FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Developing Image Similarity Solutions for Smart Urban Monitoring

**Nuno André Gomes França**




Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Nuno Feixa Rodrigues

Second Supervisor: Prof. António Coelho

February 9, 2026

# Developing Image Similarity Solutions for Smart Urban Monitoring

**Nuno André Gomes França**

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Alexandre Valle
Referee: Prof. José Torres
Referee: Prof. António Coelho

February 9, 2026

# Resumo

O projeto *Theia Vision* recentemente tem capturado imagens urbanas numa sequência semelhante a um disparo contínuo de máquinas fotográficas, com o objetivo de detectar ocorrências urbanas, como por exemplo, graffiti, lixo em locais incorretos, e mobília abandonada. Devido a uma alta captura de informação, a redução de imagens redundantes é essencial. Este estudo investiga a detecção de imagens semanticamente semelhantes como forma de reduzir o número total de imagens que são armazenadas. Assim, as propostas técnicas de classificação de imagens como semanticamente semelhantes são avaliadas tanto no dispositivo como no servidor, utilizando um conjunto de dados previamente recolhido, no qual ocorrências urbanas foram manualmente anotadas. São exploradas tanto técnicas de visão por computador tradicionais, como de deep learning para determinar se duas imagens podem ser consideradas semanticamente semelhantes, sendo que as abordagens de deep learning variam entre embeddings globais e a utilização de modelos de deteção de objetos para criar uma representação semântica através dos objetos detectados e das respetivas posições.

Os resultados experimentais mostram que a maioria dos métodos consegue descartar pelo menos 50% das imagens, preservando mais de 95% das ocorrências anotadas. Contrariamente à hipótese formulada, as técnicas de deep learning baseadas em detecção de objetos não superaram os métodos tradicionais. As abordagens tradicionais mostraram-se não só mais rápidas, como também menos dispendiosas em termos computacionais e mais eficazes na maioria dos cenários. Em particular, perceptual hashing preservou 100% das ocorrências anotadas enquanto descartava 43,59% das imagens, e 97,56% enquanto descartava 67,97%. De forma semelhante, a estratégia de SIFT keypoint mask preservou 100% das ocorrências enquanto descartava 54,15% das imagens, e 94,82% enquanto descartava 74,62%.

As técnicas de deteção de objetos baseadas em deep learning apresentaram um desempenho inferior, que é atribuído ao foco em objetos proeminentes, sendo mais sensíveis a pequenos deslocamentos espaciais. Tentativas de mitigar estes efeitos através de estratégias de correspondência relaxadas resultaram num aumento de falsos positivos. Entre estes métodos, o YOLOv13 alcançou uma retenção máxima de ocorrências de 98,78% com 35,95% das imagens descartadas, enquanto o OWL-ViT reteve 92,38% das ocorrências com uma taxa de descarte de 36,62%. O Grounding DINO e o Florence-2 alcançaram, respectivamente, uma retenção de 100% e 97,87% das ocorrências apenas com taxas de remoção de 31,77% e 45,60%.

Entre as técnicas de deep learning avaliadas, os embeddings globais foram a exceção, preservando o total de ocorrências ao descartar 48,60% das imagens e preservando 91,46% das ocorrências anotadas ao descartar quase 86,93% das imagens no conjunto de dados avaliado.

No geral, os resultados demonstram que as técnicas tradicionais avaliadas continuam altamente competitivas para a remoção de informação redundante em cenários de monitorização urbana, particularmente quando é necessário um processamento rápido e leve. Embora os resultados de CLIP ofereçam uma robustez ligeiramente superior em taxas extremas de remoção de imagens, por consequência também tem custos computacionais ligeiramente mais elevados, as estratégias

de similaridade baseadas em detecção de objetos não são adequadas para a deteção de imagens semanticamente semelhantes em condições realistas de condução urbana.

# Abstract

The Theia Vision Project captures burst-shot like sequences of urban images to detect urban occurrences such as graffiti, improper waste disposal, and abandoned furniture. Due to the high volume of captured data, reduction of redundant images is essential. This study investigates the detection of semantically similar images as a means of reducing the total number of images stored. The proposed approaches are evaluated both on-device and serves-side using a previously collected dataset in which unique occurrences were manually annotated. Both traditional and deep learning computer vision techniques are explored to determine whether two images can be considered semantically similar, with deep learning approaches ranging from global embeddings, to leveraging object detection models to create visual footprints by the detected objects and their respective positions.

Experimental results show that most methods are able to discard at least 50% of the images while preserving more than 95% of the annotated occurrences. Contrary to the formulated hypothesis, object detection deep learning techniques did not consistently outperform traditional methods. Traditional approaches proved to not only be faster, but also less computationally expensive, and more effective in most scenarios. In particular, perceptual hashing preserved 100% of the annotated occurrences while discarding 43.59% of the images, and 97.56% while discarding 67.97%. Similarly, the SIFT keypoint mask strategy preserved 100% of occurrences while discarding 54.15% of the images, and 94.82% while discarding 74.62%.

The inferior performance of object detection techniques based on deep learning is attributed to the focus on prominent objects and, curiously, they are more sensitive to small spatial shifts. Attempts to mitigate these effects through relaxed matching strategies resulted in increased false positives. Among these methods, YOLOv13 achieved a maximum occurrence retention of 98.78% with 35.95% of images discarded, while OWL-ViT retained 92.38% of occurrences at a discard rate of 36.62%. Grounding DINO and Florence-2 achieved, respectively, 100% and 97.87% occurrence retention only at conservative removal rates of 31.77% and 45.60%.

Across the deep learning techniques evaluated, global embeddings was the outlier, preserving the total number of occurrences when discarding 48.60% of the images, and preserving 91.46% of the annotated occurrences while discarding almost 86.93% of the images in the evaluated dataset.

Overall, the results demonstrate that traditional computer vision techniques remain highly competitive for near-duplicate image detection in urban monitoring scenarios, particularly when fast and lightweight processing is required. With CLIP embeddings providing slightly higher robustness at extreme image removal rates with also slightly higher computational costs, object detection similarity strategies are not well suited for near-duplicate detection under realistic urban driving conditions.

# UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

In this context, the development of a computer vision model able to reduce internet traffic through image similarity detection directly contributes to some of the stated goals, where the network efficiency is optimized and there is a reduction in energy consumption. This dissertation explores how this work aligns with the SDGs and its impact on the world.

The specific Sustainable Development Goals have the following names:

**SDG 7** Ensure access to affordable, reliable, sustainable and modern energy for all;

**SDG 9** Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation;

**SDG 13** Take urgent action to combat climate change and its impacts.

| **SGD** | **Target** | **Contribution** | **Performance Indicators and Metrics** |
|---|---|---|---|
| 7 | 7.3 | Reducing redundant image transmissions, lowering energy use in networks | Reduction in total data transmitted |
| 9 | 9.4 | Optimizing data transmission infrastructure, making networks more efficient | Reduction in data transmitted by edge devices |
| 13 | 13.2 | Reducing energy waste by filtering the data before sending it to the central server | Filtered out data in edge devices |

# Acknowledgements

I would like to express my deepest gratitude to my supervisors, Professor Nuno Feixa Rodrigues and Professor António Coelho, for their guidance and support throughout this work. Wihout them, I would have been completely lost.

I am also grateful to my doctoral colleague, André Torneiro, for his constant availability and insightful discussions.

Finally, I would like to thank my future wife, my brother, and my parents, that were always were there for me, giving me the strength to continue.

Nuno França

# Contents


**1 Introduction** **1**
- 1.1 Context and Motivation . . . 1
- 1.2 Problem Identification . . . 1
- 1.3 Dissertation Structure . . . 4

**2 State of The Art** **5**
- 2.1 Methodology . . . 5
- 2.2 Literature review results . . . 6
  - 2.2.1 Summary of results . . . 7
- 2.3 Near-Duplicate Image Problem . . . 8
- 2.4 Background . . . 9
  - 2.4.1 Preprocessing techniques . . . 9
  - 2.4.2 Traditional Feature Extraction Methods . . . 10
  - 2.4.3 Hashing Techniques for Image Similarity Identification . . . 11
- 2.5 Deep Learning Approaches . . . 12
  - 2.5.1 What is a Convolutional Neural Network? . . . 12
  - 2.5.2 CNN-Based Similarity Models . . . 13
  - 2.5.3 Transformer-Based Vision Models . . . 16

**3 Problem Statement** **17**
- 3.1 Problem . . . 17
- 3.2 Hypothesis . . . 18
- 3.3 Research Questions . . . 18
- 3.4 Solution Requirements . . . 19
- 3.5 Scope . . . 20
- 3.6 Methodology Plan . . . 20

**4 Materials and Methods** **22**
- 4.1 Dataset Preparation . . . 22
  - 4.1.1 Occurrence Detection . . . 23
  - 4.1.2 Occurrence Characteristics . . . 25
- 4.2 Near-Duplicate Image Removal Algorithms . . . 28
  - 4.2.1 Traditional Similarity Algorithms . . . 28
  - 4.2.2 Deep Learning Similarity Algorithms . . . 31
- 4.3 Iterative Image Comparison Procedure . . . 35
- 4.4 Evaluation Methodology . . . 36
- 4.5 Testing Systems . . . 39

**5 Results** **41**
5.1 Control Algorithm . . . 42
5.1.1 Control Algorithm - Global Evaluation Methodology . . . 42
5.1.2 Control Algorithm - Focused Evaluation Methodology . . . 43
5.2 Traditional Similarity Techniques . . . 44
5.2.1 Traditional Similarity Techniques - Global Evaluation Methodology . . . 44
5.2.2 Traditional Similarity Techniques - Focused Evaluation Methodology . . . 45
5.3 Deep Learning Techniques . . . 46
5.3.1 Deep Learning Techniques - Global Evaluation Methodology . . . 46
5.3.2 Deep Learning Techniques - Focused Evaluation Methodology . . . 48
5.4 Selected Similarity Techniques . . . 51
5.5 Near-Duplicate Classification Time Metrics . . . 52
5.6 Edge-Device Logged Metrics . . . 55

**6 Discussion** **59**
6.1 Traditional Similarity Techniques . . . 59
6.2 Deep Learning Similarity Techniques . . . 60
6.3 Runtime and Deployment Evaluation . . . 63

**7 Conclusions and Future Work** **65**
7.1 Future Work . . . 66

**References** **67**

**A Appendix: Edge-Device Metrics** **70**
A.1 Edge-Device Metrics for Remaining Algorithms . . . 70
A.2 Edge-Device Active Cooling Results . . . 78

# List of Figures


1.1 Consecutive images representing near-duplicates. . . . . 2
1.2 Consecutive images representing near-duplicates. . . . . 3
1.3 Consecutive images representing near-duplicates. . . . . 3

4.1 Schematic representing an error found in the new API dataset . . . . . 22
4.2 Figure with examples of the annotation tool. . . . . 24
4.3 Consecutive images representing a distinct *blocked road sign* bag. . . . . 25
4.4 Consecutive images representing a distinct *improper waster disposal* bag. . . . . 25
4.5 Number of images per occurrence bag and box plot of the distribution. . . . . 26
4.6 Frequency of distinct urban occurrence classes. . . . . 26
4.7 Figure with variation metrics across distinct urban occurrences bags. . . . . 27
4.8 Traversal Path with Occurrences. . . . . 28
4.9 SIFT keypoint mask - Determining keypoints. . . . . 29
4.10 SIFT keypoint mask - Drawing keypoints canvas. . . . . 30
4.11 SIFT keypoint mask - Drawing matched keypoints canvas. . . . . 30
4.12 SIFT keypoint mask - Determining contours. . . . . 30
4.13 Grounding Dino object detection example - urban objects. . . . . 32
4.14 OWL-ViT object detection example - urban objects. . . . . 33
4.15 Florence-2 object detection example - OD Prompt. . . . . 34
4.16 YOLOv13 object detection example - No Prompt. . . . . 35
4.17 Pairwise Comparison Iteration. . . . . 36
4.18 Dataset conceptual Overview. . . . . 36
4.19 False positive first example. . . . . 38
4.20 False positive second example. . . . . 39

5.1 Pareto Chart with control algorithm results - global evaluation methodology. . . . 42
5.2 Pareto Chart with control algorithm results - focused evaluation methodology. . . 43
5.3 Pareto Chart with traditional similarity techniques - global evaluation methodology. 44
5.4 Legend for Figure 5.3. . . . . 45
5.5 Pareto Chart with traditional similarity techniques - focused evaluation methodology. 46
5.6 Legend for Figure 5.5. . . . . 46
5.7 Pareto Chart with deep learning similarity techniques - global evaluation methodology. . . . . 48
5.8 Legend for Figure 5.7. . . . . 48
5.9 Pareto Chart with deep learning similarity techniques - focused evaluation methodology. . . . . 50
5.10 Legend for Figure 5.9. . . . . 50
5.11 Pareto Chart with selected similarity techniques - global evaluation methodology. 51

5.12 Legend for both Figure 5.11 and 5.13. . . . . 51
5.13 Pareto Chart with selected similarity techniques - focused evaluation methodology. 52
5.14 Traditional similarity techniques - Time distribution. . . . . 52
5.15 Deep learning similarity techniques - Time distribution. . . . . 53
5.16 Traditional similarity techniques - Total execution time. . . . . 54
5.17 Deep learning similarity techniques - Total execution time. . . . . 54
5.18 SIFT Keypoint Mask metrics - edge device. . . . . 55
5.19 Perceptual Hashing metrics - edge device. . . . . 56
5.20 Grounding DINO metrics - edge device. . . . . 57
5.21 CLIP Embeddings metrics - edge device. . . . . 58

6.1 Grounding DINO example - low number of detected objects. . . . . 62
6.2 Grounding DINO example - incorrectly detected objects. . . . . 62

A.1 SSIM metrics - edge device. . . . . 71
A.2 Florence OD metrics - edge device. . . . . 72
A.3 OWL ViT metrics - edge device. . . . . 73
A.4 YOLOv13 n - edge device. . . . . 74
A.5 YOLOv13 s - edge device. . . . . 75
A.6 YOLOv13 l - edge device. . . . . 76
A.7 YOLOv13 x - edge device. . . . . 77
A.8 Perceptual Hashing active cooling - edge device. . . . . 78
A.9 CLIP active cooling - edge device. . . . . 79
A.10 Florence 2 OD active cooling - edge device. . . . . 80

# List of Tables


4.1 Variation metrics across distinct urban occurrences bags. . . . . 27

5.1 Control algorithm metrics - global evaluation methodology. . . . . 42
5.2 Control algorithm metrics - focused evaluation methodology. . . . . 43
5.3 Traditional similarity techniques metrics - global evaluation methodology. . . . . 44
5.4 Traditional similarity techniques metrics - focused evaluation methodology. . . . 45
5.5 Deep Learning Similarity Techniques Metrics - global evaluation methodology. . 47
5.6 Deep Learning Similarity Techniques Metrics - focused evaluation methodology. 49

# List of Acronyms

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **aHash** | Average Hash |
| **BoVW** | Bag-of-Visual Words |
| **CNNs** | Convolutional Neural Networks |
| **COCO** | Common Objects in Context |
| **CPU** | Central Processing Unit |
| **CUDA** | Compute Unified Device Architecture |
| **dHash** | Difference Hash |
| **GPU** | Graphics Processing Unit |
| **HE** | Hamming Embedding |
| **IoT** | Internet of Things |
| **LSH** | Locality-Sensitive Hashing |
| **mAP** | Mean Average Precision |
| **NDD** | Near-Duplicate Detection |
| **ORB** | Oriented FAST and Rotated BRIEF |
| **pHash** | Perceptual Hash |
| **PRISMA** | Preferred Reporting Items for Systematic Reviews and Meta-Analyses |
| **R-CNN** | Region-based Convolutional Neural Network |
| **RAM** | Random Access Memory |
| **SDG** | Sustainable Development Goals |
| **SIFT** | Scale-Invariant Feature Transform |
| **SURF** | Speeded-Up Robust Features |
| **ViT** | Vision Transformer |
| **wHash** | Wavelet Hash |
| **YOLO** | You Only Look Once |

# Chapter 1

# Introduction

The expansion of urban areas has led to an increased need for efficient city management. Smart urban monitoring, facilitated by artificial intelligence (AI) and the internet of things (IoT), plays a crucial role in improving security, traffic control, and environmental monitoring.

## 1.1 Context and Motivation

The INESC TEC team, supervised by professor Nuno Feixa Rodrigues, has been developing *Project Theia Vision*, which aims to improve urban living by leveraging smart monitoring systems. This project focuses on capturing images from the urban environment at a steady frame rate, with multiple users collecting information through edge devices. At the end of each collection cycle, the images are sent to a centralized database, where an occurrence detection algorithm identifies areas that require intervention. This initiative contributes to more efficient urban management and an improved quality of life for city residents.

To ensure robust and reliable detection of urban occurrences, the image acquisition rate must remain high. However, this rapid data acquisition strategy results in the accumulation of a substantial amount of redundant visual information. Many of these captured images provide no additional semantic value for occurrence detection, yet these impose a significant burden on the system, by increasing storage requirements, operational costs, and inference time for models deployed on the dataset.

Reducing the volume of redundant images stored on the Theia Vision servers would therefore bring multiple benefits. Such data reduction would not only alleviate storage and computational constraints on the central servers, but more importantly enable faster processing and inference for future deployed models, and simultaneously reduce the overall operational costs of the project.

## 1.2 Problem Identification

In the expected deployment scenario, edge devices continuously capture images at a steady frame rate of five frames per second. With multiple users contributing data, the volume of images be-

ing sent to the central server grows drastically, this leads to significant storage, processing, and network bandwidth challenges.

A major inefficiency arises due to the redundancy of images being collected, as many captured frames contain semantically similar content, as seen in Figures 1.1 and 1.2. This data accumulation not only burdens the central server but also affects edge devices on an individual level, as they must manage and transmit large amounts of redundant data. This leads to increased network traffic, longer upload times, higher latency and overall reducing system efficiency.



Figure 1.1: Consecutive images representing near-duplicates.



Figure 1.2: Consecutive images representing near-duplicates.

Figure 1.3 also showcases an ideal scenario for this task, where the capturing vehicle is immobile due to a red traffic light, and the two images captured in quick succession, exhibit high semantic similarity.



Figure 1.3: Consecutive images representing near-duplicates.

Over the past decades, image matching techniques have steadily evolved, where it is possible to identify and then correspond the same or similar content from two or more different images [7]. However, in recent years, the pace of innovation in this field has increased drastically, with improvements driven by the advancements in artificial intelligence, deep learning, and increasingly sophisticated computer vision algorithms. In particular, the rise of convolutional neural networks (CNNs) has greatly improved feature extraction models, making them more robust and accurate. Additionally, embedding techniques such as feature descriptors and similarity metrics have improved image comparison capabilities, even under varying conditions, such as lighting, perspective and environmental occlusion.

Given these currently known challenges and advancements, the focus of this project is on how to optimize image collection on edge devices to reduce redundant data, while at the same time allowing the server-based model to maintain effective occurrence detection.

This issue is critical for several reasons:

- **Scalability and Performance:** As smart urban monitoring systems expand, the volume of image data will continue to increase, which creates storage and network limitations. Addressing this challenge ensures that the systems remains scalable.
- **Resource Efficiency:** Edge devices have limited storage. Filtering out similar images helps conserve storage of the devices, allowing for more efficient operation.
- **Network and Cloud Optimization:** By minimizing redundant data transfers, network traffic can be alleviated, leading to faster response times and lower operational costs.

## 1.3 Dissertation Structure

In addition to the introduction, this dissertation is organized into six further chapters.

In Chapter 2, the literature review with an overview of existing techniques are presented, as well, the state of the art relevant to near-duplicate image detection.

Chapter 3, elaborates on the problem addressed in this work, including the research questions, provides solution requirements, and the defined problem scope.

Chapter 4 describes the dataset, materials, and methodologies employed, as well as the evaluation procedures used to assess the proposed approaches.

Chapter 5 presents the experimental results obtained for all evaluated similarity techniques.

Chapter 6 discusses these results, highlighting the best performing approaches and providing an analysis of their observed behavior.

Finally, Chapter 7 concludes the dissertation by summarizing the main findings.

# Chapter 2

# State of The Art

## 2.1 Methodology

The methodology used for the literature review was primarily systematic, with the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) methodology serving as the main method to guide the work executed. The process began with the formulation of the first four research queries. After analyzing the results obtained from these queries, a fifth research question was developed later in the project to address gaps in the collected information and ensure a more comprehensive coverage of the topic. The five research questions can be seen bellow:

- **QS1:** ("Image Similarity" OR "Image Matching" OR "Feature Extraction") AND ("Edge" OR "Fog" OR "Fast" OR "Light") AND ("Urban")
- **QS2:** ("zero-shot" OR "few-shot" OR "low-shot") AND ("Edge" OR "Fog")
- **QS3:** ("Embedding" OR "VLM") AND ("Edge" OR "Fog" OR "Fast" OR "Light") AND "Urban"
- **QS4:** ("Reduction" OR "Deduplication") AND ("Edge" OR "Fog") AND "Image" AND ("Urban" OR "City")
- **QS5:** (("Image Comparison" OR "Image Compression") AND ("Urban" OR "City")) OR ("Near-Duplicate")

With the research queries formulated, in order to retrieve the relevant information, four different academic databases were utilized: Scopus [20], ScienceDirect [19], the ACM Digital Library [1], and IEEE Xplore [5]. In all these databases, only articles published after the 1st of January 2021 were considered. Filters were applied to search for the indexed terms exclusively in the title, keywords, and abstract. Additionally, in databases that provided exclusion filters, studies related to medicine and astronomy were excluded.

In total, combining all queries across the select databases, 1760 documents were retrieved. The distribution of results for the different queries and databases can be observed in Table 2.1.

| | QS1 | QS2 | QS3 | QS4 | QS5 | Total |
|---|---|---|---|---|---|---|
| Scopus | 439 | 420 | 56 | 21 | 256 | 1192 |
| Science Direct | 37 | 63 | 87 | 27 | 10 | 224 |
| ACM | 2 | 25 | 2 | 0 | 14 | 43 |
| IEEE | 65 | 163 | 17 | 0 | 56 | 301 |
| Total | 543 | 671 | 162 | 48 | 336 | 1760 |

Of the 1760 retrieved documents, 388 were duplicates, leaving 1372 unique documents. Once the duplicated documents were flagged, the focus shifted to selecting the most relevant documents. To determine the relevance of each document in relation to the dissertation's objectives, the following four screening questions were applied:

1. Is the document written in English?
2. Does the document explicitly takes advantage of image matching or feature matching algorithms?
3. Are the algorithms being tested on edge devices or have as principle edge computing?
4. Is the document using urban images as the main scenario for investigation?

After applying these criteria to the remaining documents, 53 were assessed as relevant. A noticeable difference in results emerged between the documents retrieved through the first four queries and by the fifth, later added query.

