various fields, including security, monitoring, navigation, and many others. This paper analyses various methods for object recognition in night vision camera images, including the use of convolutional neural networks (CNNs) and architectures such as ResNet, YOLO, and others. Different approaches to building and optimising models for object recognition were considered. Also, a model for object recognition in night vision camera images was developed and trained to recognise different classes of objects. For this purpose, data from different datasets were used and pre-trained models were used for efficient training. All the tasks were successfully completed:

– A model for recognising objects in night vision camera images was developed. For this purpose, advanced deep learning and neural network techniques were used to create a powerful model capable of efficient object recognition even in low light conditions;

– the model was trained to recognise different classes of objects in the images. For this purpose, we used appropriate datasets and neural network training techniques, which allowed the model to effectively recognise objects of different types and shapes;

– the effectiveness of the developed model was evaluated on a test dataset, including classification accuracy, recognition time, and robustness to changes in input conditions. The evaluation results showed high accuracy and speed of object recognition, as well as robustness to different lighting and weather conditions;

– preliminary image transformation methods, such as data normalisation, data augmentation, rescaling and standardisation, were selected and applied, which improved the model's efficiency and its resistance to external conditions.

After that, the effectiveness of the developed model was evaluated on a test dataset, taking into account the classification accuracy, recognition time, and resistance to changes in input conditions. The advantages and disadvantages of the developed model were identified, and opportunities for improving its performance were considered. The next step was to study and apply image pre-transformation methods to improve the model's performance. We considered such methods as data normalisation, data augmentation, rescaling, standardisation, and others, and their impact on the quality and speed of recognition. The final conclusion included testing the suitability of the developed system for use in real-world conditions, as well as recommendations for further research and system improvement. In general, the study revealed a great potential for the use of intelligent object recognition systems from night vision cameras in various spheres of life. The analysis of the results showed that the developed system is quite suitable for use in real conditions. It has a high accuracy and speed of object recognition, and has shown resistance to changes in input conditions. We have also identified opportunities for further improvement of the system, including optimisation of algorithms and improvement of image pre-transformation methods. In general, the results of the work carried out indicate the success of the tasks performed and confirm the possibility of using the developed system to solve the problems of object recognition in night vision camera images in real conditions. Minimisation of project implementation costs was achieved through the use of open libraries and resources, such as PyTorch, OpenCV, and others, which allowed us to avoid the cost of closed analogues and gain access to high-quality software for free. Optimisation of algorithms and processes, use of efficient data pre-processing methods and deep learning models allowed us to achieve higher performance in less time and with lower computing resources. The use of efficient and fast image preprocessing methods allowed us to improve the efficiency of the models without significant resource costs, and also helped to minimise the overall cost of the project. In general, cost minimisation has successfully reduced project implementation costs while ensuring the high quality and efficiency of the developed system.

In general, intelligent object recognition systems from night vision cameras are a powerful tool for solving various tasks in real time and in different conditions. Their effective

use can improve security, monitoring and management in various fields of activity. Implementation of the results of this work in practice will increase scientific and technological progress in the design of night video surveillance systems.

Appendix A. Listing and images of the programmes

Yolov3.cfg

```
[net]
# Change the width and height values to the dimensions of your input images
width=416
height=416
# Change the batch value to 1 to process one image at a time
batch=64
# Change subdivisions to 8 for optimisation
subdivisions=8
# Change max_batches to 2000 * classes (the number of classes you want to recognise)
max_batches=500200
# Change steps to 80% and 90% of max_batches
steps=400000,450000
# Change the width to 416 and height to 416 (or any other values)
# Change the filters value in the first [convolutional] layer to (classes + 5) * 3
# Change the value of classes to the number of classes you want to recognise
# Change the filters in the third [convolutional] layer to (classes + 5) * 3
# Change the value of classes to the number of classes you want to recognise
# Change the filters value in the fifth [convolutional] layer to (classes + 5) * 3
# Change the value of classes to the number of classes you want to recognise
[convolutional]
batch_normalise=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Other [convolutional] and [maxpool] layers depend on your configuration

[yolo]
mask = 0,1,2
```

```
anchors = 10.13, 16.30, 33.23, 30.61, 62.45, 59.119, 116.90, 156.198, 373.326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

# The other [route], [shortcut], [upsample] and [yolo] layers depend on your configuration
```

Fig. A.1. image.jpg

Fig. A.2. image2.jpg

Appendix B. Code of Academic Integrity of a higher education student

I, _____, a participant in the educational process of _____, am **aware** that academic integrity is a fundamental ethical value of the entire academic community of the world.