The distribution of the relevant documents can be seen in Table 2.1.

| | QS1 | QS2 | QS3 | QS4 | QS5 | Total |
|---|---|---|---|---|---|---|
| Scopus | 9 | 6 | 1 | 1 | 28 | 55 |
| Science Direct | 2 | 1 | 0 | 0 | 0 | 5 |
| ACM | 0 | 0 | 0 | 0 | 0 | 0 |
| IEEE | 0 | 1 | 1 | 0 | 3 | 8 |
| Total | 11 | 8 | 2 | 1 | 31 | 53 |

Although 53 documents were identified as relevant, only a few fully satisfied all four screening questions. Moreover, none of the documents addressed the central research theme of this dissertation. Documents were considered relevant if they met at least one of the screening criteria to a significant extent, particularly if their content contributed meaningfully to understanding the intersection of image matching, or edge computing.

A more detailed analysis and discussion of the retrieved results is presented in the following subchapter.

## 2.2 Literature review results

From the 1372 retrieved documents only 53 were assessed as relevant. In the following sections it will be discussed the main themes found in this step of the literature review.

### 2.2.1 Summary of results

By analyzing the relevant documents, multiple different areas were identified that are closely connected to this problem of near-duplicate image removal. These themes show a great range of approaches and how different approaches that handle near-duplicate images can significantly affect performance, efficiency and the utility of resulted application.

Two primary categories of documents emerged in the relevant part of the literature review. The first category involves algorithms and strategies to solve the near-duplicate problem. Notably, Thyagharajan et al. [22] provide a comprehensive literature review on this topic. Their work highlights how a variety of techniques have been developed to detect and manage near-duplicate images, each tailored to specific goals and application contexts.

One such application is image retrieval from large-scale databases, where the presence of near-duplicate or manipulated copies of images can degrade the quality and reliability of retrieval results. In these systems, it's quite important to identify and remove near duplicates, this improves both accuracy and user experience. Similarly, in information retrieval systems, an ever expanding part of today's society with the presence of Big Data leads to increased storage requirements and higher operational costs. Effective near-duplicate detection mitigates these issues by reducing redundancy and also by optimizing indexing strategies, since it makes possible the clustering of semantic similar images.

Thyagharajan et al. also emphasizes the role of feature extraction techniques in identifying near-duplicate images, even when images have been altered due to variations in viewpoint, lighting conditions, or transformations such as cropping and resizing. This is particularly relevant to burst photography and panoramic stitching, where images are captured in rapid succession with potential camera or subject motion, leading to subtle variations of the captured images and must be flagged as a duplicated image.

An important note and key take away from this literature is that no single algorithm consistently performs well across all scenarios. The effectiveness of the chosen near-duplicate detection method, often depends on the specific requirement of the task. For example, an algorithm designed for high precision retrieval in a robust system, doesn't meet the performance or latency constraints of a real-time mobile application that selects the best frame from a burst sequence.

Another prevalent category within the relevant literature involves the use of pretrained deep learning models and the fine-tuning of existing architectures to perform object identification in images. While models are often employed as a technique to evaluate image relevance, filter redundant content, or even understand scene composition, the main idea is that by identifying the presence and location of objects within images, one can make an informed decision about whether the two images, are meaningfully different or near-duplicates.

The reviewed documents vary a lot in focus, ranging from approaches aimed at identifying a single target object(e.g., a fire [25], or another type of object) to those attempting to detect all objects present in a scene. Most of the models discussed across these studies are variants of already well established architectures, including MobileNet, YOLO (You Only Look Once), and R-CNN

(Region-based Convolutional Neural Network). Each of these models strikes a different balance between speed, accuracy, and computational demand. MobileNet is frequently used in resource constrained environments, such as mobile or embedded devices. In contrast, the main variants of YOLO and R-CNN are designed for more computational capable devices, but there are variants of both these models that were designed for mobile devices, and results of all types of variants were found in the relevant documents.

Although the focus of these documents was not specifically on near-duplicate image detection, the use of object models provides a valuable framework for evaluating image similarity. By analyzing which objects are present, their positions, and their relationship within a scene, it becomes possible to infer whether two images are semantically similar or distinct. For example, if two images contain the same detected objects in nearly identical configurations, they are most likely to be similar and therefore redundant. In contrast, if new objects appear, or the existing ones change position or are interacting with the environment differently, the images should be deemed as unique and worthy of retention.

This type of object level analysis may introduce a valuable additional semantic layer to traditional pixel or feature based similarity metrics. It's a technique worth exploring since it enables systems to move beyond surface level comparisons and consider scene understanding, as part of the decision-making process for duplicate removal. However, when evaluating these models across high computational devices, is still important to take into account, if future deployment of such techniques on edge devices will be feasible on lower performing devices.

## 2.3 Near-Duplicate Image Problem

As the literature review findings suggest, identifying and removing such near-duplicate images is quite crucial across multiple domains, such as:

- Reducing storage requirements and network bandwidth.
- Improving the efficiency of image retrieval systems.
- Optimizing the training of deep learning models.

For example, Li, et al. [9] demonstrated that having these near-duplicate images on training datasets for convolutional neural networks (CNNs), would pose unnecessarily extra resources and computation consumption. By running their deduplication model, CE-Dedup, they were able to reduce the dataset size by 23% while maintain the same validation accuracy. Moreover, if they were to allow a small validation accuracy drop of 5%, their model was able to trim the dataset size by 75%.

Rahman, et al. [16] have similar findings, with image deduplication techniques focused in the context of image retrieval systems. Their method was able to preserve the same validation mean average precision (mAP) and reduce the training dataset by 25.13%, while also being able to lower

the energy consumption of the Deep Neural Network (DNN) training by 27.17%. This leads to more practical and sustainable DNN training practices.

Given the range of possible variations between near-duplicate images, the methodology for the identification of these images must begin with preprocessing the selected images. This step is quite important for normalizing data, and preparing images for accurate similarity assessment. The following section, 2.4.1, reviews common preprocessing techniques used to improve the performance of both traditional and deep learning image similarity methods.

## 2.4 Background

### 2.4.1 Preprocessing techniques

With the collected image data, preprocessing techniques must be applied to transform raw images into a more suitable format for analysis by computer vision models. The rationale behind these techniques is to standardize image properties, reduce noise and artifacts and enhance visual features, therefore making the images more useful and easier to process, whether by traditional algorithms or deep learning models.

Common preprocessing operations include image resizing, grayscale conversion, noise reduction, normalization, and contrast enhancement. Selecting which techniques to use mostly depends on the type of model being used.

When employing more traditional feature extraction methods, such as, like Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Feature (SURF) and Orientated FAST and Robust BRIEF (ORB), the pre-processing pipeline tends to be relative simple. Standardization is often achieved by simply resizing and applying a noise reduction filters, such as, median filter or Gaussian blur. For example, Sundaram, et al. [21], goes on a detailed explanation on how they used the median filter as a noise reduction strategy, being an effective strategy for preserving edge information while eliminating small scale noise.

In contrast, when using deep learning models, more complex and strict preprocessing techniques needed to be explored. Mohiuddin et al. [14], for example, utilized multiple techniques when using MobileNetV3. Their methodology included histogram equalization for contrast enhancement, resizing all input images to the model's required dimensions, that in this case is 224×224 pixels, and normalizing pixel intensity values to the [0,1] range.

Similarly, Li, et al. [9], performed additional preprocessing steps, converting RGB images to grayscale, moreover, in this case the documents states a down-sampling to an extremely low resolution, being 8x8 pixels, that is a common target for most hashing algorithms. In their preprocessing stage, they also employed data augmentation to artificially expand the training dataset, introducing pixel shifts, intensity variations and rotations to simulate near-duplicates and evaluate their model's capabilities more precisely.

These examples underscore the importance of customizing the pre-processing methodology to the target algorithm. As the next sections will explore both traditional similarity metrics and

the performance of deep neural networks, it's consistently important to note the preprocessing techniques as the backbone for these algorithms.

### 2.4.2 Traditional Feature Extraction Methods

In the reviewed literature, a consistent theme is the wide spectrum of model complexity. That can range from traditional, handcrafted feature extraction to techniques, to more sophisticated deep learning based models, or even a combination of both strategies [6].

As for the feature extraction methods, those can be categorized intro three different types: keypoint feature extraction, pixel based feature extraction, and area based feature extraction. Keypoint extraction focuses on identifying local features that are invariant to transformations, while pixel based and area based methods extract global or regional image information.

The three following sections (2.4.2.1, 2.4.2.2 and 2.4.2.3), describe these methods in detail work.

#### 2.4.2.1 Keypoint Feature Extraction

Keypoint feature extraction is a low-level method that focuses on detecting and describing salient points or regions in an image. These points are typically chosen for their invariance to changes in scale, rotation, lighting, which is a good strategy for identifying near-duplicate images that were captured under similar conditions and close in temporal distance.

**Scale-Invariant Feature Transform (SIFT)**: Created in 2004, by Lowe [12]. This technique solves the scale variance problem, however it's expensive to compute, since it takes advantage of the Difference of Guassian (DoG) to identify local extrema. For each detected keypoint, it computes a feature descriptor based on the orientation and magnitude of image gradients in the local neighborhood and creates an orientation histogram. This is an iterative process that is calculated for each cell, therefore allowing it to be, robust to detecting affine transformations, perspective changes and additive noise [22].

**Speeded-Up Robust Features (SURF)**: Described by Bay et al. [2], is also scale and rotation invariant detector. This technique uses box filters, and it can also be used for classification tasks while being faster to compute than SIFT. This is due to the fact that SURF integrates the gradient information within a sub patch, whereas SIFT depends on the orientation of the individual gradients [22]. The time execution advantage usually comes at the cost of accuracy from the model.

**Orientated FAST and Robust BRIEF (ORB)**: A method that combines two different algorithms, FAST and BRIEF, was created by Rublee et al. [18]. They demonstrated the ORB is able to outperform SIFT by at least two times while maintaining good accuracy across many situations. This feature descriptor uses simple binary tests between pixels in a smoothed image path. The main drawback is that scale invariance is not addressed and is sensitive to in-planar rotations [22].

With the keypoints detected and described, matching is performed using distance metrics, such as Euclidean or Hamming distance, between descriptors from different images. Having a high

number of good matches between the keypoints of images will often indicate that exists strong similarity, this metric can be used as a technique to identifying near-duplicates.

All these methods are quite useful, especially when working with datasets that are consistent with object presence and structure across different images, with only minor spatial shifts or illumination differences.

#### 2.4.2.2 Pixel based Feature Extraction

Pixel based feature extraction methods, analyze pixel values on an image directly, without relying on specific keypoints. These techniques typically involve comparing pixel intensity values, color histograms, gradients, or structural similarity across the entire image, or sample grids.

As Thyagharajan et al. [22] state, unlike keypoint based methods, pixel based methods consider global image properties, so with these techniques it's easier to detect near-duplicates images from real duplicates. Although in the context of this dissertation, both near-duplicates images and real duplicated images should be deleted, so having a more strict method could turn out to be a disadvantage, the results are still worth exploring and as such it is relevant to state the main known techniques, such as:

**Color Histogram**: A simple method to represent images by the distribution of color or intensity values, can be then compared by simple correlation metrics.

**Mean Squared Error (MSE)** and **Peak Signal-to-Noise Ratio (PSNR)**: Similar, to color histograms, measures the pixel-wise similarity between two images but are sensitive to transformations.

#### 2.4.2.3 Area based Feature Extraction

Area based methods, compare subregions or patches between two images. These approaches are well suited for detecting near-duplicates with partial similarity or slight motion. Consequently, these approaches excel on retrieval systems, where the goal is to rank images based on content relevance. However, area based strategies can struggle in scenarios involving camera movement, or viewpoint changes, since the spatial correspondence may be difficult to preserve.

As Thyagharajan et al. [22] describes, the main techniques to perform area based feature extraction are, Weighted visual words, Contrast context histogram, and Pulse Coupled Neural Network (PCNN).

### 2.4.3 Hashing Techniques for Image Similarity Identification

#### 2.4.3.1 Traditional Hashing

Hashing techniques such as Average Hash (aHash), Difference Hash (dHash), Wavelet Hash (wHash), and Perceptual Hash (pHash) are able to compact and fixed size binary strings that approximate the visual structure of an image [23]. These methods are computationally efficient and robust to minor transformations such as resizing, compression, and slight color changes.

- **aHash:** Computes the average pixel intensity and assigns 1 or 0 to each pixel based on whether it is above or below the average.
- **dHash:** Compares adjacent pixels and encodes whether each pixel is lighter or darker than its neighbor.
- **wHash:** Utilizes wavelet transforms to extract frequency features at multiple scales.
- **pHash:** Utilizes discrete cosine transform (DCT) to capture more robust perceptual features of the image.

While these methods are quite useful for fast filtering, they are often limited in their ability to handle complex transformations, or fine-grained changes.

## 2.5 Deep Learning Approaches

In recent years, deep learning approaches have gained significant popularity and demonstrated outstanding performance in a wide range of computer vision tasks, in some scenarios being able to outperform even the human vision system. This rise in popularity is largely attributed to three key factors:

- The substantial increase in the availability of large-scale annotated datasets.
- The improvements in computational resources, particularly through GPUs.
- The continuous development of more efficient neural network architectures and better training strategies.

These advancements enabled researchers to design deep learning models that excel in object detection, with high accuracy and scalability. Notable architectures such as YOLO (You Only Look Once), and MobileNet have become foundational in modern object detection. They are capable of learning rich hierarchical features from data, making them a good option for identifying differences between similar images.

Before discussing the state-of-the-art models in detail, it is essential to first understand how Convolutional Neural Network (CNN) works, the technology behind most of these models. CNNs are at the core of many deep learning applications in computer vision, due to their ability to automatically learn spatial hierarchies of features through convolutional operations. The following subsection provides an overview of CNNs.

### 2.5.1 What is a Convolutional Neural Network?

Convolutional Neural Network (CNN) is a class of deep neural networks specifically designed to process data organized in grids, such as images. Contrary to other traditional fully connected neural networks, CNNs take advantage if the spatial structure of images by employing convolutional layers to extract features at different levels of data [23].

CNN architectures are composed of several key components:

- **Convolutional Layers:** The fundamental building block of a CNN is the convolutional layer, which applies a set of kernels that slide over the input images, to detect local features such as edges
- **Activation Functions:** Transformations applied to the output of convolutional layers, which enables the model to capture patterns. Initially Sigmoid function was used but recently, ReLU has been demonstrating better performance.
- **Pooling Layers:** Reduction of spatial dimensions from feature maps, which decreases computational cost and prevents overfitting.
- **Fully Connected Layers:** At the end of the network, where classification is performed on the extracted features.

CNNs came with a major advantage, that is their ability to learn relevant features from raw pixel data, without the need for manual feature engineering, which makes them very useful for tasks such as image classification, and object detection.

The following section explore popular CNN architectures that build on this foundational structure.

### 2.5.2 CNN-Based Similarity Models

#### 2.5.2.1 Siamese and Triplet Networks

Described by Yash Jakhar et al. [6], it's an architecture, where two similar subnetworks are interconnected at the exit level. With the subnetworks being "Siamese twins" and having the same set of operations and weights, the entire construction is called a Siamese network. The primary purpose is to learn a metric space in which the distance between two points is a direct measure of the similarity of the corresponding images. This excels in the task of finding near-duplicate images, since a Siamese network is forced to project the images onto a high-dimensional space so that attracting pictures occupy nearby points, and non attracting pictures are far from each other. This process is able to be achieved with a contrast loss function or a triplet loss function.

Yash Jakhar et al. [6], implemented their Siamese Network architecture by taking advantage of two different strategies, perceptual hashing (pHash) and vision transformers (ViT). Their model combines the two feature vectors generated, and then uses both ViT and pHash to predict image similarity. The primary motivation behind this dual approach is that pHash, the less computationally intensive operation, quickly filters out images that are clearly dissimilar, which reduces the number of comparisons that need to go through the more computationally intensive operation, done by the ViT. The outputs from both methods are concatenated into a single feature vector, this vector is then passed through a fully connected network to produce a similarity score between 0 and 1. In order to validate the model, the authors, during preprocessing, performed a set of data

augmentation to simulate real-world variations, such as changes in orientation, scale, brightness, and noise. Model performance was evaluated on two benchmark datasets: *California-ND* and *INRIA Holidays*. On *California-ND*, the model achieved a precision of 0.987 and recall of 0.946, while on *INRIA Holidays*, it attained a precision of 0.884 and recall of 0.878.

#### 2.5.2.2 CNN-Based Detection Models

Object detection is one of the core tasks in computer vision, and several CNN models have been developed to identify and localize objects in images. These models, vary in architecture, detection speed, accuracy, and computational complexity. In the context of near-duplicate detection, object detection via CNN models can be used to assess the semantic similarity between images based on object features such as their presence and absence, or their positional changes.

This section discusses two influential models, with high speed capabilities: You Only Look Once (YOLO) and MobileNet. These have distinct advantages depending on the application context, and how they are implemented.

##### You Only Look Once (YOLO)

YOLO is a family of real-time object detection systems introduced by Redmon et al. [17]. The various YOLO versions differ in terms of speed, architectural complexity, and computational cost. YOLO formulates object detection as a single regression problem. It divides the input image into a grid, and for each cell, predicts a fixed number of bounding boxes along with confidence scores and the class probabilities.

Some key characteristics of YOLO include:

- **One-shot detection:** As the name implies, YOLO predicts all bounding boxes and class probabilities in one pass through the network.
- **Speed oriented:** YOLO is designed for real-time applications, which offers fast object inference on limited hardware, such as mobile devices.
- **Trade-off:** YOLO early versions prioritized speed but struggled with detecting small objects or densely packed scenes. However later versions, have improved accuracy substantially while maintaining efficiency.

Using YOLO then proves to be a relevant strategy to detect near-duplicates images. This model can be used to identify and localize objects in images with small variations. This can be done by comparing, the number, type, and positions of the detected objects across the image frame. If two images, in the same temporal space, contain the same objects in similar arrangements, they can be considered near-duplicates and filtered out.

However, as stated in section 2.2.1, the reviewed literature primarily employed YOLO as a model to detect a single class of objects, in order to complete a certain task, rather than a method

to determine the total number of objects and their respective position in the scene, as a method to determine if two images are near-duplicates. Despite this, these studies are still relevant, as they demonstrate the effectiveness of YOLO variants in object detection under various constraints.

Liu, Yuhang, et al. [11] proposed a lightweight YOLO-based architecture named PV-YOLO, designed for pedestrian and vehicle detection. Their models builds on YOLOv8n, replacing the convolution layers in the backbone with receptive-field attention convolution (RFAConv) to enhance feature extraction capabilities. Additionally, they incorporated a bidirectional feature pyramid network (BiFPN) neck to improve multiscale feature fusion and introduced a P2 detection layer to improve detection of small, distant targets. PV-YOLO was evaluated on two benchmark datasets, and demonstrated superior performance compared to YOLOv8n and other baseline models, achieving higher detection accuracy while maintaining a low computational consumption.

Linsong Xiao et al. [25], also proposed a lightweight YOLO-based architecture to accurately detect fire detection on resource-constrained embedded devices. They propose a lightweight object detection model called EMG-YOLO, designed to improve detection performance while minimizing computational overhead. To achieve this, they introduce a Multi-scale Attention Module (MAM) that integrates multi-scale information to enhance feature representation. They also develop an Efficient Multi-scale Convolution Module (EMCM), embedded within the C2f structure, to improve the extraction of flame and smoke features without increasing computational complexity. Experimental results show that EMG-YOLO improves detection performance while maintaining efficiency, and reducing the number of parameters, when compared to YOLOv8n.

#### MobileNet

MobileNet [4] is also a family of lightweight CNN models designed for efficiency on mobile and embedded devices. These models achieve competitive performance with a reduced number of parameters and lower computational cost, by using depthwise separable convolutions.

Some key characteristics of MobileNet include:

- **Lightweight and fast:** Is suitable for edge deployment, where computational resources are limited.
- **Customizable:** Can be easily customized with multipliers and input resolution to balance speed and accuracy.
- **Variants:** Like YOLO, early versions of MobileNet prioritized speed, sometimes struggling with detecting small objects. Later versions, also improved model accuracy while maintaining efficiency.

In the context of near-duplicate image detection, MobileNet can be utilized in a manner similar to YOLO, using object detection to compare the number, type, and positions of objects across image pairs. Additionally, it can be used as a feature extractor to generate image embeddings, that can be compared with similarity metrics to identify near-duplicates.

A relevant implementation of MobileNet for filtering near-duplicate images is presented by Mohiuddin et al. [14]. This model leverages a more recent MobileNet implementation, MobileNetV3, as a lightweight feature extractor to generate high-dimensional embeddings from images. The extracted embeddings are used to compute cosine similarity scores between image pairs, facilitating near-duplicate image removal.

The model was evaluated on multiple datasets, including *INRIA Copydays*, *QUALINET*, and *CIFAR-10*. It achieved high performance with an accuracy of 95.5%, precision of 93.3%, recall of 98.2%, and an F1-score of 95.6%.

### 2.5.3 Transformer-Based Vision Models

A Vision transformer as described by Yash Jakhar et al. [6], is a method of computer vision that adapts the transformer architecture, originally developed for natural language processing, to image analysis. Instead of processing an image as a grid of pixels, this neural architecture treats images as a sequence of patches, similar to how the transformer models are utilized in natural language processing, and is great technique to capture global relativity in an image.

Vision transformers (ViTs) stand in contrast to Convolutional Neural Networks, which operate on local receptive fields. While CNNs are highly effective at learning local patterns, they can struggle to differentiate images with subtle global variations. In such scenarios, ViTs offer a significant advantage due to their ability to directly model the relational structure between distant parts of an image. Recently, ViTs have shown increasing performance in tasks involving image similarity, where subtle differences must be detected. With some important models to highlight such as, Florence-2 developed by Bin Xiao et al. [24], and Grounding DINO developed by Shilong Liu et al. [10].

# Chapter 3

# Problem Statement

This chapter expands on the problem introduced in Section 1.2, creates a hypothesis for the problem, and outlines the proposed solutions along a detailed plan that addresses the multiple aspects of the challenge. Section 3.1 details the specific problem targeted in this study. Section 3.2 constructs a hypothesis for the previously identified problem. Next, in Section 3.3, potential answers to the research questions are explored. Section 3.4 discusses the key requirements a solution must meet to effectively address the problem defined earlier. Finally, Section 3.5 defines the scope of this work, highlighting the primary focus areas of the study.

## 3.1 Problem

In the current era of exponential digital data growth, image detection systems are increasingly burdened by the volume of visual information they must process. A significant portion of the captured data contains images with highly similar or even identical semantic content, resulting in substantial redundancy. This data is collected from a wide range of sources, including dashboard cameras, surveillance systems, drones, and other devices, all of which operate at high capture rates.

Such redundancy negatively impacts downstream processing pipelines, particularly models that rely on generative artificial intelligence techniques. These models are computationally intensive and demand significant server resources. Consequently, processing redundant images leads to unnecessary energy consumption, increased inference time, and higher operational costs.

To mitigate these issues, it is essential to develop efficient and lightweight methods capable of identifying and removing images that do not contribute with unique semantic information. An effective near-duplicate detection strategy can significantly reduce dataset size while preserving the information required for reliable occurrence detection, improving the overall system efficiency.

This is a challenge currently faced by the *Theia Vision* team: how to effectively detect and remove semantically similar images, near-duplicates, without compromising the accuracy and reliability of occurrence detection. The goal is to evaluate strategies capable of reducing redundant

visual data, prior to sending it to a centralized server, where the main model processes the detection of occurrences.

To address this challenge, it is necessary to develop and evaluate strategies, that are able to assess the near-duplicate status between two different images. The developed strategies should be robust to marginal cases, where near identical images with different semantic meanings can be non near-duplicate images, while also being able to classify images with some more accentuated visual differences but identical semantic content as near-duplicates. As such, the methods must reliably identify sequences of consecutive images that have the same semantic information and classify them as near-duplicates, allowing unnecessary data to be filtered during urban image monitoring. Crucially, this process must also take into account the performance of the occurrence models, that will be deployed after in the already reduced dataset, which necessitates thorough integration and multiple different testing sessions with that system, to determine the impact of data reduction and to ensure that no important visual information is being removed.

An additional layer of complexity arises from the need to deploy this solution on edge devices powered by a low amount of energy. Therefore, when evaluating the techniques developed, a huge factor to take into consideration is if the techniques are lightweight and energy efficient, avoiding significant battery drain or device overheating while maintaining high performance in semantic similar detection under limited computational resources. To achieve this, multiple methods will be evaluated, and the trade-offs between detection accuracy and processing time will be analyzed and documented.

## 3.2 Hypothesis

As mentioned in previous sections, the occurrence detection model is deployed on the central server after the collected image data has been transferred to it. These events of interest can take multiple forms, being events that need human action, such as garbage in the wrong places or full garbage bins, obstructed signals, abandoned buildings, dead animals or even invasive species that should not be present in the monitored area.

Considering the problem identified and the definition of occurrence, the following hypothesis was formulated:

The size of data collected by edge devices can be reduced by at least a factor of four through near-duplicate identification, without negatively affecting the accuracy of occurrence detection performed by the server-based model.

## 3.3 Research Questions

The problem identified in the previous section can be divided into the three following research questions:

- **RQ1 - What are the most effective feature extraction and embedding techniques for identifying and discarding semantically similar images on low-power edge devices?** To enable efficient near-duplicate detection, it is essential to evaluate lightweight yet effective techniques for visual extraction. These methods must strike a balance between semantic classification capabilities and computational simplicity.
- **RQ2 - What trade-offs are necessary in order to achieve high semantic similarity detection, with the computational constraints of edge devices?** Identify trade-offs, such as reduced model size with accuracy loss, or inference speed with consumption, is critical for deploying models that are both effective and sustainable in edge scenarios.
- **RQ3 - How does reducing redundant image transmission impact the overall storage, and occurrence detection in smart city applications?** Investigate the implications of near-duplicates removal. By reducing the number of redundant images, what is the impact in the storage resources and in the accuracy of the occurrence detection.

## 3.4 Solution Requirements

The solution needs to fulfil the following requirements to address the problem described in Section 3.1 and to answer the research questions, revisited in the previous section:

- **R1: Implement lightweight, edge compatible feature extraction techniques.** The solution must integrate feature extraction methods that are computationally efficient for low-power edge devices. These techniques should present a robust classification performance with small model size and inference time.
- **R2: Identify and remove semantically similar images.** A core requirement for the solution is to accurately detect temporally close images with similar content and remove the ones deemed as redundant. This image removal model should operate after the device completes a collection cycle and before the images are uploaded to main central server.
- **R3: Ensure compatibility and occurrence preservation with subsequent models.** The main objective of the project is to optimize the occurrence detection model. As such, when performing near-duplicate removal is important to ensure preservation of essential data for the subsequent interaction. Integration testing with occurrence detection is required to ensure that no visual information is lost.
- **R4: Optimize for battery efficiency and thermal constraints on edge devices.** As the solution will be deployed in user edge devices, it must have minimal power consumption and prevent overheating during continuous operations.
- **R5: Evaluate and tune similarity thresholds to control redundancy and information retention.** Experiments must be conducted to evaluate how different threshold levels affect redundancy reduction and potential information loss.

- **R6: Provide logging and benchmarking tools to evaluate system performance.** To support model experimentation and validation, the solution should include tools for logging filtered images, resource usage, and evaluation metrics.

By incorporating these considerations into the solution design and implementation it is possible to develop and evaluate techniques that can provide a semantic similar image detection, capable on edge devices, that doesn't affect the occurrence object detection model.

## 3.5 Scope

As stated in Section 3.4, this work focuses on the development and evaluation of lightweight, edge compatible strategies for detecting and removing semantically similar images in urban monitoring systems. The main goal is to evaluate how unnecessary data can be reduced and how that reduction affects transmission and storage, without heavily impacting the accuracy of the occurrence detection tasks that follow.

The study includes combining and evaluating computer vision techniques in order to create a robust iteration that is capable of working in multiple different scenarios.

## 3.6 Methodology Plan

With the solution requirements defined in Section 3.4, and the scope outlined in Section 3.5, the following methodology plan has been established to guide the implementation of the experimental and quantitative solutions:

- **SP1: Creating a sample dataset.** The first step focuses on building a representative dataset from the already collected images in urban monitoring systems. This includes organizing, labelling, and applying data augmentation techniques to ensure the dataset reflects real world variance and includes both semantically similar and distinct image pairs.

- **SP2: Evaluating traditional image similarity techniques.** The second step focuses on testing different traditional methods. The goal is to evaluate how well these techniques perform in identifying semantically similar images. All experiments and performance metrics will be documented for comparison.

- **SP3: Evaluating semantic-based similarity extraction.** This step focuses on testing deep learning techniques for semantic similarity detection. Feature extraction using pretrained CNNS will be applied, with embeddings compared using metrics such as cosine similarity. Additionally, object detection methods will be employed to generate positional semantic maps that identify and localize objects within each image. These maps will help in construct a higher level semantic representation of the scene, that can be used to compare with other images and assess if they have the same semantic meaning or not.

- **SP4: Combining traditional and semantic techniques.** Based on findings from the two previous steps, a hybrid approach will be explored. This step aims to combine the speed of traditional techniques with the semantic understanding offered by deep learning models. This method aims to quickly discard clearly redundant images using lightweight heuristics, while applying more computationally intensive semantic analysis only in ambiguous or complex cases.
- **SP5: Deployment on edge devices, with testing.** The select models will be deployed and evaluated on edge devices. Performance metrics will be collected and analyzed. Fine-tuning will be conducted to ensure that the model meets the constraints of the target edge devices, achieving a balance between semantic similarity detection accuracy and computational efficiency.
- **SP6: Writing dissertation.** The final phase involves consolidating all work, results, and insights into a structured thesis. This will include methodology, experimental results, discussion and conclusions.

By following this development plan, it is expected to find and deploy strategies that can detect images with similar semantic content, remove semantic similar content and evaluate how that reduction affects the occurrence object detection model.

# Chapter 4

# Materials and Methods

This chapter outlines all materials used to complete this study and their implementation.

## 4.1 Dataset Preparation

Over the past two years, urban image data was collected by volunteer users, resulting in a dataset of approximately 340 000 images acquired across two distinct data collection APIs. Using the new API methodology, a total of 72 703 images were collected, while the old API methodology yielded 281 635 images.

During the initial quality assessment of the dataset collected with the new API, a critical error was identified. In continuous capture sessions, multiple consecutive images were found to be pixel-wise identical despite having different timestamps, followed by an abrupt temporal jump once the error ceased. This behavior is illustrated in Figure 4.1, where timestamps of the captured images are represented as integer values. Across the new API dataset, this issue results in more than 20 000 duplicated images.

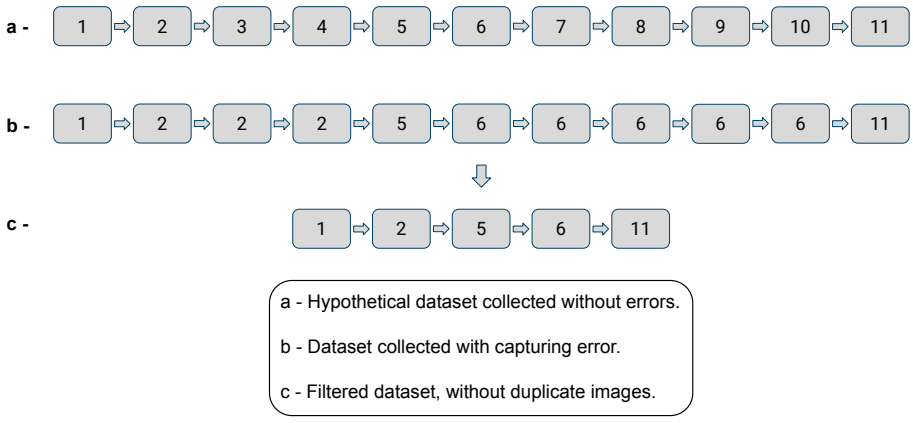



Figure 4.1: Schematic representing an error found in the new API dataset

The presence of these exact duplicates breaks the temporal continuity that is fundamental to the objectives of this study. As a consequence, all images collected with the API methodology

were discarded, and the remainder of this study relies exclusively on image data collected using the old API.

From the old API dataset, a smaller sample of 80 000 images was selected for this study. This subset was chosen based on the availability of mandatory metadata, namely timestamp and GPS coordinates corresponding to the exact moment each image was captured. The images reflect typical driving behavior, including scenarios with rapid vehicle motion, where the semantic content between consecutive images may vary significantly, but also stationary scenarios, where consecutive images often exhibit highly similar semantic content.

### 4.1.1 Occurrence Detection

Once the subset was selected, the next step consisted in identifying and annotating urban occurrences. Although this task is intended to be performed by an automated inference model deployed on the main server, the model was still under development at the time of this study.

Consequently, all inference was performed manually, using a developed annotation tool. An example of its utilization for the annotation of three different urban occurrences can be seen in Figure 4.2. This tool allowed to create a bounding box around the detected occurrences, and assign each occurrence to a predefined class. A total of 21 occurrence classes were considered:

- Improper waste disposal;
- Abandoned large appliance;
- Abandoned furniture;
- Blocked road sign;
- Decaying infrastructure;
- Graffiti;
- Pest;
- Stray animal;
- Dead animal;
- Urban wildlife;
- Unchecked vegetation growth;
- Public safety incident;
- Parking infraction;
- Streetlight malfunction;
- Obstruction on sidewalk or pathway;

- Vomit or other liquid waste;
- Drain water accumulation;
- Construction site;
- Micromobility vehicle;
- Advertisement;
- Public art or monument.




Figure 4.2: Figure with examples of the annotation tool.

Following the annotation of all 80 000 images, a total of 4 937 images were identified as containing at least one occurrence. A second annotation pass was then conducted exclusively on these images. During this stage, consecutive images depicting the same occurrences were grouped into clusters, referred to as *bags*. Upon completion of this process, a total of 328 distinct urban occurrences were identified.

In Figures 4.3 and 4.4, two distinct urban occurrences bags, representing a blocked road sign and improper waste disposal occurrences, respectively, are depicted with all their consecutive images, showcasing how the presence of the urban occurrences can evolve throughout an image capture session.



Figure 4.3: Consecutive images representing a distinct *blocked road sign* bag.



Figure 4.4: Consecutive images representing a distinct *improper waster disposal* bag.

### 4.1.2 Occurrence Characteristics

As described in the previous section, the annotated subset of 80 000 images contains a total of 328 distinct urban occurrences. These occurrences exhibit highly distinct characteristics. Between different images in singular occurrence bags there's a high spatial inconsistency of the detected class occurrence within the image, with additional high temporal and geospatial displacement. It's also important to note, that some occurrences are represented by a single image, while others spun up to 300 consecutive images.

This variability is illustrated in Figure 4.5, which presents both a bar chart showing the number of images per occurrence bag and a box plot summarizing the overall distribution.

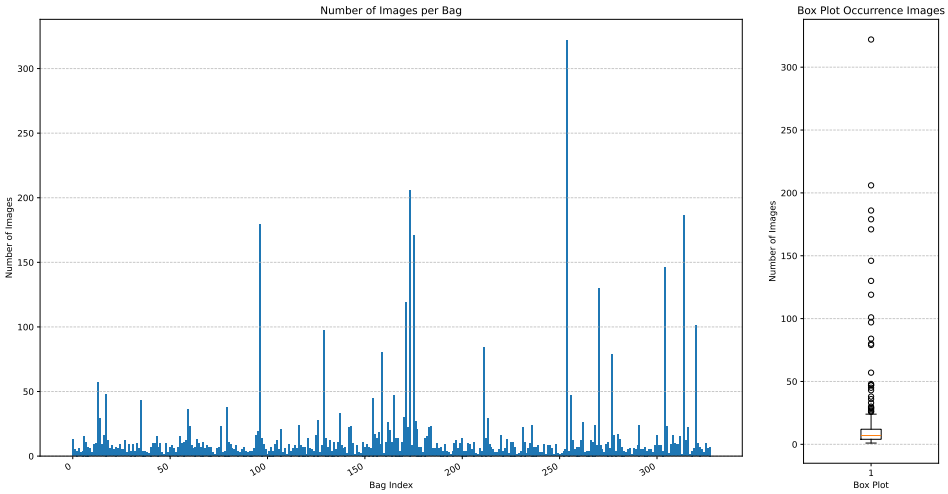



Figure 4.5: Number of images per occurrence bag and box plot of the distribution.

Although 328 distinct urban occurrences were detected, not every class from previous section stated list was found. Out of the 21 occurrence classes defined during the annotation process, only 11 distinct classes were observed in the dataset. Each detected class exhibits a different frequency of distinct occurrences. Figure 4.6 presents the frequency distribution of the detected urban occurrences. Expected patterns from an urban environment can be observed, with *advertisement* being the most frequent urban occurrence, followed by *micromobility vehicle*, *decaying infrastructure*, *improper waste disposal*, and *parking infraction*.

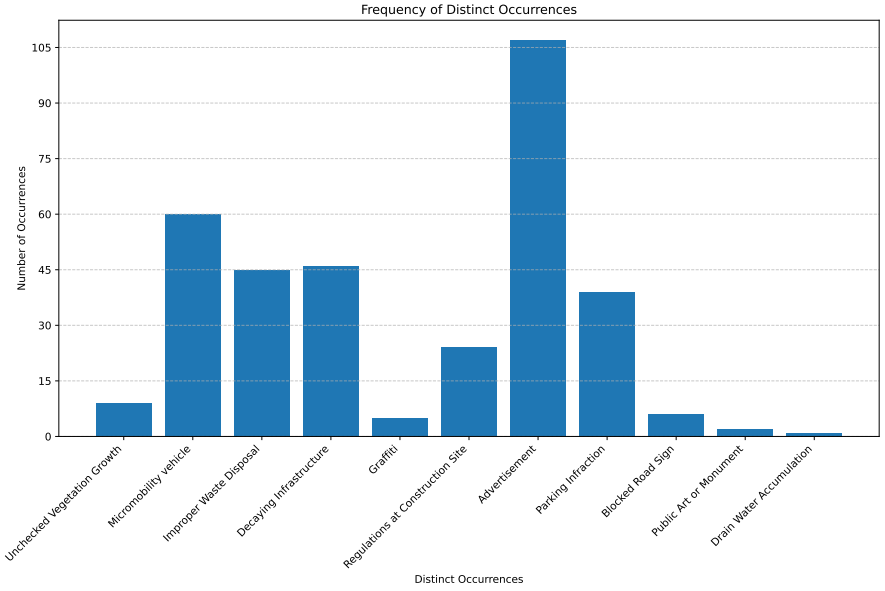



Figure 4.6: Frequency of distinct urban occurrence classes.

Table 4.1 and Figure 4.7 both summarize the variation in distance, time, and bounding box position between the first and last image of each occurrence bag. In scenarios where there is only one image per occurrence bag, the variation will always be zero. Distance variation is computed using GPS coordinates, time variation is derived from image timestamps, and bounding box variation is measured as the Euclidean distance (in pixels) between the centers of the bounding boxes in the first and last frames of each occurrence.

Table 4.1: Variation metrics across distinct urban occurrences bags.

| Metric | Min | Max | Mean | Median |
|---|---|---|---|---|
| Distance variation (m) | 0.000 | 165.122 | 9.643 | 5.007 |
| Time variation (s) | 0.000 | 869.955 | 8.092 | 1.657 |
| Bounding box variation (px) | 0.000 | 956.747 | 251.904 | 956.747 |

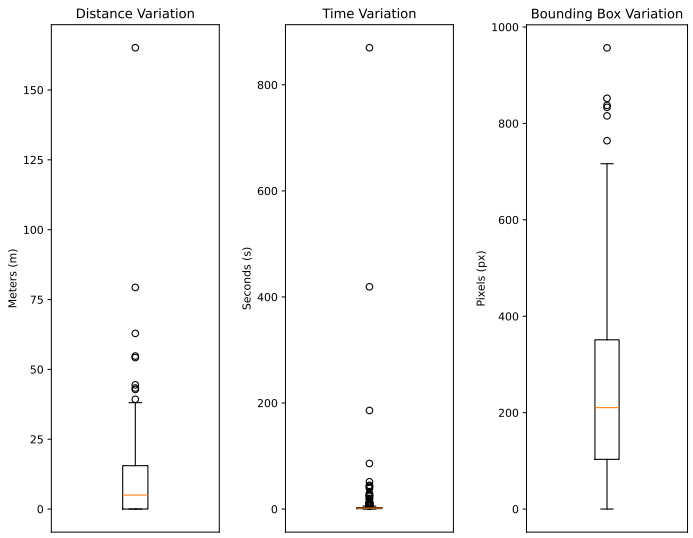



Figure 4.7: Figure with variation metrics across distinct urban occurrences bags.

It is important to note that not all urban occurrences that exhibit zero geospatial variation have that value as an actual characteristic. This behavior is attributed to a metadata error in the dataset that was identified only after completing the dataset preparation phase. In certain capture sessions, the edge device intermittently failed to update GPS coordinates, resulting in multiple consecutive images being assigned equal location values.

In order to provide additional spatial context across the map, the GPS coordinates of images both with and without detected urban occurrences were plotted on a map using different color codings, as shown in Figure 4.8. To contextualize with geographical reference, the complete path is overlaid on a map of Madeira, where all image data was collected. The trajectory spans multiple road environments, including highways and urban roads, with a higher concentration of detected occurrences observed in the Funchal area.

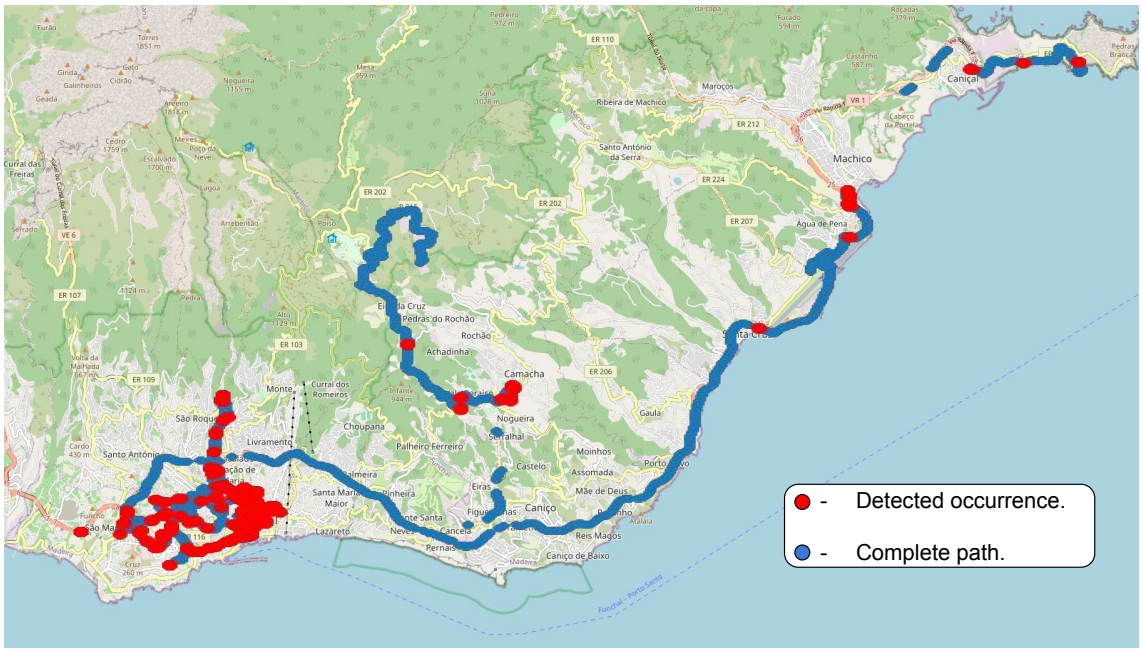



Figure 4.8: Traversal Path with Occurrences.

## 4.2 Near-Duplicate Image Removal Algorithms

As stated the previous section, both traditional computer vision techniques and deep learning methods were evaluated to analyze their effectiveness in detecting near-duplicate images from the dataset and removing them.

### 4.2.1 Traditional Similarity Algorithms

For the traditional techniques, three different traditional algorithms were explored: Perceptual Hashing (pHash), Structural Similarity Index Measure (SSIM), and an approach using Scale-Invariant Feature Transform (SIFT) keypoints. All these algorithms were evaluated under multiple threshold configurations in order to analyze how varying levels of strictness affect near-duplicate detection performance. An explanation of how these algorithms were applied to detect near-duplicate images is provided in the following sections.

#### 4.2.1.1 Perceptual Hashing (pHash)

Perceptual Hashing is an image similarity technique designed to produce compact representations that capture the global visual structure of an image. When images are similar, perceptual hashes generate similar outputs.

In this study, pHash was applied in the grayscale version of the images. Subsequently, the discrete cosine transform was computed to extract low-frequency components. These components were then binarized to form an array hash representation. Similarity between two images was

computed by using the Hamming distance for their respective hashes. If the Hamming distance was below a predefined threshold, the images were classified as near-duplicates.

#### 4.2.1.2 Structural Similarity Index Measure (SSIM)

The Structural Similarity Index Measure is an image similarity metric that evaluates perceptual similarity by comparing local patterns of pixel intensities. SSIM decomposes image similarity into luminance, contrast, and structural consistency. By taking these three components into account, this technique is able to produce a similarity between 0 and 1.

In this work, SSIM was computed between pairs of grayscale images after normalizing the images resolutions. If the resulting similarity score exceeded a predefined threshold, the images were classified as near-duplicates.