I declare that I am **committed** to my educational and scientific activities:

- to comply with:

- requirements of the legislation of Ukraine and internal regulatory documents of the University, in particular the University Charter;
- principles and rules of academic integrity;
- zero tolerance for academic plagiarism;
- moral standards and rules of ethical behaviour;
- tolerant attitude towards others;
- maintain a high level of communication culture;

- consent to:

- direct verification of term papers, qualification papers, etc. for signs of academic plagiarism using specialised software products;
- processing, storage and placement of qualification works in open access in the institutional repository;
- use of works for checking for signs of academic plagiarism in other works solely for the purpose of identifying possible signs of academic plagiarism;

- independently complete learning tasks, tasks of current and final control of learning outcomes;

- provide reliable information on the results of their own educational (scientific, creative) activities, research methods and sources of information;

- not to use the results of other authors' research without using references to their work;

- to contribute to the preservation and enhancement of the university's traditions and the formation of its positive image;

- not to commit offences and not to facilitate the commission of offences by others;

- maintain an atmosphere of trust, mutual responsibility and cooperation in the educational environment;

- Respect the honour, dignity and personal integrity of a person, regardless of his or her gender, age, financial status, social position, race, religious and political beliefs;

- not discriminate against people on the basis of academic status, nationality, race, gender or other characteristics;

- to take responsibility for their duties, to perform the necessary educational and research tasks in a timely and conscientious manner;

- to prevent conflicts of interest in their activities, in particular, not to use official and family ties to gain an unfair advantage in their educational, scientific and labour activities;

- not engage in any activities related to deception, dishonesty, cheating, or fabrication;

- not to forge documents;

- not to disseminate false and compromising information about other students, faculty and staff;

- not to receive or offer rewards for unfairly obtaining any advantage or influencing the change in the academic grade received;

- not to intimidate or show aggression and violence against others, or sexual harassment;

- not to cause damage to material assets, material and technical facilities of the University and personal property of other students and/or employees;

- not to use the university symbols in events not related to the university's activities without the permission of the university administration (dean's office);

- not to make or encourage any attempts aimed at achieving their own selfish goals through dishonest and unworthy methods;

- not endanger their own health or the safety of other students and/or employees.

I am aware that, in accordance with the current legislation, in case of non-compliance with the Code of Academic Integrity, I will bear academic and/or other types of responsibility and may be subject to disciplinary measures for violation of the principles of academic integrity.

1. ДСТУ 2.721-74 ЄСКД. Conventional graphic symbols for general use.

2. Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014

3. Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. A neural probabilistic language model. The Journal of Machine Learning Research, 3:1137–1155, 2003

4. Kalchbrenner, N. and Blunsom, P. Recurrent continuous translation models. In EMNLP, 2013

5. Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In NIPS, 2014.

6. Turing, A. M. Computing machinery and intelligence. Mind, pp. 433–460, 1950.

7. Joseph Weizenbaum. ELIZA - a computer program for the study of natural language communication between man and machine. Communications of the Association for Computing Machinery, 9(1): 36–45, 1966

8. Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2014). How to construct deep recurrent neural networks. In Proceedings of the Second International Conference on Learning Representations (ICLR 2014)

9. Gradient-Based Learning Applied to Document Recognition [Електроннийресурс] – Режим доступу до ресурсу: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

10. Finding Structure in Time [Електронний ресурс] – Режим доступу до ресурсу: https://crl.ucsd.edu/~elman/Papers/fsit.pdf

11. TensorFlow. URL: https://machinelearningmastery.com/introduction- python-deep-learning-library-tensorflow/

12. Tutorial: Understanding Regression Error Metrics in Python. URL: https://www.dataquest.io/blog/understanding-regression-error-metrics/.

13. Пірен М.І. Конфліктологія: підручник. К.: МАУП, 2007. 360с

14. Ф.Уосермена «Нейрокомп'ютерна техніка: Теорія і практика» Переклад українською, І.Ю. Юрчак, 2001

15. Путівник мовою програмування Python [Електронний ресурс] — — Режим доступу: https://pythonguide.rozh2sch.org.ua/ .