#### 4.2.1.3 Scale-Invariant Feature Transform (SIFT) keypoints Mask

The SIFT keypoint Mask approach is the most complex of the traditional similarity algorithms. Rather than directly simply comparing descriptors or match counts, this method estimates visual dissimilarity by leveraging both matched and unmatched keypoints in order to create a similarity mask for each image.

The algorithm consists of the four following steps, with the ordered Figures 4.9, 4.10, 4.11 and 4.12 illustrating the implementation the representation of each step.

1. SIFT keypoints are extracted from each grayscale image.




Figure 4.9: SIFT keypoint mask - Determining keypoints.

2. For each image a black canvas is created. The canvas has the same resolution as the input image. All respective keypoints from each image are projected onto their canvas as white circles.

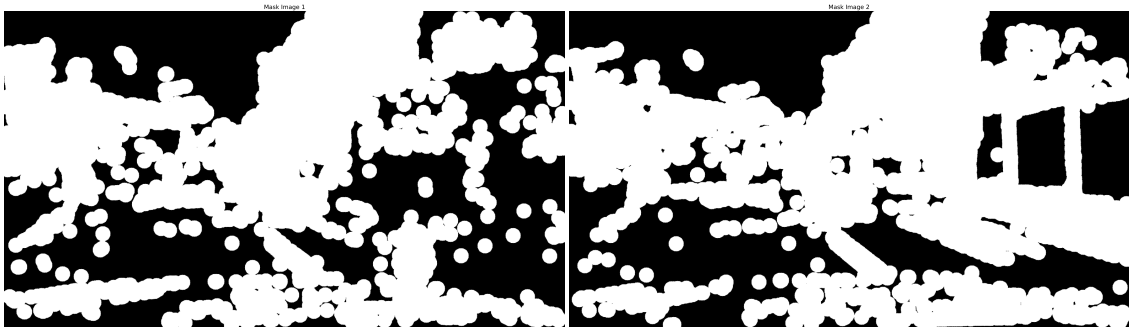



Figure 4.10: SIFT keypoint mask - Drawing keypoints canvas.

3. A brute-force keypoint matcher is applied to identify corresponding keypoints between the two images. Matched keypoints are then drawn as black circles on top of each image canvas.

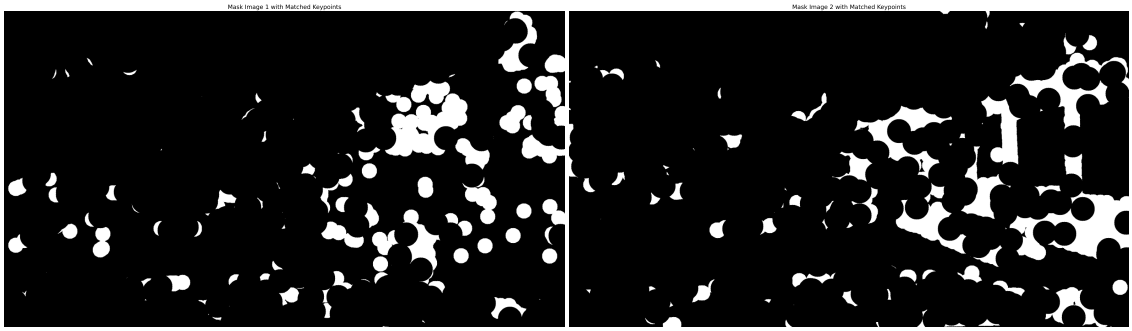



Figure 4.11: SIFT keypoint mask - Drawing matched keypoints canvas.

4. The remaining white regions in the canvas represent unmatched keypoints. A contour detection algorithm is applied to estimate the total area occupied by these regions.

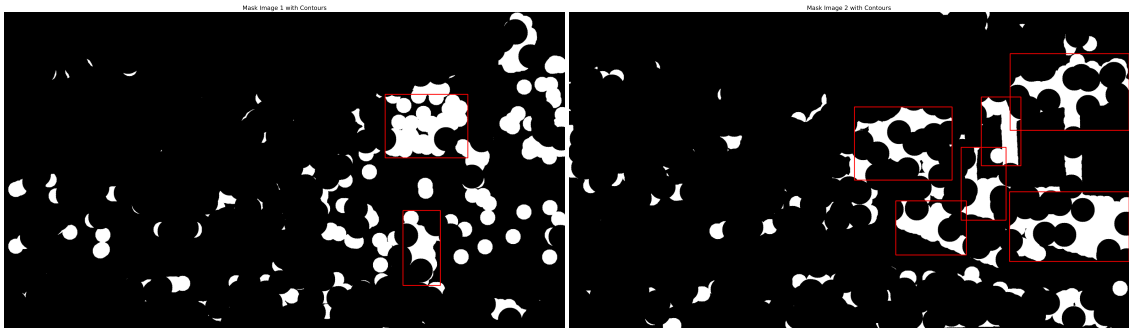



Figure 4.12: SIFT keypoint mask - Determining contours.

The final number of contour areas and their sizes serve as a dissimilarity score between the two images. If the total number of contours and their respective areas exceed the predefined thresholds, the images are classified as non near-duplicates images. Otherwise, they are considered near-duplicates.

### 4.2.2 Deep Learning Similarity Algorithms

Two main deep learning similarity strategies were evaluated in this study. The first strategy leverages global image embeddings extracted using CLIP [15] (Contrastive Language-Image Pre-training), where the similarity between two different images is calculated by computing the cosine similarity of their respective embeddings results. The second strategy is object centric and consisted on attempting to detect all objects present in an image, creating a visual footprint from the detected classes and their spatial locations, and then comparing these footprints across images. For this strategy both vision transformer and convolutional neural network architectures were evaluated. Transformer models included Grounding DINO [10], Florence-2 [24], and OWL-ViT [13], as for CNN models all YOLOv13 [8] variants were evaluated. These models were used to detect objects and their corresponding spatial locations within each image.

Based on the detected objects and their bounding boxes, three distinct image comparison testing configurations were applied:

- **Strict object correspondence**: Two images are classified as near-duplicates if all detect objects belong to the same classes in both images and their corresponding bounding boxes intersect.
- **Relaxed object correspondence**: Mostly similar to the previous strategy, however this strategy introduces an additional spatial margin to the bounding boxes, allowing for larger position variations of the objects detect in the images.
- **Relaxed and weighted object correspondence**: This approach expands the relaxed approach of the previously stated strategy by allowing two images to still be classified as near-duplicates even if the number of detected objects differs. An extra parameter to determine if two images are near-duplicates is thus considered by taking into account the total number of detected objects and the size of their bounding boxes.

All models were evaluated using their default, *out-of-the-box* configurations to ensure a fair comparison and to reflect realistic deployment scenarios. In the following subsections a brief explanation of how the stated models were implemented is provided.

#### 4.2.2.1 Grounding Dino

The Grounding DINO model was deployed using the *Tiny* variant to perform object detection tasks. In order to evaluate near-duplicate images, two types of text prompts were used. The first one consisted of the most common urban objects, and a second consisted of objects specifically associated with urban occurrences.

An example of object detection with the urban objects prompt is shown in Figure 4.13, with bounding boxes highlighting the detected objects.




Figure 4.13: Grounding Dino object detection example - urban objects.

#### 4.2.2.2 OWL-ViT

The OWL-ViT model was deployed using the *base* variant. Similarly to the Grounding Dino implementation, the model was evaluated using with two different prompts. The prompts consisted of common urban objects and urban occurrence objects.

Figure 4.14 shows an example of OWL-ViT detecting urban objects, with bounding boxes highlighting those.




Figure 4.14: OWL-ViT object detection example - urban objects.

#### 4.2.2.3 Florence-2

For the Florence-2 implementation, it was evaluated using the variant *Base*. This model was evaluated under two different testing configurations. In the first configuration, *OD*, the model attempts to detect all possible objects without any extra text query. The second configuration was *CAPTION_TO_PHRASE_GROUNDING*, where the same prompts as the previous models were evaluated.

An example of Florence-2 detection in OD configuration is shown in Figure 4.15, with bounding boxes highlighting the detected objects.




Figure 4.15: Florence-2 object detection example - OD Prompt.

#### 4.2.2.4 YOLO v13

All YOLOv13 variants were evaluated in their pre-trained configurations using the COCO (Common Objects in Context) dataset. No prompts were deployed to detect extra objects.

Examples of detected objects are shown in Figure 4.16.




Figure 4.16: YOLOv13 object detection example - No Prompt.

#### 4.2.2.5 CLIP Embeddings

CLIP embeddings were implemented in the near-duplicate scenario, where for each image in the comparison pair, the model generates a fixed length vector. The similarity between two images was computed as the cosine similarity between their embeddings. Images with cosine similarity above a defined threshold were considered near-duplicates.

## 4.3 Iterative Image Comparison Procedure

Both traditional and deep learning computer vision similarity techniques, described in the previous Section 4.2, were applied to the dataset following the same sequential procedure. Images were processed iteratively in chronological order, starting from the image with the earliest timestamp and progressing to the most recent one. Near-duplicate detection was performed by comparing consecutive image pairs, as illustrated in Figure 4.17. In the example shown, a dataset consisting of a total of 14 images results in 13 pairwise comparisons executed.

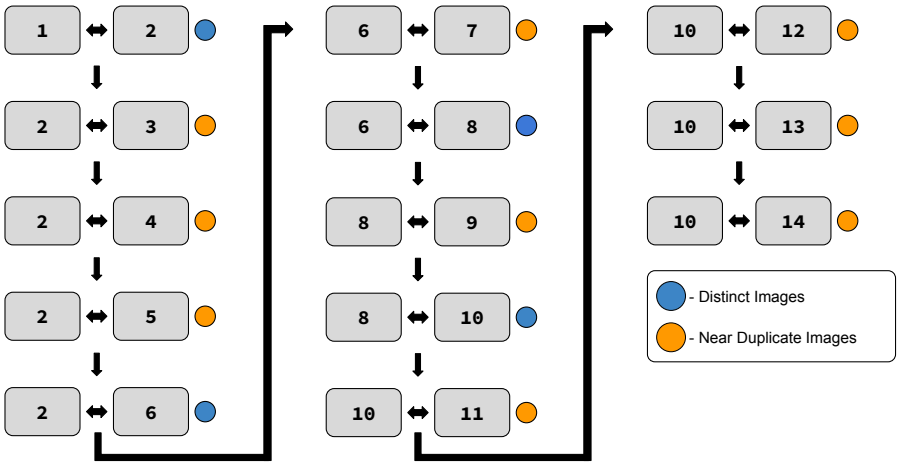



Figure 4.17: Pairwise Comparison Iteration.

In Figure 4.18 a conceptual overview of the dataset organization before and after applying the near-duplicate image removal algorithm is provided. As illustrated, the initial dataset of 14 images is reduced to a final dataset of 5 representative images with redundant near-duplicate images being removed. Together, Figures 4.17 and 4.18 demonstrate the effectiveness of the proposed iterative strategy.

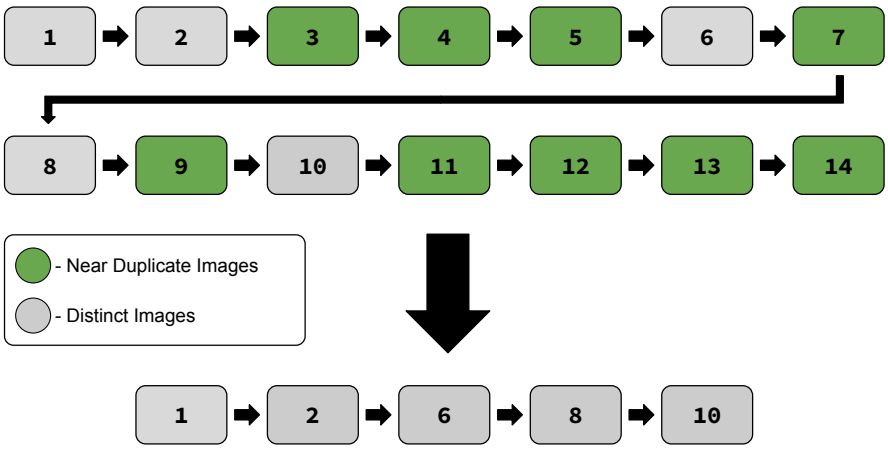



Figure 4.18: Dataset conceptual Overview.

## 4.4 Evaluation Methodology

In previous chapters, near-duplicate images were described as different images sharing the same semantic content. While intuitive, this definition is difficult to implement when quantitative evaluation is required. The ambiguity arises from the context in which the semantic similarity is being evaluated, two images may be considered semantically equivalent under one interpretation and distinct under another. For example, a detection model that merely detects the presence of a person may present two images as semantically similar regardless of whether the person is standing

or dancing, whereas a detection model focused on pose estimation would exhibit these images as semantically different.

As such, a better task-oriented definition of near-duplicate images is adopted in this study. A *near-duplicate image* is defined as an image that can be removed from the dataset without invalidating the inference results of a future model applied to that dataset. A near-duplicate image is an individual image that doesn't have any individual quality, not contributing with unique information to the dataset. When comparing the results of an inference model that was deployed on a complete dataset to the results of the same model deployed on a reduced dataset, where near-duplicates were removed, the results should present the same number of unique urban occurrences.

The evaluation of algorithms for near-duplicate image removal relies on two complementary metrics, both essential for characterizing algorithmic performance. The first metric is *image removal rate*, defined as the percentage of images removed from the original dataset, the second metric is the *occurrence retention rate*, defined as the percentage of distinct occurrences that remain in the dataset after the near-duplicate images were removed.

These two metrics exhibit a clear trade-off. Increasing the image removal rate typically leads to a decrease in the occurrence retention rate, as a more aggressive strategy risks removing images that contribute with unique semantic information. As such, a perfect algorithm that simultaneously maximizes both metrics is unattainable. Rather than optimizing a single metric, the objective of this work is to compare different algorithms across these two dimensions in order to provide insight into their strengths and limitations under different levels of strictness.

To contextualize the results, a *control algorithm* was introduced. This baseline method does not use any image content or similarity information to guide the removal decisions. Instead, it removes images purely based on a predefined removal ratio, for example by sequentially removing four out of every five images in dataset. With this strategy it is possible to establish a lower bound baseline for algorithmic performance.

The control algorithm also allows the occurrence retention rate to be estimated as a function of the image removal rate. For a given percentage of removed images, if a similarity algorithm yields a lower retention rate than the control strategy, it can be concluded that the algorithm performs worse than blindly removing images from a dataset. As such, any algorithm that performs worse than this strategy can be classified as a substandard algorithm.

Several alternative validation strategies were explored during the design phase of this study, but were ultimately discarded due to the introduction of additional sources of evaluation error. The first approach involved an existing inference model developed by the *Theia Vision* team, capable of detecting eight different object classes. The detected bounding boxes, together with temporal metadata, and coordinates of the location the image was taken were used to construct a *visual footprint* for each image, which could then be compared across the reduced dataset to determine how much of the original information of the dataset was retained after the near-duplicate image removal.

However, this approach had two major limitations. First, the model was not specifically trained to detect urban occurrences, leading to a uniformed conclusion of the semantic content present in

an image. The second problem was that defining consistent thresholds for spatial overlap, temporal proximity, and geographic distance introduced significant validation bias. As a consequence, after applying near-duplicate removal, multiple false positives occurred when attempting to validate the reduced dataset. Examples are illustrated in Figures 4.19 and 4.20, where the validation strategy incorrectly validated visually distinct images as near-duplicates.

Although some of these issues could potentially be mitigated by deploying a hypothetical more complex detection model to generate visual footprints, this solution would introduce additional challenges. Specifically, increasing the number of detectable classes would lead to higher object detection errors with greater variability between frames across the complete dataset. This strategy as such would fail by simply intensifying the validation inconsistencies that were trying to be reduced.




Figure 4.19: False positive first example.




Figure 4.20: False positive second example.

A similar approach was still investigated using the already manually annotated urban occurrences, however, the same limitations persisted. Distinct occurrences presented highly variable spatial, temporal, and geographic characteristics, which can be seen in both Table 4.1 and Figure 4.7. This stated variation makes it difficult to define universal threshold values without introducing errors. As a result, the final evaluation methodology relied on manual grouping of consecutive images containing the same manually annotated occurrences, as described in Section 4.1.2, providing an error-free ground truth for occurrence retention.

## 4.5 Testing Systems

All algorithms described in Section 4.2 were evaluated with two distinct hardware platforms representing the main server and the edge device deployment scenarios.

The main server was simulated using a desktop computer, using Ubuntu 24.04 [3] as the operating system, and equipped with the following specifications:

- **CPU**: AMD Ryzen 9 7950X3D;

- **GPU**: RTX 5080 MSI Ventus 3X OC Plus 16G GDDR7;
- **Motherboard**: ROG STRIX X670E-E GAMING WIFI;
- **SSD**: M.2 2280 KIOXIA Exceria 500GB 3D TLC NVMe;
- **RAM**: G.SKILL Flare X5 64GB (4x16GB) DDR5-6000MHz CL30.

This stated configuration represents a relatively high performance environment suitable for centralized processing.

The edge device platform consisted of a Raspberry Pi 5 with 8 GB of RAM, with the official case, which has active cooling. This system represents a resource constrained environment typical of edge deployments, where computational power, memory, and energy availability are limited.

Evaluating both platforms enables a comparative analysis of execution time and allows for the assessment of algorithm suitability for edge versus server deployment. In addition to execution time, several extra system metrics were monitored on the edge device, including CPU utilization, memory usage, energy consumption, operating temperature, and error logging. It is important to highlight error logging, since in scenarios where there is no active cooling, errors of thermal throttling are displayed.

# Chapter 5

# Results

As stated in section 4.4, all similarity techniques were evaluated with a sample dataset composed of 80 000 images and 328 distinct urban occurrences. The similarity techniques are evaluated across two different metrics, *image removal rate* and *occurrence retention rate*. The image removal rate represents the percentage of images removed from the dataset, while the occurrence retention rate represents the percentage of distinct urban occurrences retained after reduction. An urban occurrence is considered retained if at least one image from its annotated bag remains in the reduced dataset.

Two evaluation methodologies were employed to better understand how the evaluated algorithms behave under varying levels of temporal persistence. Low temporal persistence is characterized by a low number of images per urban occurrence, whereas high temporal persistence corresponds to occurrences represented by a high number of images.

The first evaluation considers the full set of 328 distinct urban occurrences as the denominator for the *occurrence retention rate*. This reflects the normal characteristics of the dataset, where most urban occurrences persist across multiple consecutive frames.

The second evaluation focuses exclusively on urban occurrences composed of at most three images, totaling 51 distinct occurrences. In this scenario, the *occurrence retention rate* denominator is restricted to these occurrences for better assessment of the algorithms characteristics under conditions characterized by rapid scene changes.

In both evaluation methodologies, the complete dataset of 80 000 images is used as the common denominator for computing the *image removal rate*.

Sections 5.1, 5.2, and 5.3 present the results for the control algorithm, traditional similarity techniques, and deep learning similarity techniques, respectively, using both evaluation methodologies. Section 5.4 compares the performance of the selected similarity techniques. Section 5.5 presents both the distributions of pairwise similarity classification times, and total execution times, highlighting the differences between the edge device and the server system. Finally, Section 5.6 details edge device metrics for the selected techniques (Perceptual Hashing, SIFT Keypoint Mask, Grounding DINO, and CLIP embeddings), including RAM usage, temperature, CPU usage, energy consumption, and CPU frequency over the total execution period.

## 5.1 Control Algorithm

### 5.1.1 Control Algorithm - Global Evaluation Methodology

Table 5.1 presents the results obtained using the global evaluation methodology for the control algorithm. The table reports both image removal rate and occurrence retention rate across different threshold configurations.

Figure 5.1 illustrates the Pareto fronts for the control algorithm under the global evaluation methodology.

Table 5.1: Control algorithm metrics - global evaluation methodology.

| Retention Level (Images) | Image Removal Rate (%) | Occurrence retention rate (%) |
|---|---|---|
| 4 out of 5 | 19.99 | 99.09 |
| 3 out of 4 | 24.99 | 99.09 |
| 2 out of 3 | 33.33 | 98.48 |
| 1 out of 2 | 50.00 | 97.87 |
| 1 out of 3 | 66.66 | 96.04 |
| 1 out of 4 | 75.00 | 93.29 |
| 1 out of 5 | 80.00 | 86.89 |
| 1 out of 6 | 83.33 | 84.45 |
| 1 out of 7 | 85.71 | 81.40 |
| 1 out of 8 | 87.49 | 76.52 |
| 1 out of 9 | 88.88 | 70.73 |
| 1 out of 10 | 90.00 | 68.60 |
| 1 out of 15 | 93.32 | 56.70 |
| 1 out of 20 | 94.99 | 46.65 |
| 1 out of 25 | 95.99 | 35.37 |

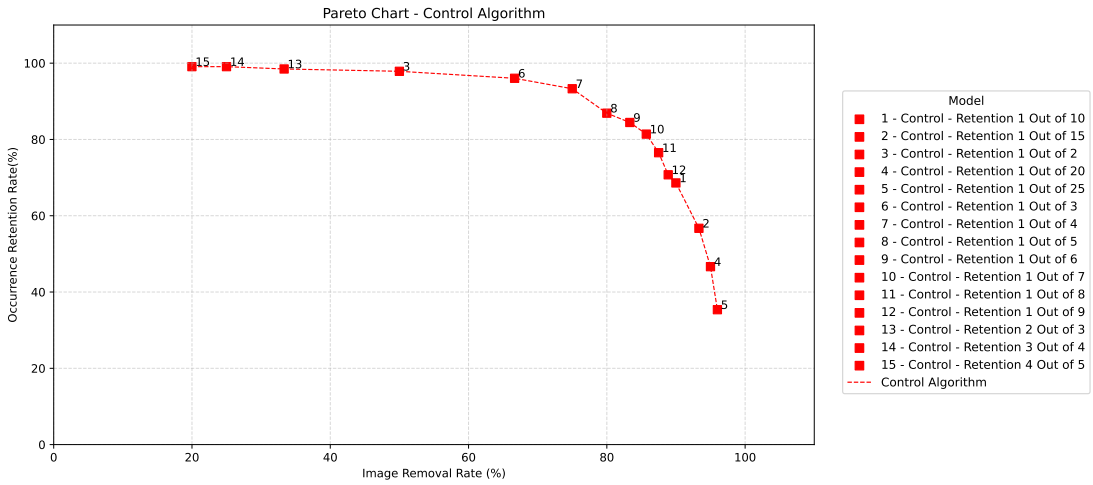



Figure 5.1: Pareto Chart with control algorithm results - global evaluation methodology.

### 5.1.2 Control Algorithm - Focused Evaluation Methodology

Table 5.2 presents the results obtained using the focused evaluation methodology for the control algorithm. The table reports both image removal rate and occurrence retention rate across different threshold configurations.

Figure 5.2 illustrates the Pareto fronts for the control algorithm under the focused evaluation methodology.

Table 5.2: Control algorithm metrics - focused evaluation methodology.

| Retention Level (Images) | Image Removal Rate (%) | Occurrence retention rate (%) |
|---|---|---|
| 4 out of 5 | 19.99 | 94.12 |
| 3 out of 4 | 24.99 | 94.12 |
| 2 out of 3 | 33.33 | 90.20 |
| 1 out of 2 | 50.00 | 86.27 |
| 1 out of 3 | 66.66 | 74.51 |
| 1 out of 4 | 75.00 | 56.86 |
| 1 out of 5 | 80.00 | 35.29 |
| 1 out of 6 | 83.33 | 33.33 |
| 1 out of 7 | 85.71 | 35.29 |
| 1 out of 8 | 87.49 | 15.69 |
| 1 out of 9 | 88.88 | 23.53 |
| 1 out of 10 | 90.00 | 13.73 |
| 1 out of 15 | 93.32 | 15.69 |
| 1 out of 20 | 94.99 | 09.80 |
| 1 out of 25 | 95.99 | 01.96 |

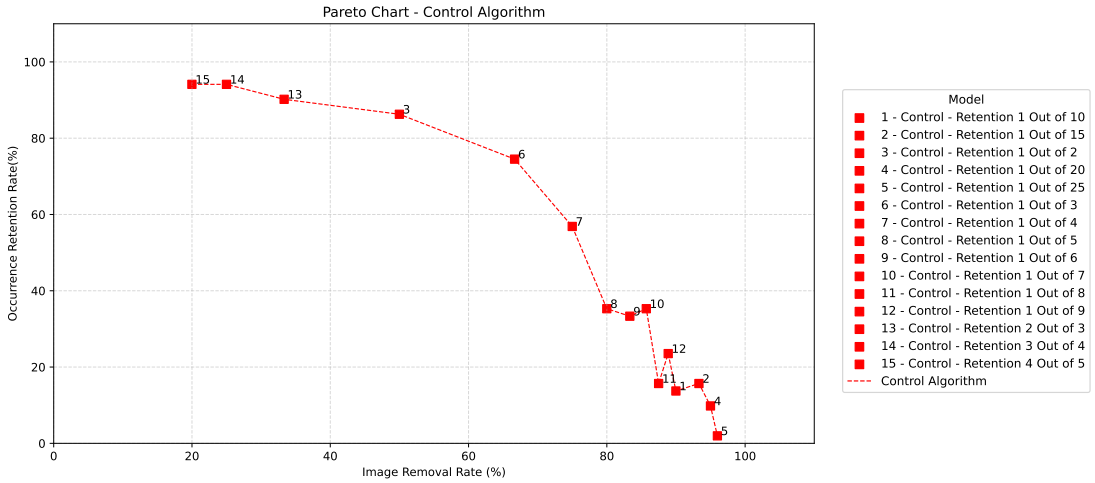



Figure 5.2: Pareto Chart with control algorithm results - focused evaluation methodology.

## 5.2 Traditional Similarity Techniques

### 5.2.1 Traditional Similarity Techniques - Global Evaluation Methodology

Table 5.3 presents the results obtained using the global evaluation methodology for the traditional similarity techniques. The table reports both image removal rate and occurrence retention rate across different threshold configurations.

Figure 5.3 illustrates the Pareto fronts for the traditional similarity techniques under the global evaluation methodology, together with the control strategy.

Table 5.3: Traditional similarity techniques metrics - global evaluation methodology.

| Similarity Technique | Strictness | Image Removal Rate (%) | Occurrence retention rate (%) |
|---|---|---|---|
| pHash | Threshold of 6 | 43.59 | 100.0 |
| | Threshold of 10 | 50.89 | 99.39 |
| | Threshold of 14 | 58.99 | 99.08 |
| | Threshold of 18 | 67.97 | 97.56 |
| | Threshold of 25 | 82.21 | 89.32 |
| | Threshold of 32 | 96.47 | 38.72 |
| SSIM | Threshold of 0.9 | 34.75 | 100.0 |
| | Threshold of 0.8 | 44.77 | 99.09 |
| | Threshold of 0.7 | 61.18 | 95.12 |
| | Threshold of 0.6 | 81.58 | 73.48 |
| SIFT Keypoint Mask | Lowe of 0.80 | 54.15 | 100.0 |
| | Lowe of 0.85 | 74.62 | 94.82 |
| | Lowe of 0.90 | 98.69 | 21.34 |

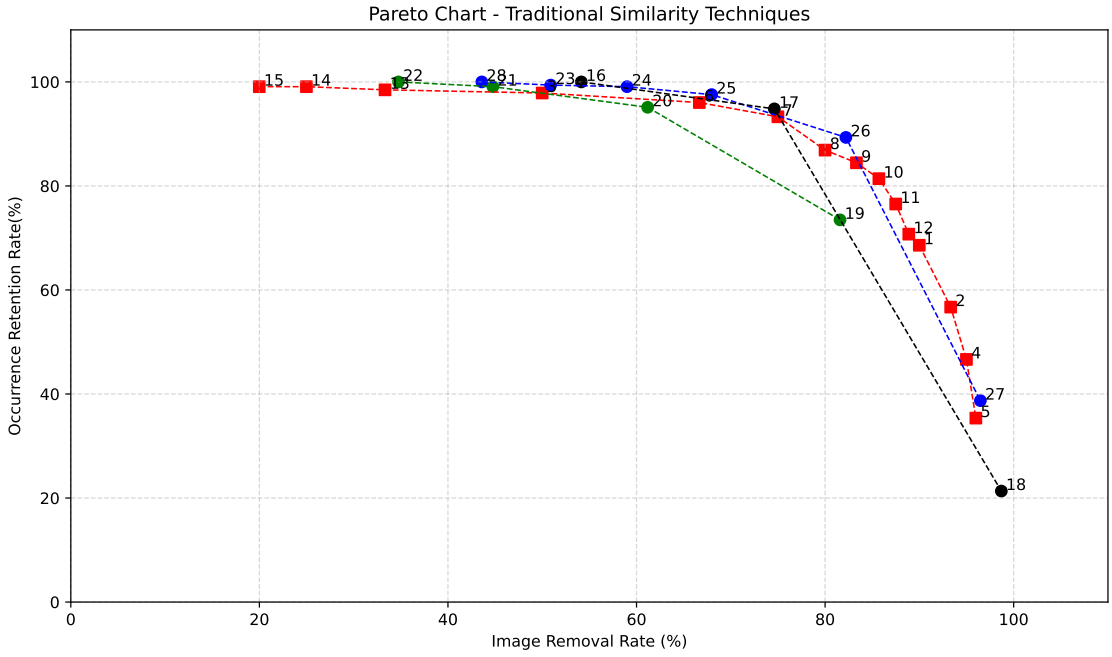



Figure 5.3: Pareto Chart with traditional similarity techniques - global evaluation methodology.

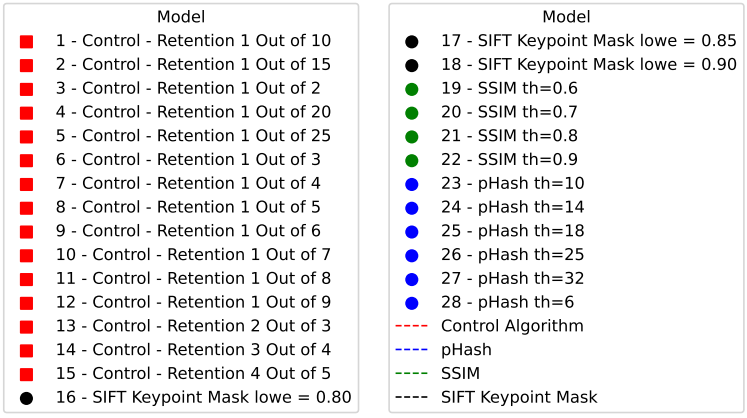



Figure 5.4: Legend for Figure 5.3.

### 5.2.2 Traditional Similarity Techniques - Focused Evaluation Methodology

Table 5.4 presents the results for the traditional similarity techniques obtained using the evaluation methodology focused on occurrences with at most 3 images. The table reports both image removal rate and occurrence retention rate across different threshold configurations.

Figure 5.5 illustrates the Pareto fronts for the traditional similarity techniques under the focused evaluation methodology, together with the control strategy.

Table 5.4: Traditional similarity techniques metrics - focused evaluation methodology.

| Similarity Technique | Strictness | Image Removal Rate (%) | Occurrence retention rate (%) |
|---|---|---|---|
| pHash | Threshold of 6 | 43.59 | 100.0 |
| | Threshold of 10 | 50.89 | 96.08 |
| | Threshold of 14 | 58.99 | 94.12 |
| | Threshold of 18 | 67.97 | 90.20 |
| | Threshold of 25 | 82.21 | 72.55 |
| | Threshold of 32 | 96.47 | 11.76 |
| SSIM | Threshold of 0.9 | 34.75 | 100.0 |
| | Threshold of 0.8 | 44.77 | 98.04 |
| | Threshold of 0.7 | 61.18 | 90.20 |
| | Threshold of 0.6 | 81.58 | 52.94 |
| SIFT Keypoint Mask | Lowe of 0.80 | 54.15 | 100.0 |
| | Lowe of 0.85 | 74.62 | 78.43 |
| | Lowe of 0.90 | 98.69 | 15.69 |

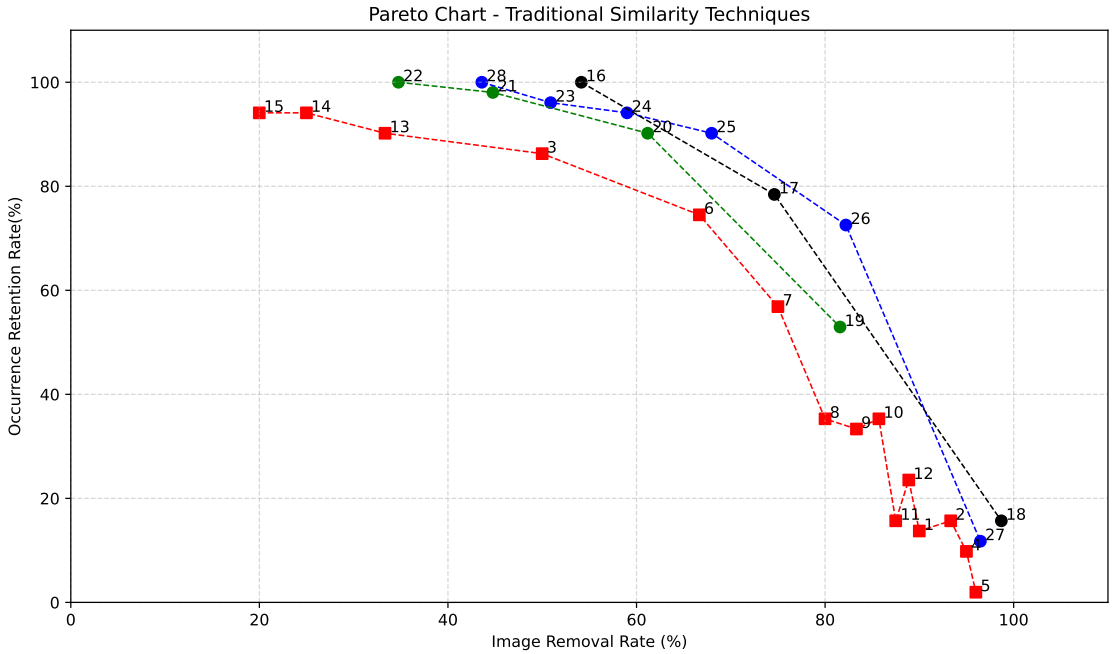



Figure 5.5: Pareto Chart with traditional similarity techniques - focused evaluation methodology.

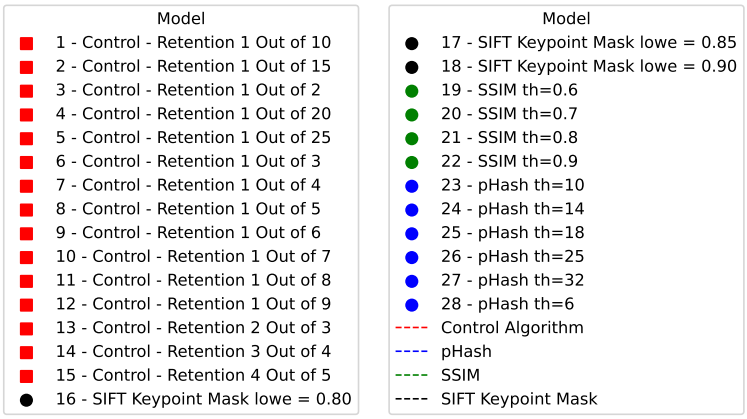



Figure 5.6: Legend for Figure 5.5.

## 5.3 Deep Learning Techniques

### 5.3.1 Deep Learning Techniques - Global Evaluation Methodology

Table 5.5 presents the results obtained using the global evaluation methodology for the deep learning similarity techniques. The table reports both image removal rate and occurrence retention rate across different threshold configurations.

Figure 5.7 illustrates the Pareto fronts for the deep learning similarity techniques under the global evaluation methodology, together with the control strategy.

Table 5.5: Deep Learning Similarity Techniques Metrics - global evaluation methodology.

| **Deep Learning Model** | **Strategy** | **Image Removal Rate (%)** | **Occurrence retention rate (%)** |
|---|---|---|---|
| YOLOv13 - N | Strict | 35.95 | 98.78 |
| | Relaxed | 43.28 | 96.65 |
| | Relaxed & Weighted | 52.46 | 91.46 |
| YOLOv13 - S | Strict | 32.86 | 99.09 |
| | Relaxed | 40.67 | 96.34 |
| | Relaxed & Weighted | 49.29 | 93.29 |
| YOLOv13 - L | Strict | 39.23 | 98.17 |
| | Relaxed | 47.81 | 96.04 |
| | Relaxed & Weighted | 58.35 | 90.85 |
| YOLOv13 - X | Strict | 35.12 | 98.78 |
| | Relaxed | 44.40 | 96.65 |
| | Relaxed & Weighted | 55.28 | 91.16 |
| OWL-ViT - Occurrences Objects | Strict | 41.33 | 84.76 |
| | Relaxed | 42.34 | 84.15 |
| | Relaxed & Weighted | 44.48 | 79.88 |
| OWL-ViT - Urban Objects | Strict | 36.62 | 92.38 |
| | Relaxed | 38.77 | 92.07 |
| | Relaxed & Weighted | 42.39 | 87.80 |
| Grounding Dino - Occurrences Objects | Strict | 27.23 | 100.0 |
| | Relaxed | 31.92 | 99.70 |
| | Relaxed & Weighted | 45.24 | 98.48 |
| Grounding Dino - Urban Objects | Strict | 28.15 | 100.0 |
| | Relaxed | 31.77 | 100.0 |
| | Relaxed & Weighted | 48.68 | 98.78 |
| Florence 2 - Occurrences Objects | Strict | 18.58 | 99.70 |
| | Relaxed | 23.30 | 99.70 |
| | Relaxed & Weighted | 48.39 | 97.56 |
| Florence 2 - Urban Objects | Strict | 16.52 | 100.0 |
| | Relaxed | 21.08 | 99.70 |
| | Relaxed & Weighted | 47.45 | 98.17 |
| Florence 2 - OD | Strict | 45.60 | 97.87 |
| | Relaxed | 58.95 | 96.95 |
| | Relaxed & Weighted | 74.34 | 92.68 |
| CLIP | Threshold of 0.99 | 32.06 | 100.0 |
| | Threshold of 0.98 | 39.49 | 100.0 |
| | Threshold of 0.97 | 48.60 | 100.0 |
| | Threshold of 0.96 | 58.06 | 99.39 |
| | Threshold of 0.95 | 66.25 | 98.17 |
| | Threshold of 0.93 | 77.71 | 96.34 |
| | Threshold of 0.92 | 81.54 | 95.12 |
| | Threshold of 0.90 | 86.93 | 91.46 |
| | Threshold of 0.85 | 93.64 | 77.74 |
| | Threshold of 0.80 | 96.68 | 57.93 |

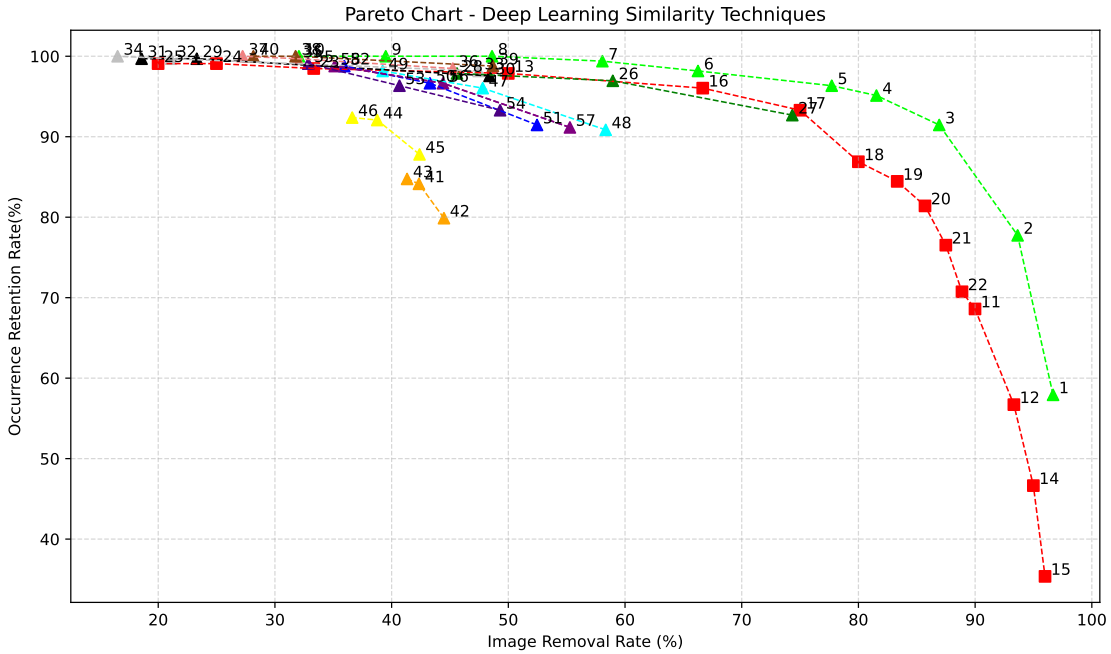



Figure 5.7: Pareto Chart with deep learning similarity techniques - global evaluation methodology.

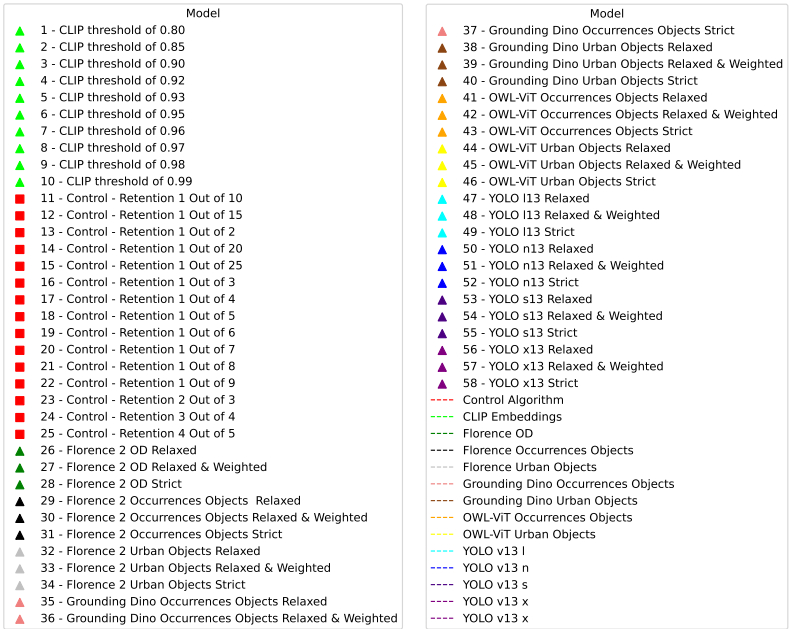



Figure 5.8: Legend for Figure 5.7.

### 5.3.2 Deep Learning Techniques - Focused Evaluation Methodology

Table 5.6 presents the results obtained for the deep learning similarity techniques using the evaluation methodology focused on urban occurrences with at most three images. The table reports both image removal rate and occurrence retention rate across different threshold configurations.

Figure 5.9 illustrates the Pareto fronts for the deep learning similarity techniques under the focused evaluation methodology, together with the control strategy.

Table 5.6: Deep Learning Similarity Techniques Metrics - focused evaluation methodology.

| **Deep Learning Model** | **Strategy** | **Image Removal Rate (%)** | **Occurrence retention rate (%)** |
|---|---|---|---|
| YOLOv13 - N | Strict | 35.95 | 96.08 |
| | Relaxed | 43.28 | 86.27 |
| | Relaxed & Weighted | 52.46 | 70.59 |
| YOLOv13 - S | Strict | 32.86 | 96.08 |
| | Relaxed | 40.67 | 86.27 |
| | Relaxed & Weighted | 49.29 | 76.47 |
| YOLOv13 - L | Strict | 39.23 | 96.08 |
| | Relaxed | 47.81 | 90.20 |
| | Relaxed & Weighted | 58.35 | 68.63 |
| YOLOv13 - X | Strict | 35.12 | 98.04 |
| | Relaxed | 44.40 | 94.12 |
| | Relaxed & Weighted | 55.28 | 74.51 |
| OWL-ViT - Occurrences Objects | Strict | 41.33 | 72.55 |
| | Relaxed | 42.34 | 68.63 |
| | Relaxed & Weighted | 44.48 | 56.86 |
| OWL-ViT - Urban Objects | Strict | 36.62 | 84.31 |
| | Relaxed | 38.77 | 82.35 |
| | Relaxed & Weighted | 42.39 | 72.55 |
| Grounding Dino - Occurrences Objects | Strict | 27.23 | 100.0 |
| | Relaxed | 31.92 | 100.0 |
| | Relaxed & Weighted | 45.24 | 94.12 |
| Grounding Dino - Urban Objects | Strict | 28.15 | 100.0 |
| | Relaxed | 31.77 | 100.0 |
| | Relaxed & Weighted | 48.68 | 92.16 |
| Florence 2 - Occurrences Objects | Strict | 18.58 | 98.04 |
| | Relaxed | 23.30 | 98.04 |
| | Relaxed & Weighted | 48.39 | 86.27 |
| Florence 2 - Urban Objects | Strict | 16.52 | 100.0 |
| | Relaxed | 21.08 | 98.04 |
| | Relaxed & Weighted | 47.45 | 88.24 |
| Florence 2 - OD | Strict | 45.60 | 94.12 |
| | Relaxed | 58.95 | 90.20 |
| | Relaxed & Weighted | 74.34 | 74.51 |
| CLIP | Threshold of 0.99 | 32.06 | 100.0 |
| | Threshold of 0.98 | 39.49 | 100.0 |
| | Threshold of 0.97 | 48.60 | 100.0 |
| | Threshold of 0.96 | 58.06 | 96.08 |
| | Threshold of 0.95 | 66.25 | 88.24 |
| | Threshold of 0.93 | 77.71 | 80.39 |
| | Threshold of 0.92 | 81.54 | 74.51 |
| | Threshold of 0.90 | 86.93 | 58.82 |
| | Threshold of 0.85 | 93.64 | 43.14 |
| | Threshold of 0.80 | 96.68 | 39.22 |

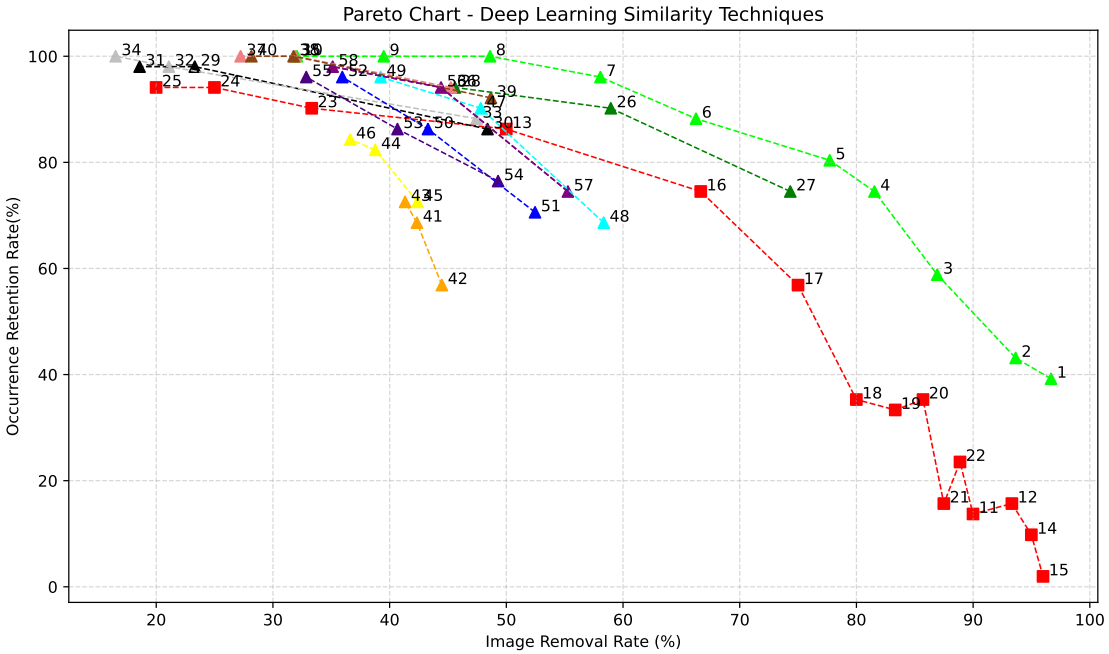



Figure 5.9: Pareto Chart with deep learning similarity techniques - focused evaluation methodology.

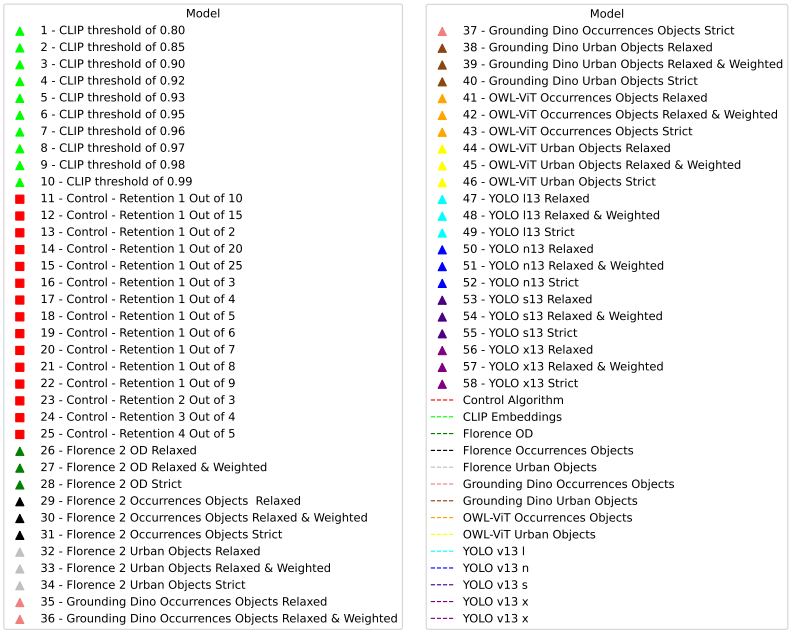



Figure 5.10: Legend for Figure 5.9.

## 5.4 Selected Similarity Techniques

This section presents a comparative evaluation of the near-duplicate image removal techniques that achieved the most competitive results across previous experiments. The comparison focuses on pHash, the SIFT Keypoint Mask approach, and CLIP embeddings.

All selected techniques are compared under the two previously defined evaluation methodologies, and the control algorithm is retained in both plots as a baseline reference.

In Figures 5.11 and 5.13, Pareto fronts illustrate the results of the stated algorithms across the global evaluation methodology and the methodology focused on urban occurrences with at most 3 images.

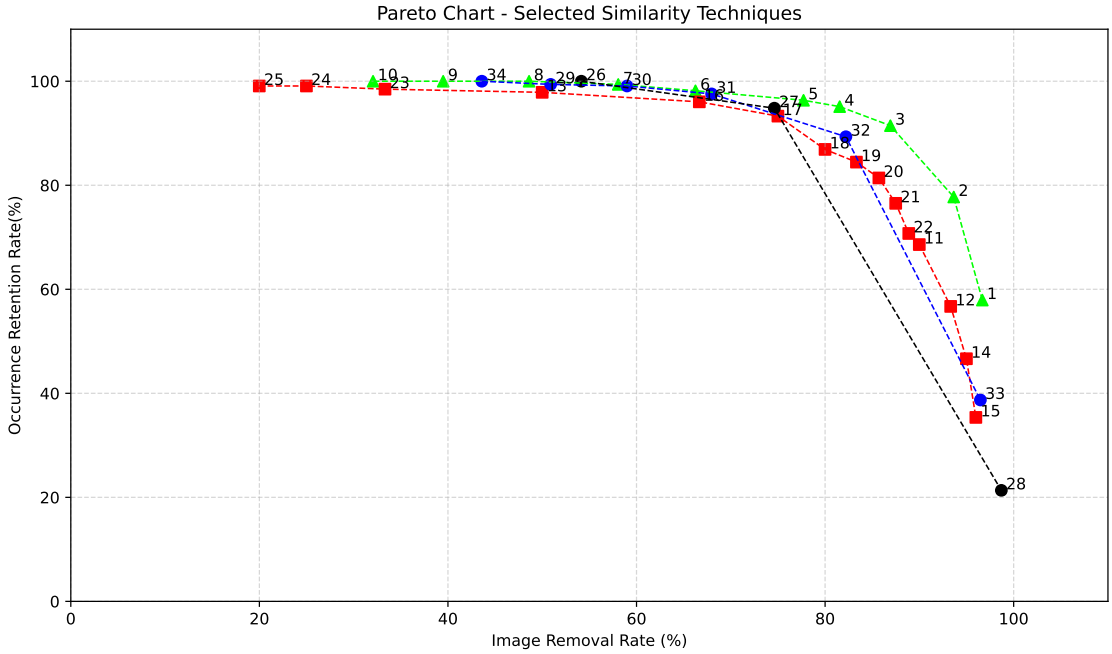



Figure 5.11: Pareto Chart with selected similarity techniques - global evaluation methodology.

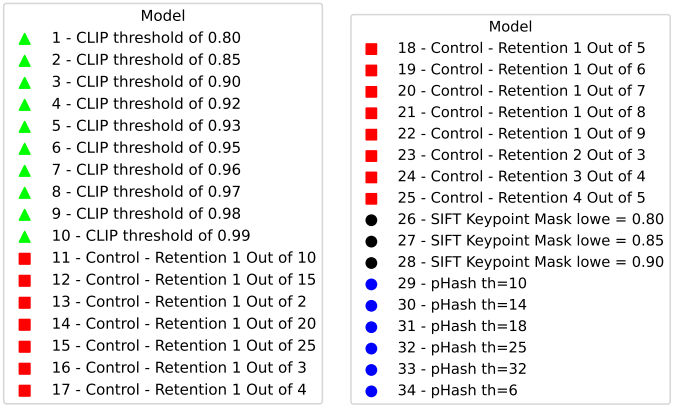



Figure 5.12: Legend for both Figure 5.11 and 5.13.

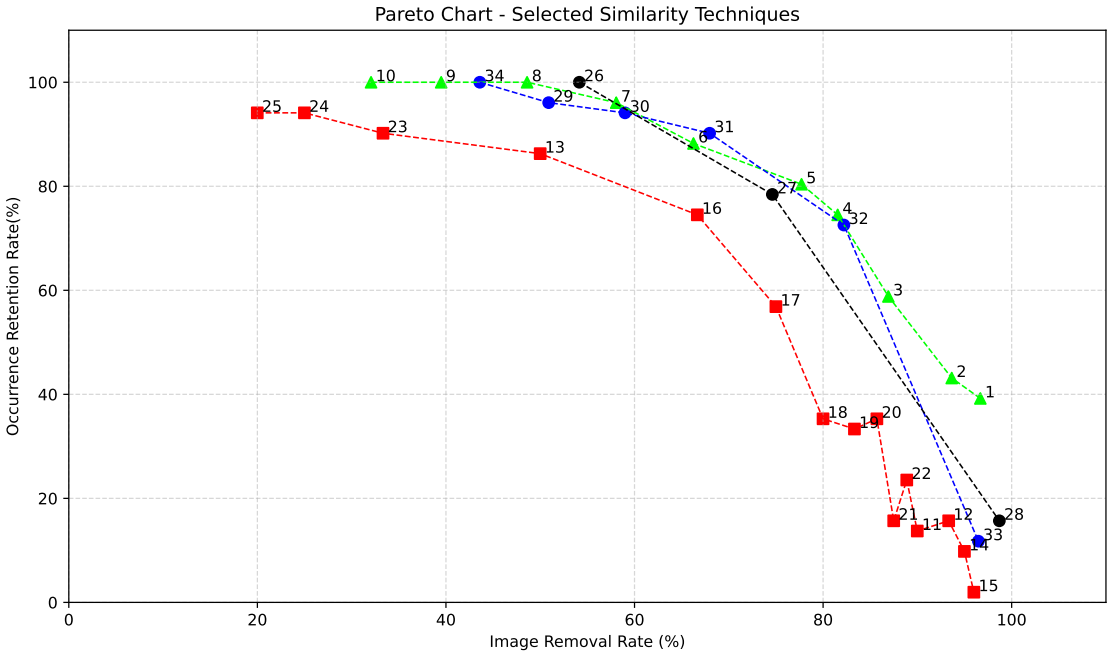



Figure 5.13: Pareto Chart with selected similarity techniques - focused evaluation methodology.

## 5.5 Near-Duplicate Classification Time Metrics

As stated in Section 4.5, all these algorithms were evaluated on the desktop computer (simulating a centralized server) and then ported to edge device platform. In the followings Figures 5.14 and 5.15, across the two different evaluated systems, time distribution comparisons for the assessment of images pairs as near-duplicates can be seen, respectively, for the traditional and deep learning techniques.

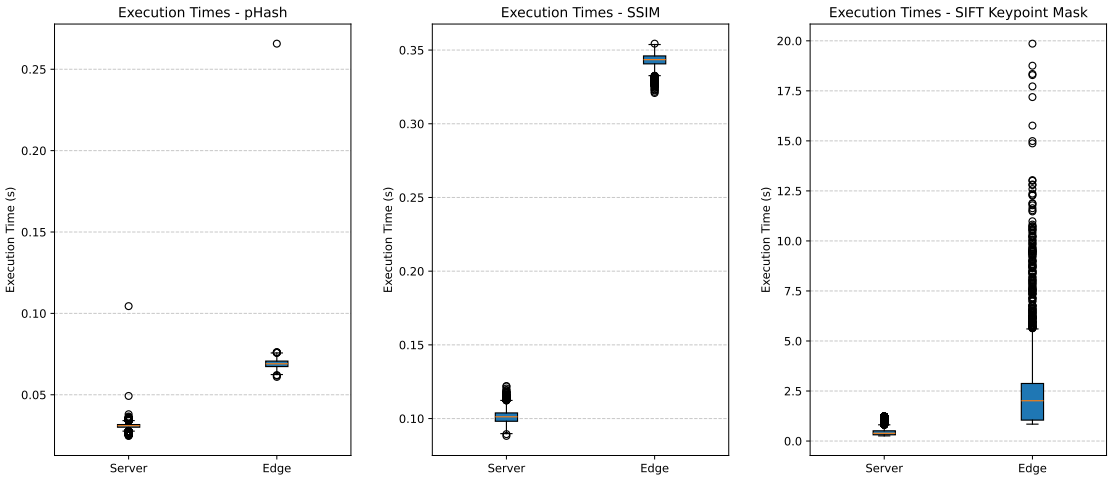



Figure 5.14: Traditional similarity techniques - Time distribution.

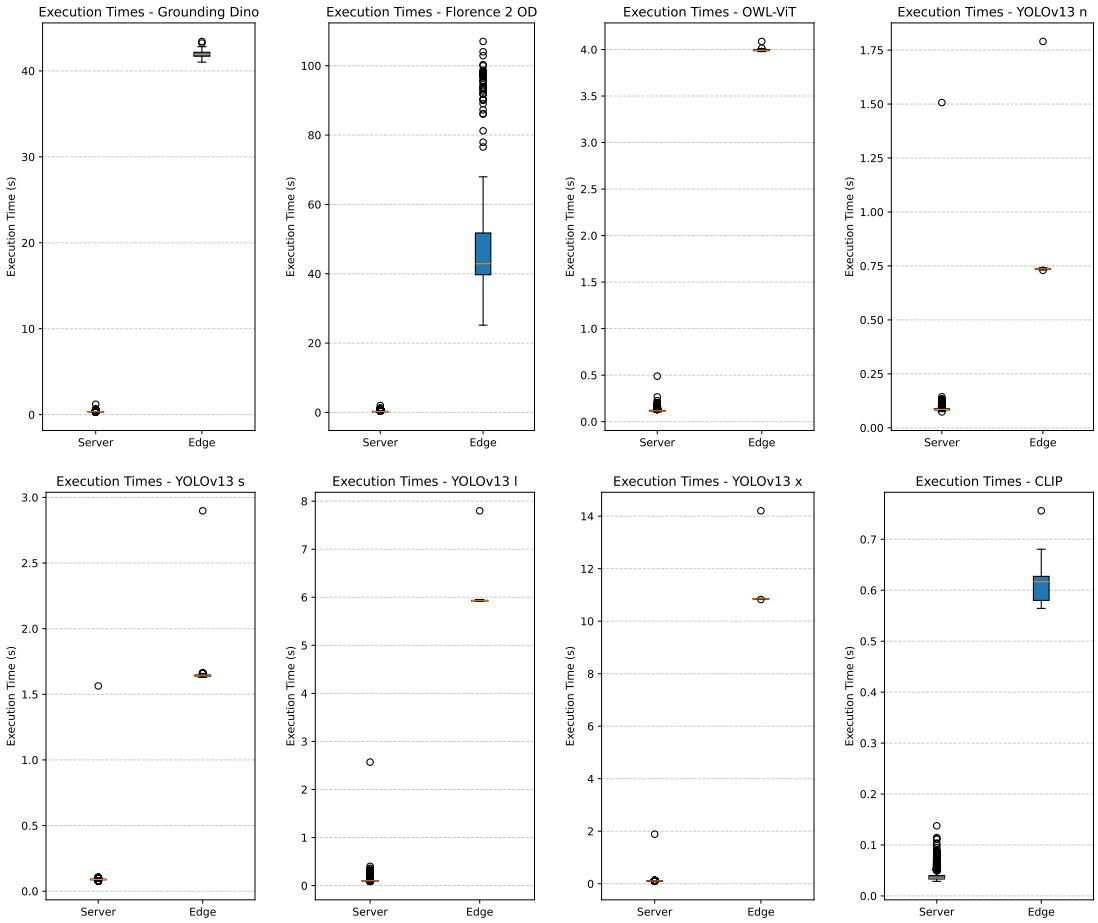



Figure 5.15: Deep learning similarity techniques - Time distribution.

A sample of 1 000 images of the annotated dataset was also gathered to compare total execution times across the two different evaluated systems. In the followings Figures 5.16 and 5.17, total execution time for the assessment of images as near-duplicates can be seen respectively for the traditional and deep learning techniques.

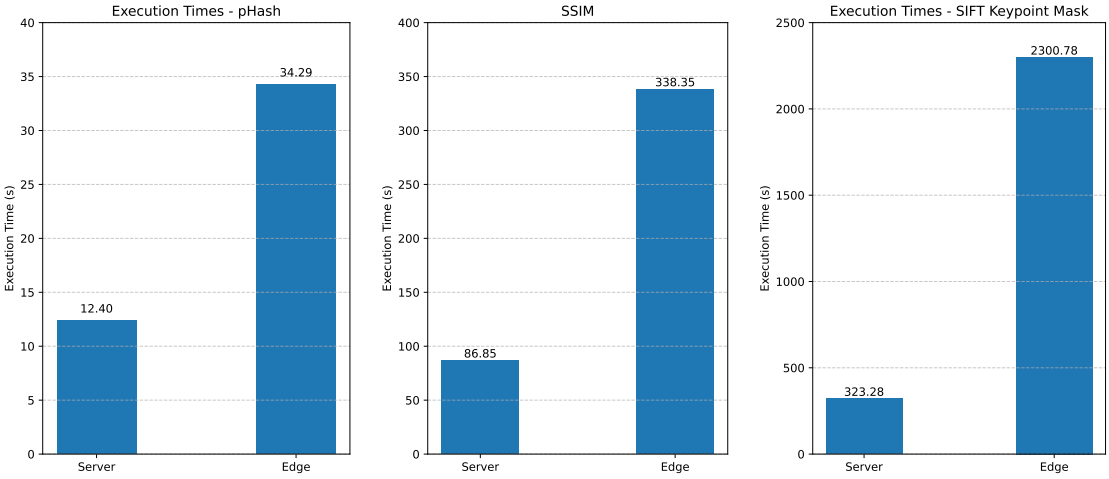



Figure 5.16: Traditional similarity techniques - Total execution time.

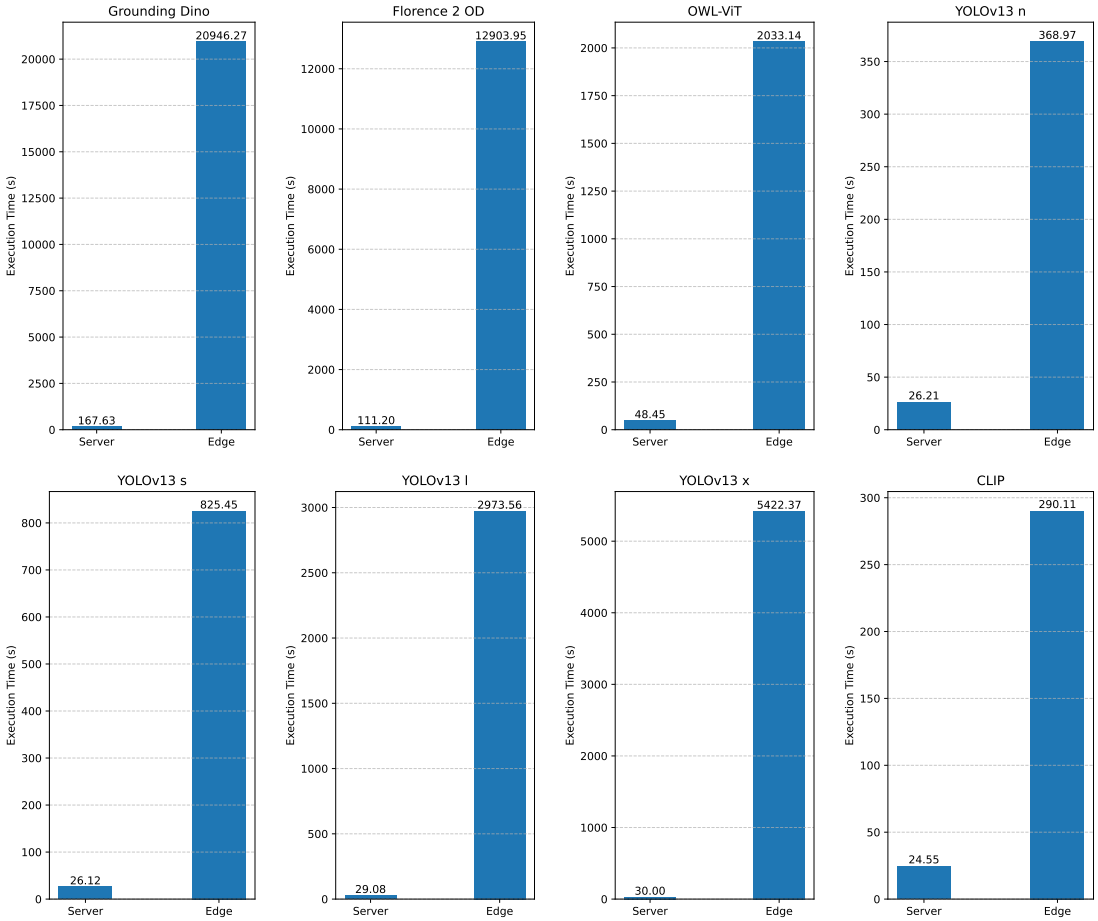



Figure 5.17: Deep learning similarity techniques - Total execution time.

## 5.6 Edge-Device Logged Metrics

This section presents edge device metrics for the selected techniques: Perceptual Hashing, SIFT Keypoint Mask, Grounding DINO, and CLIP embeddings. The metrics include RAM usage, device temperature, CPU usage, energy consumption, and CPU frequency. All these metrics were gathered across the total execution time of each stated similarity technique.

Although all algorithms operate at CPU frequency of 2.4 GHz, as seen in the following results, this behavior is not consistently maintained on the evaluated device when active cooling is not employed. Computationally intensive models result in higher processing loads, which leads to increased device temperature, thermal throttling, and consequently reduced CPU frequency, as illustrated in Appendix A.2.

The following Figure 5.18, presents, respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for SIFT Keypoint Mask on the evaluated edge device.

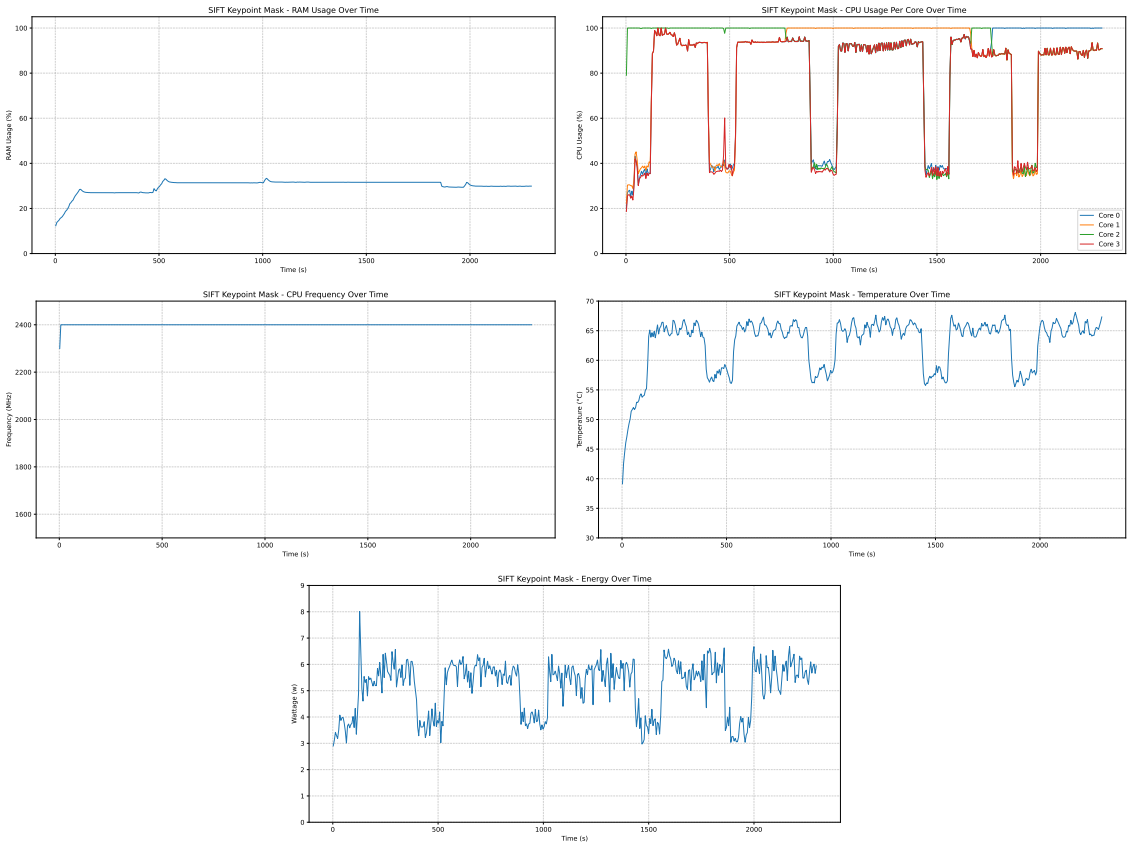



Figure 5.18: SIFT Keypoint Mask metrics - edge device.

The following Figure 5.19, presents, respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for Perceptual Hashing on the evaluated edge device.

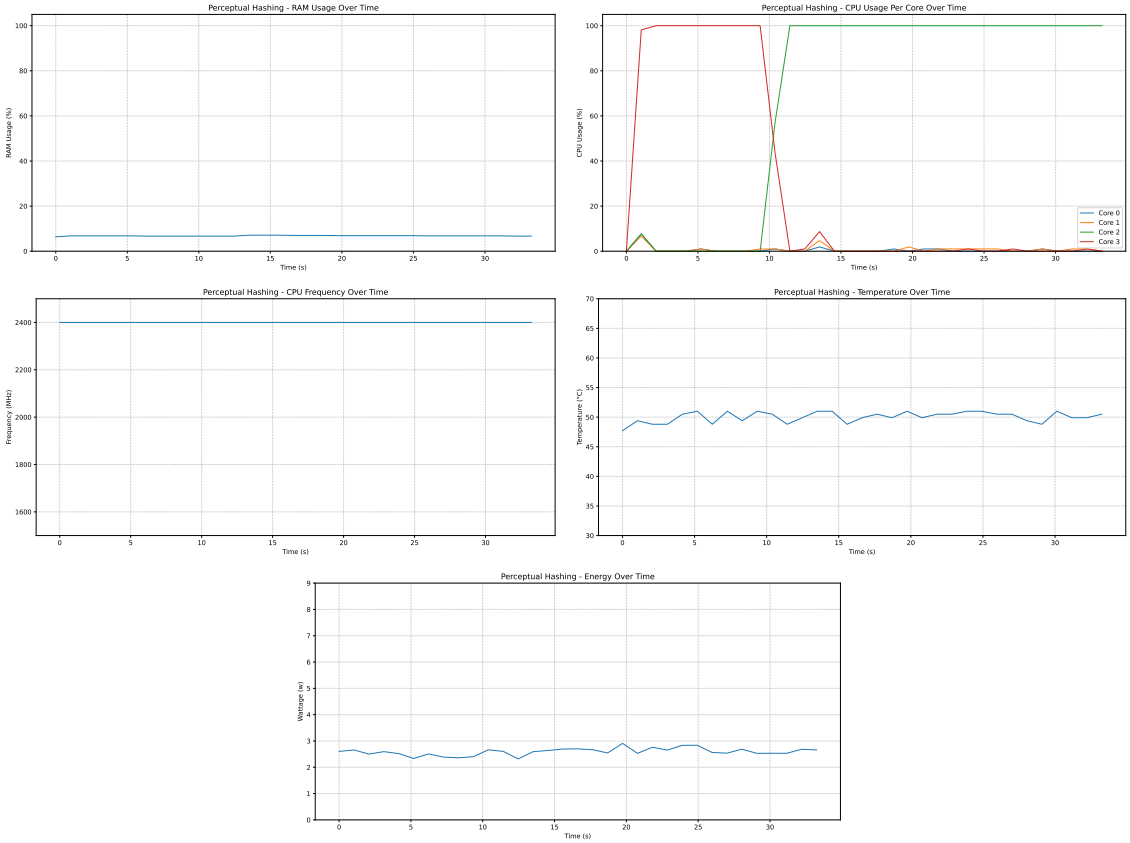



Figure 5.19: Perceptual Hashing metrics - edge device.

The following Figure 5.20, presents, respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for Grounding DINO on the evaluated edge device.

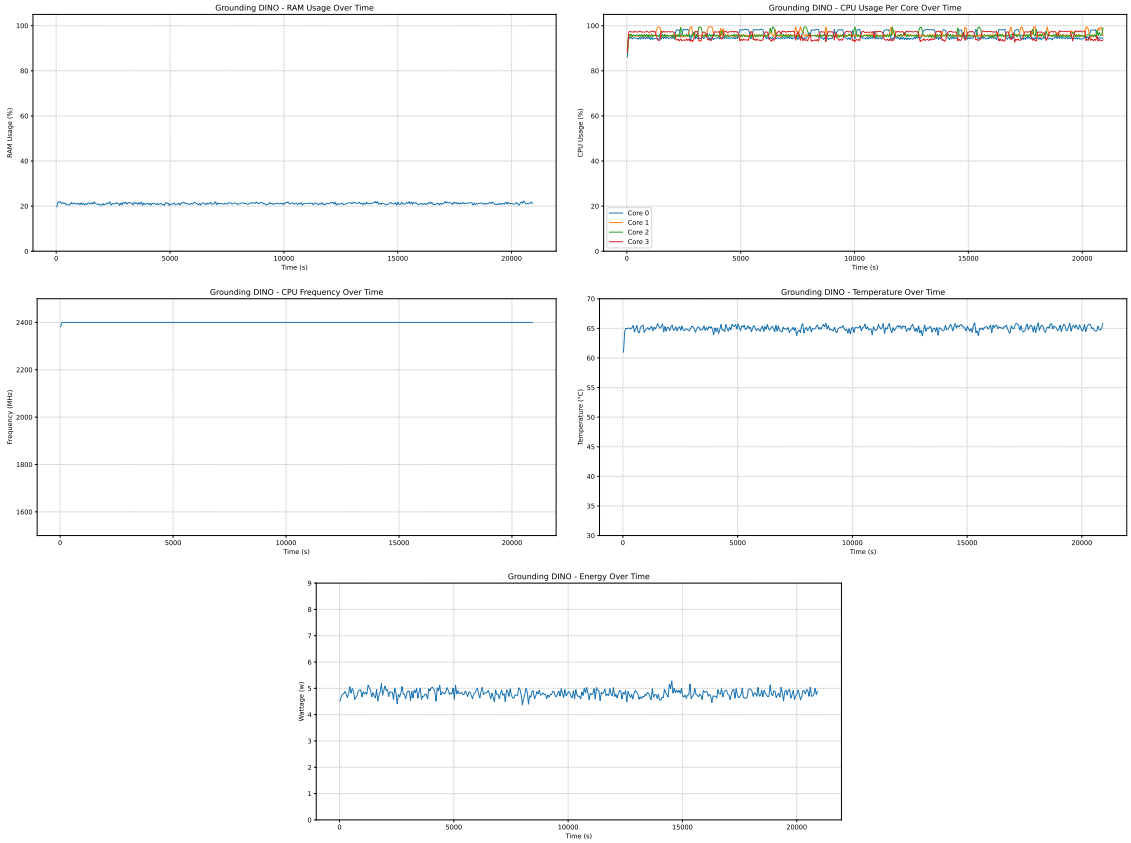



Figure 5.20: Grounding DINO metrics - edge device.

The following Figure 5.21, presents, respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for CLIP embeddings on the evaluated edge device.

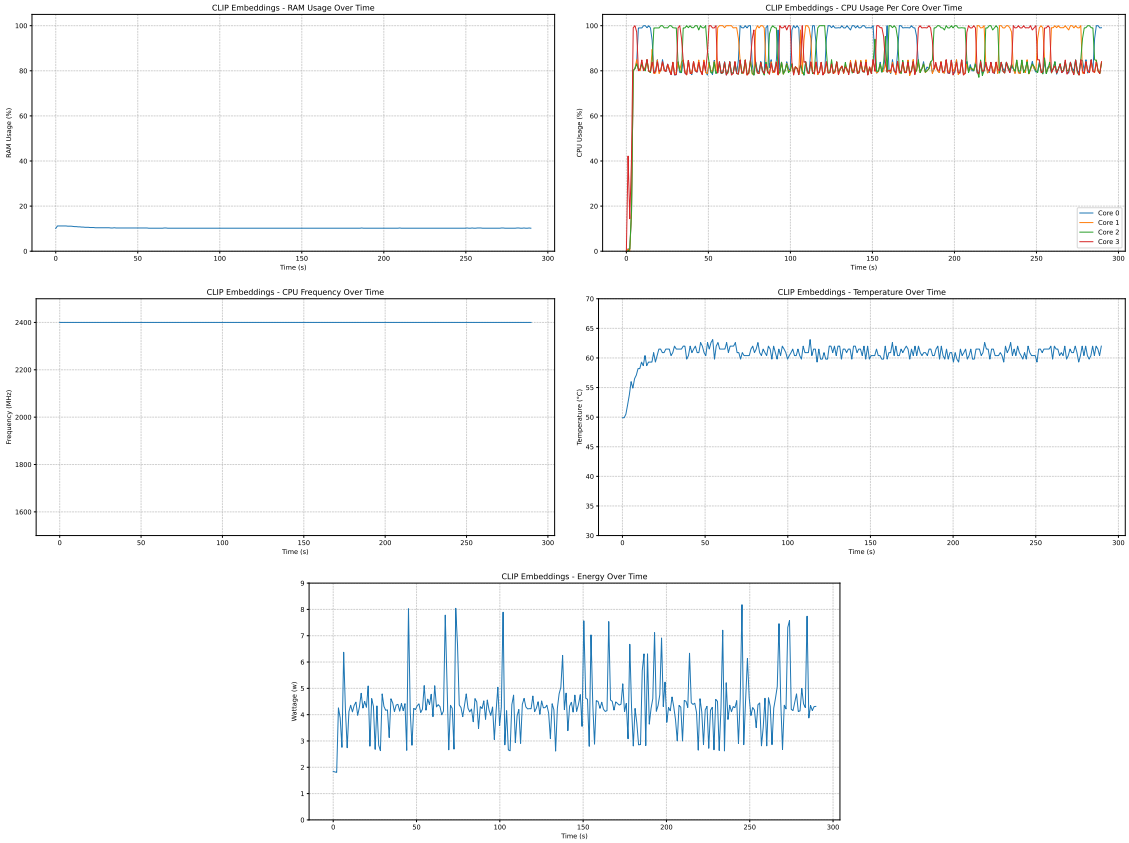



Figure 5.21: CLIP Embeddings metrics - edge device.

# Chapter 6

# Discussion

## 6.1 Traditional Similarity Techniques

Overall, as shown in Section 5.2, the traditional similarity methods achieved competitive performance across the different evaluated configurations. With both perceptual hashing and the SIFT keypoint mask consistently outperforming SSIM across the wide range of image removal rates evaluated. For moderate image removal targets, all three traditional methods generally achieved higher occurrence retention than the control strategy. However, once the similarity thresholds became less strict, their performance degraded and eventually became worse than the control algorithm.

This behavior is largely explained by how threshold similarity decisions are made. At high strictness, only near-identical frames are removed, which allows the occurrence retention to remain high. In such cases, near-duplicates typically correspond to images captured with minimal camera motion and mostly negligible scene variation, such as when the vehicle is stationary. As the strictness is relaxed, in an attempt to identify near-duplicate images under more pronounced scene changes, the decision boundary consequently becomes more noisy, removing distinct frames with unique semantic information that was incorrectly identified. At sufficiently low strictness levels, all algorithms tested reach a point where having a threshold almost resembles a random comparison.

Among the traditional techniques, SSIM exhibited the weakest performance. SSIM is only capable of detecting accurately near-duplicate images under very controlled conditions, as this strategy is highly sensitive to camera motion, viewpoint changes, and object displacement. As such, in scenarios involving image capture from a moving vehicle, even if the scene changes are not drastic, using this technique to infer near-duplicate status will produce worse results than both pHash and SIFT keypoint mask, techniques that are slightly more robust to viewpoint changes.

Dataset characteristics, particularly the number of images per distinct urban occurrence bag, are important to take into account when comparing these strategies with the control algorithm. As shown in Section 4.1.2, most urban occurrences span across multiple consecutive images, meaning that even aggressive removal from the control algorithm may still leave at least one representative

frame for many occurrences. Therefore, in scenarios where the traditional algorithms are performing better than the control algorithm, bellow the 80% image removal rate, for the pHash and SIFT Keypoint Mask, and bellow 50% for SSIM, observed differences between occurrence retention rate are still moderately low as seen in Figure 5.3.

In contrast, if the dataset was characterized by urban occurrences with low temporal persistence, being where distinct urban occurrences bags have at most three images, as shown in Figure 5.5, traditional algorithms perform worse while keeping a similar behavior as before. However the control algorithm ends up being far less competitive, as simply blindly removal becomes significantly less effective.

## 6.2 Deep Learning Similarity Techniques

As presented in Section 5.3, deep learning strategies based on object detection exhibited less consistent performance across the different models evaluated. While Grounding Dino and Florence generally surpassed the control algorithm, YOLOv13 and OWL-ViT models consistently performed worse than blind image removal.

Both Grounding Dino and Florence-2 generally present better results when compared to the control algorithm, particularly under strict comparison strategies. However, once the comparison strategy is less strict, these models end up converging towards the control algorithm results. Grounding Dino remained a decent performer, as it was able to detect near-duplicate images even in scenarios where object detection errors would occur. In such cases, the same incorrectly detected objects would be present in both images, resulting in a similar visual footprint despite the flawed reasoning.

Florence-2 exhibited highly variable behavior across the different configurations evaluated, with distinct failure modes depending on the prompting strategy. When prompting Florence-2 solely with *OD* task, the model would tend to detect only the most common object classes, and consequently failed to detect sufficient semantic detail to accurately identify near-duplicate images. When prompting Florence-2 with *CAPTION_TO_PHRASE_GROUNDING*, with either the urban objects query or the occurrences objects query, the model would heavily attempt to detect all prompted objects classes regardless of their presence in the image, resulting in extensive hallucinations. Even so, with this strategy the model was still be able to correctly detect some near-duplicate images, in scenarios where there was no viewpoint shift, as it would create two identical but incorrect visual footprints for consecutive images.

The YOLOv13 variants also failed to achieve strong results, which is expected given the use of out-of-the-box pre-trained models. Due to the COCO training dataset, YOLOv13 lacks several urban specific classes. Consequently, since the pretrained label does not align with the complete and distinct number of classes present in the urban environment, the detector fails to capture some relevant elements of the scene, not being able to create a complete semantic view of such. This ends up limiting the effectiveness in detecting near-duplicate images, and as such, comparisons based on detected objects and bounding boxes may not reflect the true semantic redundancy.

OWL-ViT consistently underperformed across the evaluated strategies, while also consistently being the weakest performer across all evaluated deep learning approaches, always yielding retention rates worse than blind removal. Although OWL-ViT is designed as a zero-shot detector, the model, due to being trained on the COCO training dataset, shares the same limitations as the YOLOv13 variants. Additionally, the model prioritizes matching all objects in the prompt, usually creating hallucinations, incorrectly identifying multiple objects. The combination of incomplete class coverage and excessive false positives, were the reason for OWL-ViT consistently achieving poor performance.

Across all evaluated object detection models, prompting with either urban objects or occurrences objects, resulted in similar results for both image removal rate and occurrence retention rate. However, the occurrences object prompts generally produced slightly worse results. This small difference can mostly be explained by the nature of urban occurrences, which are not solely defined by individual objects, but also by broader contextual cues, unrecognizable by the evaluated models. Furthermore, the intended strategy in detecting objects across the image was to create a visual footprint. As such, by restricting detection to a limited set of occurrence objects, the resulting visual footprints end up lacking information, leading to weaker similarity comparisons.

Nonetheless, both configurations involving object detection underperformed across the evaluated dataset due to shared limitations present in all evaluated models. None of the models were capable enough to detect all relevant objects under real world conditions, where factors such as partial occlusion, lighting variation, motion blur, and viewpoint changes are prevalent. These factors caused frequent fluctuations in the detected object sets between two consecutive images. Any attempt to mitigate these effects by relying on the use of a relaxed and weighted matching approach, as a mean to exclude incorrectly detect objects, would inadvertently also make the model less susceptible to actual non near-duplicate images, removing important information from the dataset, as observed in the *Relaxed and Weighted* approach.

Across all evaluated deep learning object detection models, the methodology inherently relies on identifying objects in the scene. This enables creating visual representations of the comparison results, helping illustrate why this strategy underperforms. For instance, in some comparisons the model may detect only a low number of objects in the image pair (Figure 6.1), clearly insufficient information to reliably assess near-duplicate status. In other cases, the model may detect the same incorrect objects in both images, producing a similar visual footprint. Even if this leads to a correct near-duplicate decision, it is due to incorrect semantic reasoning (Figure 6.2). The examples provided, while focused on Grounding DINO, were present in all evaluated object detection models, and as such should be taken into consideration when implementing similarity detection strategies based on creating visual footprints.

In Figure 6.1, only three objects were detected in both images, those being the road, sky, and a single tree. Clearly not enough content to give the actual visual footprint present in the image. In Figure 6.2, while in both images the same objects were detected, and correctly identified the two images as near-duplicates, the object detection model incorrectly identified a road sign for a person. Giving an incorrect semantic representation of the scene.




Figure 6.1: Grounding DINO example - low number of detected objects.




Figure 6.2: Grounding DINO example - incorrectly detected objects.

Despite the limitations of object detection methods, not all deep learning approaches exhibited poor performance. CLIP embedding method was able to consistently achieve strong results across

multiple image removal rates. By representing images as embeddings rather than relying on object detections, CLIP demonstrated increased robustness to minor viewpoint changes and scene variations, addressing most of the challenges that were present across all the object detection models evaluated.

When compared to pHash and the SIFT keypoint mask, the best performing traditional similarity techniques, the CLIP embedding approach consistently outperformed both methods under the global evaluation methodology (Figure 5.11). It achieved slightly higher occurrence retention rates across typical image removal targets, while maintaining substantially better retention under extreme image removal conditions.

Under the evaluation methodology focused exclusively on urban occurrences composed of at most three images (Figure 5.13), all similarity strategies exhibited largely overlapping occurrence retention rates across most image removal levels. However, CLIP embeddings remained more competitive at extreme image removal rates, where this strategy was able achieved higher retention than the traditional techniques.

As with the traditional techniques discussed previously, it is important to take into consideration the competitiveness of the control algorithm that is strongly influenced by the occurrence bag distribution. When most occurrences persist across many frames, this method has a high probability of leaving at least one representative image per occurrence. However, in dataset where all urban occurrences were characterized by having shorter temporal persistence, having only occurrences spanning across three images at most, as seen in Figure 5.9 and Table 5.6, the relative performance of all deep learning techniques improves compared to the control strategy. Nonetheless, the previously identified limitations of object detection remain, with CLIP embeddings consistently being the most competitive approach across both dataset scenarios.

## 6.3 Runtime and Deployment Evaluation

The runtime evaluation highlights a clear divide between traditional and deep learning approaches when considering edge deployment. Traditional techniques exhibited acceptable execution times on both the edge device and the server system. This outcome is expected, as these techniques are comparatively lightweight and can be implemented efficiently without specialized acceleration. As such, these methods are all viable for on-device near-duplicate identification.

Across the traditional methods, execution times on the server system were generally three times faster than those observed on the edge device. An exception to this trend was the SIFT keypoint mask approach, exhibiting a higher number of runtime outlier results. This behavior can be attributed to how the calculated SIFT keypoints are matched. Since these keypoints are matched under a brute-force algorithm, as the number of detected keypoints increases, the matching complexity grows rapidly, leading to an exponential increase in runtime. While this effect was mitigated on the server system due to higher computational resources, it was more pronounced on the edge device.

Most deep learning-based approaches, however, were significantly constrained on the edge device. The absence of CUDA (Compute Unified Device Architecture) acceleration on the Raspberry Pi makes models such as Grounding Dino and Florence impractical for edge deployment. Although OWL-ViT and YOLO variants achieved comparatively better runtime, their near-duplicate detection accuracy was not competitive in this study.

Among all evaluated deep learning approaches, CLIP embedding was the only technique that achieved a favorable balance between runtime and near-duplicate detection performance. This technique was able to perform pairwise image comparisons in under one second, with adequate device temperatures and decent energy consumption (with small spikes of energy), while also maintaining strong occurrence retention across multiple image removal rates, making it the only deep learning approach evaluated suitable for edge deployment.

During the initial design phase of this study, hybrid strategies combining traditional and deep learning techniques were considered. With the assumption that the traditional methods would perform quicker and mostly detect easy identifiable near-duplicate images, while deep learning approaches would handle more complex cases at higher computational cost. However, the experimental results invalidated this assumption. Traditional techniques often outperformed deep learning object detection methods, both in occurrence retention rate and runtime, while achieving lower device temperatures, and lower energy consumption. Therefore, rendering such hybrid strategies ineffective. As a result, this combined approach was discarded, as no benefit would be gained from sequentially applying faster and more accurate methods followed by slower and less effective models.

The sole exception to this previous stated observation was near-duplicate detection using CLIP embeddings. Nevertheless, the differences in image removal rate and occurrence retention rate between CLIP embeddings and both perceptual hashing or SIFT Keypoint Mask were insufficient to justify further exploration of stated strategy.

# Chapter 7

# Conclusions and Future Work

Over the past years, the *Theia Vision Project* has been collecting urban imagery through edge devices deployed across multiple users, with the objective of improving urban living conditions. After each capture session, images are sent to a centralized database, where an occurrence detection algorithm identifies issues that require intervention. To guarantee robust occurrence detection, close to five images are captured each second, creating excessive redundant data. This study addressed the challenge of reducing such redundancy by developing and evaluating techniques that are able to identify and remove these near-duplicates images, while preserving the total number of occurrences on the existing dataset. For this purpose, a representative subset of 80 000 images was selected from the complete dataset.

Across the evaluated dataset, characterized by normal driving conditions and captured from a vehicle dashboard, the results indicate that traditional similarity techniques, particularly perceptual hashing and the SIFT keypoint mask approach, provide great performance under realistic image removal targets. These techniques were both able to retain all occurrence information with an image removal rate close to 50%. In more aggressive image removal targets, pHash achieved an occurrence retention rate of 89.32% with an image removal rate of 82.21%, while SIFT keypoint mask achieved an occurrence retention rate of 94.82% when removing 74.62% of the images in the evaluated dataset. These methods consistently outperform blind removal at moderate strictness, are computationally efficient, and remain feasible on constrained edge hardware such as the Raspberry Pi 5.

Deep learning approaches based on object detection did not meet expectations under out-of-the-box configurations. Grounding DINO and Florence-2 showed potential at conservative image removal rates, being able to retain respectively 98.78% and 97.87% of the occurrence information when using strict comparison methods, while achieving an image removal rate of 48.68% and 45.60%. However, their retention decreased rapidly as the desired removal rate increased.

It is also important to note that, even in scenarios where object detection techniques correctly identified image pairs as near-duplicate, without reducing the occurrence retention rate, the similarity decision was not always reliable. Identification of two images as near-duplicates could have resulted from the incorrect detection of the same objects in both images, producing similar visual

footprints despite flawed object detection. As a result, good occurrence retention performance does not necessarily imply robust semantic understanding.

Among all deep learning strategies evaluated, CLIP image embedding was the most effective approach. CLIP was able to retain 100% of urban occurrences while achieving an image removal rate of 48.60%, and maintained occurrence retention of 91.46% even at an image removal rate of 86.93%. Furthermore, CLIP was the only deep learning method that demonstrated both strong similarity performance and runtime feasibility across both on server and edge device, making it one of the more compelling solutions for near-duplicate detection in the urban environment.

Finally, this study highlighted the strong influence of dataset characteristics on algorithm performance. In datasets characterized by urban occurrences with long temporal persistence, blind image removal by a fixed factor of up to four times can still achieve good occurrence retention while being the most computationally efficient strategy, as it requires no similarity computation and exhibits almost instant execution. However, in scenarios where urban occurrences exhibit low temporal persistence, blind removal performs poorly, clearly highlighting the actual impact and advantages of more robust approaches.

## 7.1 Future Work

Several directions for future research emerge from the findings of this study. First, once the occurrence detection model is fully integrated into the processing pipeline, it should be deployed on the dataset to automatically infer urban occurrences. Manual intervention may still be retained during the grouping phase of detected occurrences. This integration would enable a reevaluation of the proposed similarity techniques using the same evaluation methodology, while better reflecting the deployment conditions.

Second, although edge deployment was evaluated using a Raspberry Pi 5, future studies should explore the performance of deep learning approaches on edge devices equipped with CUDA capable GPUs. Platforms such as NVIDIA Jetson devices may allow the most promising deep learning technique, CLIP embeddings, to achieve significantly improved runtime performance, potentially widening the gap between traditional and deep learning techniques.

Additional future work may include fine-tuning object detection models on urban specific datasets, exploring how this adaptation influences near-duplicate detection performance.

# References


[1] Association for Computing Machinery. *ACM Digital Library*. https://dl.acm.org. Accessed May 2025. 2025.

[2] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

[3] Canonical Ltd. *Ubuntu 24.04 LTS*. https://ubuntu.com/download/desktop. Operating system, Long Term Support release. 2024.

[4] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: https://arxiv.org/abs/1704.04861.

[5] Institute of Electrical and Electronics Engineers. *IEEE Xplore Digital Library*. https://ieeexplore.ieee.org. Accessed May 2025. 2025.

[6] Yash Jakhar and Malaya Dutta Borah. "Effective near-duplicate image detection using perceptual hashing and deep learning". In: *Information Processing & Management* 62.4 (2025), p. 104086. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2025.104086. URL: https://www.sciencedirect.com/science/article/pii/S0306457325000287.

[7] Junjun Jiang Jiayi Ma Xingyu Jiang and Junchi Yan. "Image Matching from Handcrafted to Deep Features: A Survey". In: *International Journal of Computer Vision* 72 (2020), pp. 23–79. ISSN: 1573-1405. DOI: 10.1007/s11263-020-01359-2.

[8] Mengqi Lei et al. "YOLOv13: Real-Time Object Detection with Hypergraph-Enhanced Adaptive Visual Perception". In: *arXiv preprint arXiv:2506.17733* (2025).

[9] Xuan Li, Liqiong Chang, and Xue Liu. "CE-Dedup: Cost-Effective Convolutional Neural Nets Training based on Image Deduplication". In: Cited by: 5; All Open Access, Green Open Access. 2021, pp. 11–18. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00017. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85124120273&doi=10.1109%2fISPA-BDCloud-SocialCom-SustainCom52081.2021.00017&partnerID=40&md5=c03ec15b39ee25790e3e52aae1b7cb01.

[10] Shilong Liu et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *arXiv preprint arXiv:2303.05499* (2023).

[11] Yuhang Liu et al. “PV-YOLO: A lightweight pedestrian and vehicle detection model based on improved YOLOv8”. In: *Digital Signal Processing: A Review Journal* 156 (2025). Cited by: 1. DOI: 10.1016/j.dsp.2024.104857. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85208768982&doi=10.1016%2fj.dsp.2024.104857&partnerID=40&md5=7329323189a0292900c0db1b78634071.

[12] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[13] Matthias Minderer et al. “Simple Open-Vocabulary Object Detection with Vision Transformers”. In: *arXiv preprint arXiv:2205.06230* (2022).

[14] Mohd Hasan Mohiuddin and Latha Tamilselvan. “IDedupNet: A MobileNetV3-Based Deep Learning Framework for Efficient Image Deduplication in Cloud Computing Environments”. In: *Informatica (Slovenia)* 49.13 (2025). Cited by: 0; All Open Access, Gold Open Access, pp. 143–162. DOI: 10.31449/inf.v49i13.7514. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-86000348852&doi=10.31449%2finf.v49i13.7514&partnerID=40&md5=39f0bcf090621da006386dfe20c583fc.

[15] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)* (2021).

[16] M. M. Mahabubur Rahman et al. “Image deduplication using efficient visual indexing and retrieval: optimizing storage, time and energy for deep neural network training”. In: *Signal, Image and Video Processing* 18.12 (2024). Cited by: 1, pp. 9495–9503. DOI: 10.1007/s11760-024-03562-x. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85204108181&doi=10.1007%2fs11760-024-03562-x&partnerID=40&md5=9f9b25e6f9841b5e52c059a134267bfa.

[17] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[18] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[19] *ScienceDirect*. https://www.sciencedirect.com. Accessed May 2025. 2025.

[20] *Scopus*. https://www.scopus.com. Accessed May 2025. 2025.

[21] Srinidhi Sundaram et al. “Modelling of Firefly Algorithm with Densely Connected Networks for Near-Duplicate Image Detection System”. In: Cited by: 0. 2023, pp. 66–72. DOI: 10.1109/ICSCNA58489.2023.10370117. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85183054157&doi=10.1109%2fICSCNA58489.2023.10370117&partnerID=40&md5=19da077923820ac97b26a66df556f1

[22] K.K. Thyagharajan and G. Kalaiarasi. “A Review on Near-Duplicate Detection of Images using Computer Vision Techniques”. In: *Archives of Computational Methods in Engineering* 28.3 (2021). Cited by: 40, pp. 897–916. DOI: 10.1007/s11831-020-09400-w. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85077526631&doi=10.1007%2fs11831-020-09400-w&partnerID=40&md5=ab98048b0b9aa206fcaacc3b18b713b3.

[23] Priya Vij and Dalip. "An Deduplication Model to Enhance Image Retrieval Efficiency Through Hashing and Deep Learning (IDME-IR)". In: *2024 Second International Conference on Advanced Computing & Communication Technologies (ICACCTech)*. 2024, pp. 126–131. DOI: 10.1109/ICACCTech65084.2024.00030.

[24] Bin Xiao et al. "Florence-2: Advancing a unified representation for a variety of vision tasks". In: *arXiv preprint arXiv:2311.06242* (2023).

[25] Linsong Xiao et al. "EMG-YOLO: An efficient fire detection model for embedded devices". In: *Digital Signal Processing* 156 (2025), p. 104824. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2024.104824. URL: https://www.sciencedirect.com/science/article/pii/S1051200424004494.

# Appendix A

# Appendix: Edge-Device Metrics

The first Section of the appendix presents the metrics collected on the edge device for all evaluated algorithms not included in the main results chapter. The second Section presents comparison metrics for selected algorithms (perceptual hashing, Florence-2, and CLIP) between deploying the algorithms on the edge device, with and without active cooling.

## A.1 Edge-Device Metrics for Remaining Algorithms

The following Figure A.1, presents respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for SSIM on the evaluated edge device.

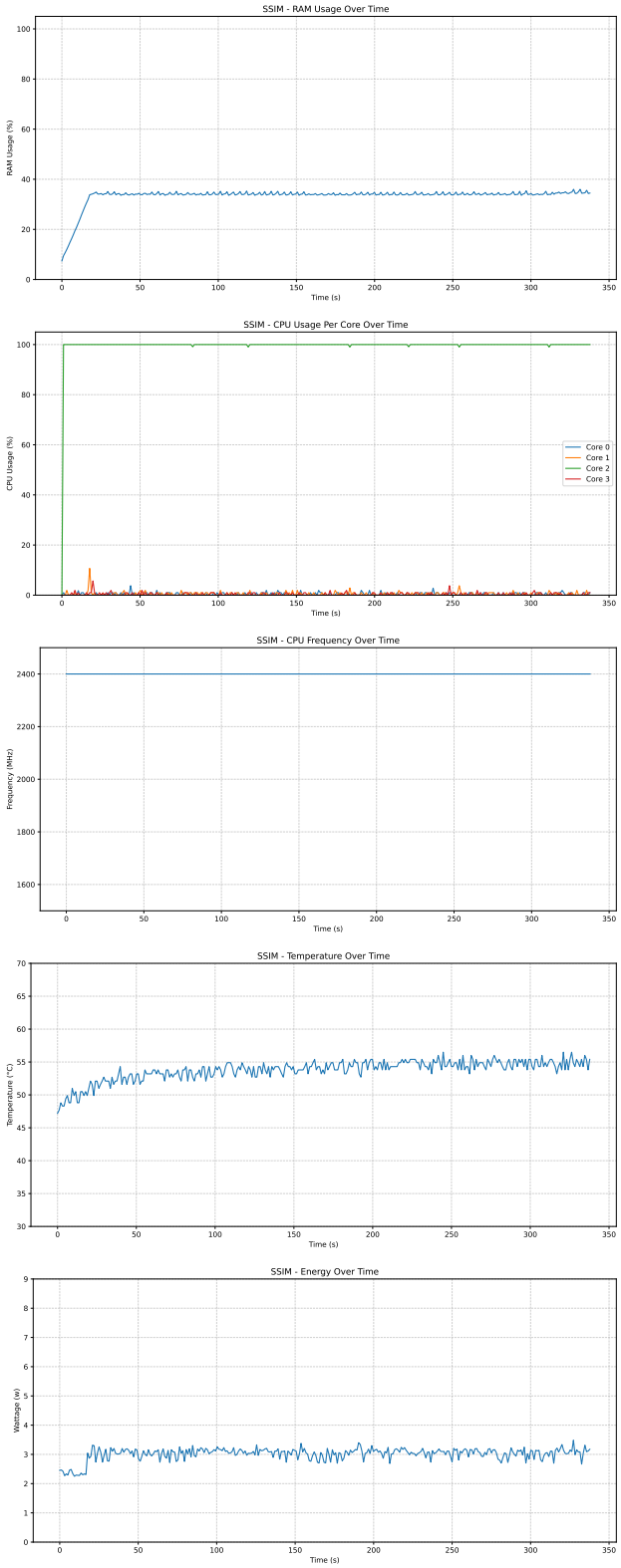



Figure A.1: SSIM metrics - edge device.

The following Figure A.2, presents respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for Florence OD on the evaluated edge device.

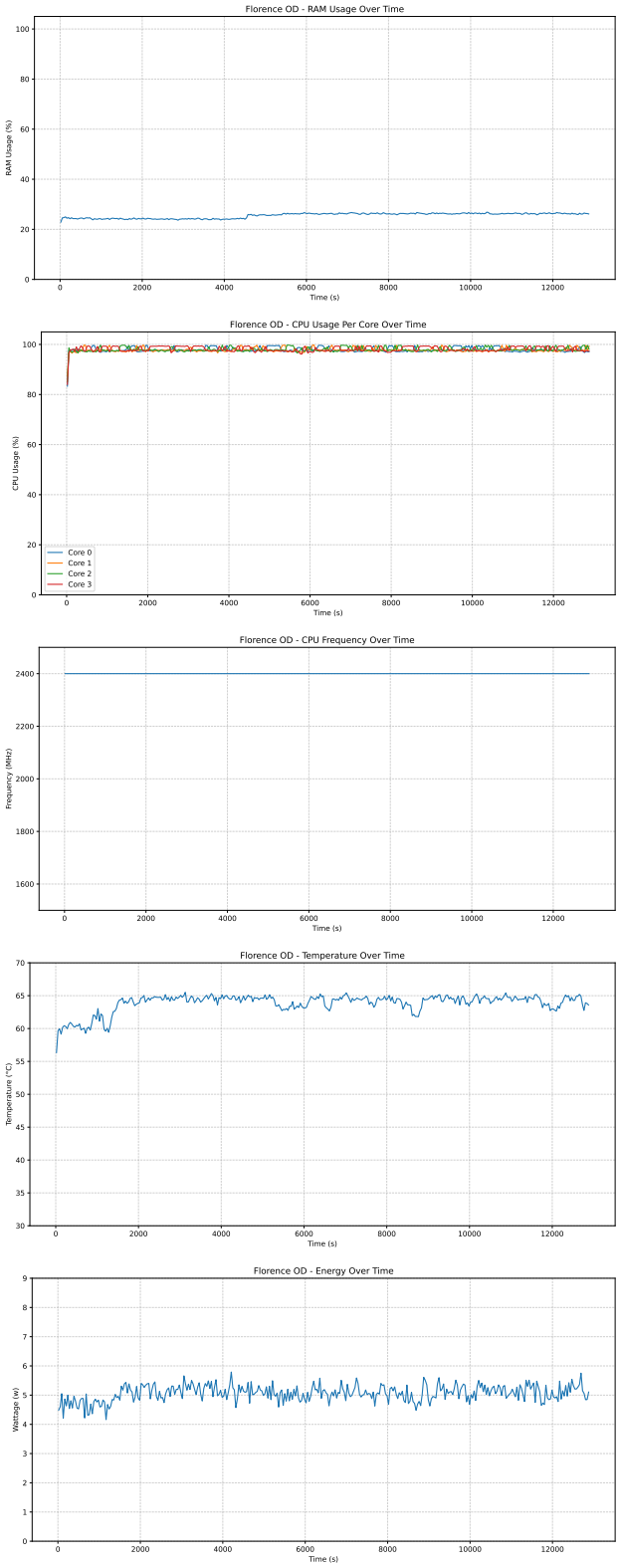



Figure A.2: Florence OD metrics - edge device.

The following Figure A.3, presents respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for OWL ViT on the evaluated edge device.

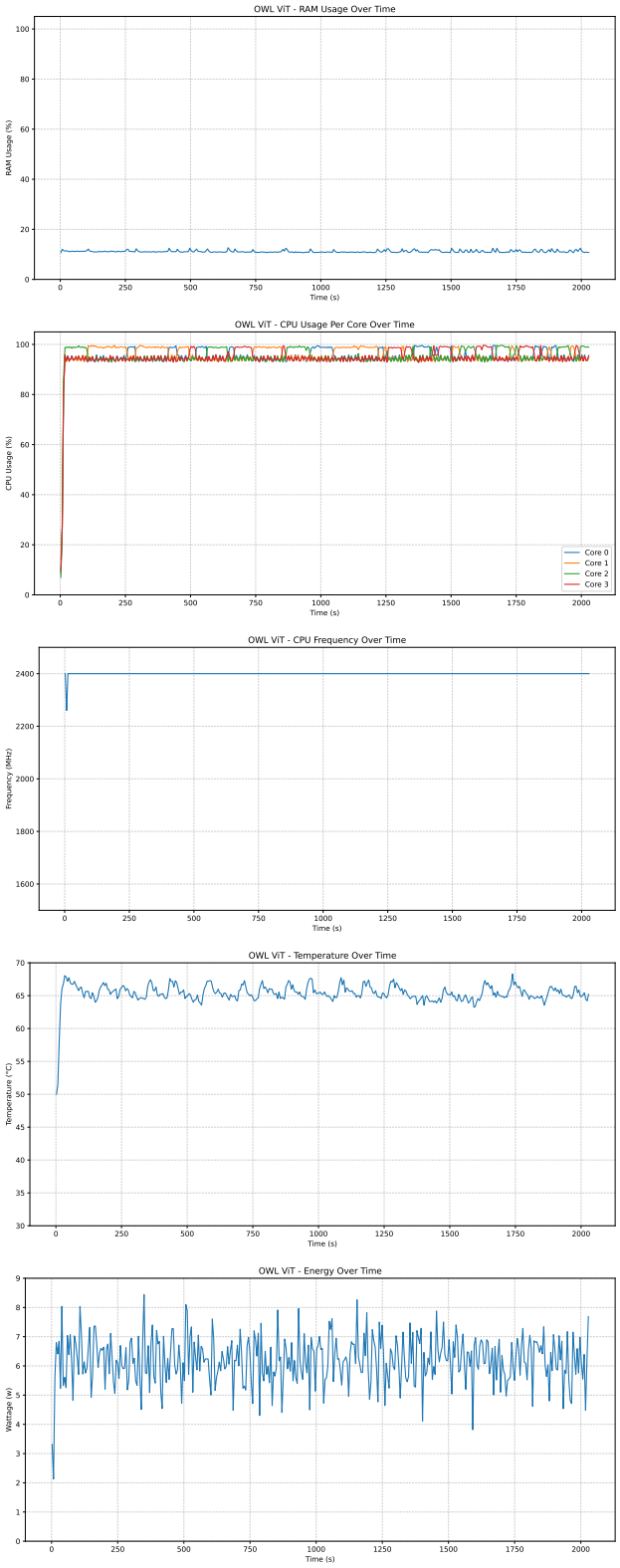



Figure A.3: OWL ViT metrics - edge device.

The following Figure A.4, presents respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for YOLOv13 n on the evaluated edge device.

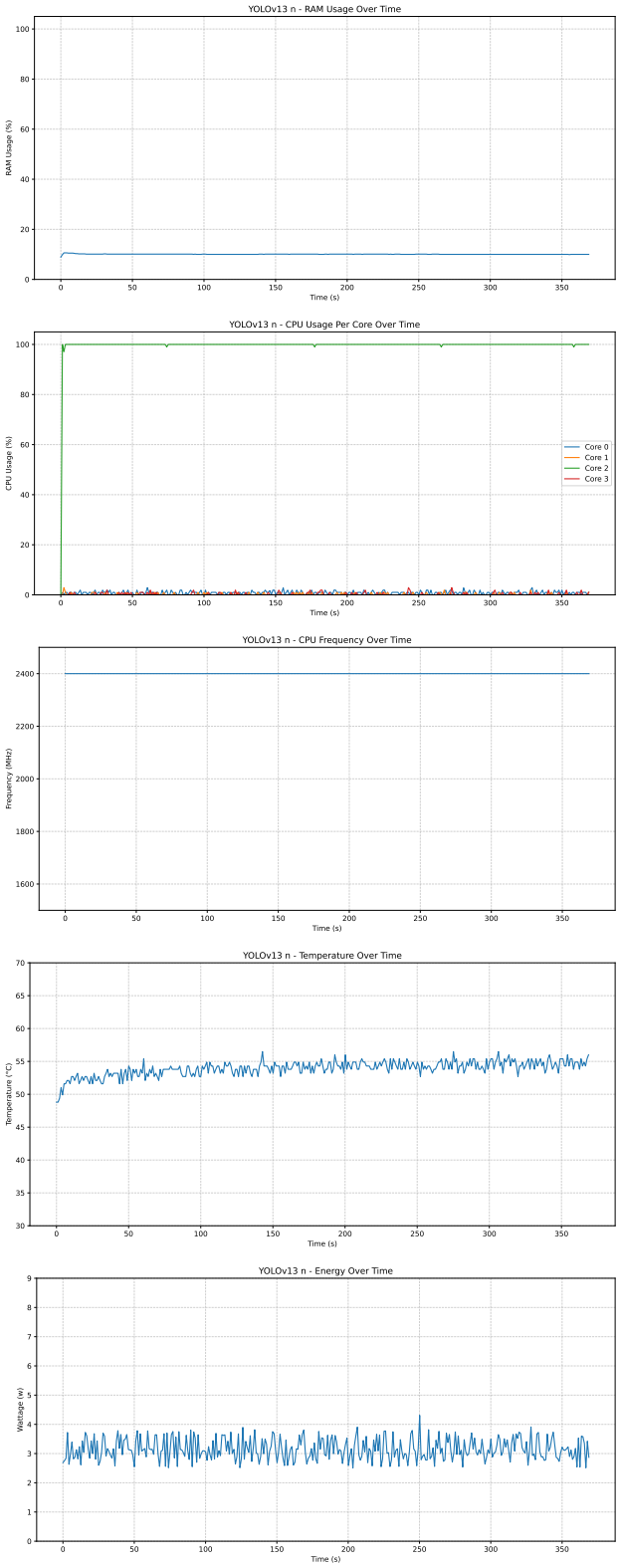



Figure A.4: YOLOv13 n - edge device.

The following Figure A.5, presents respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for YOLOv13 s on the evaluated edge device.

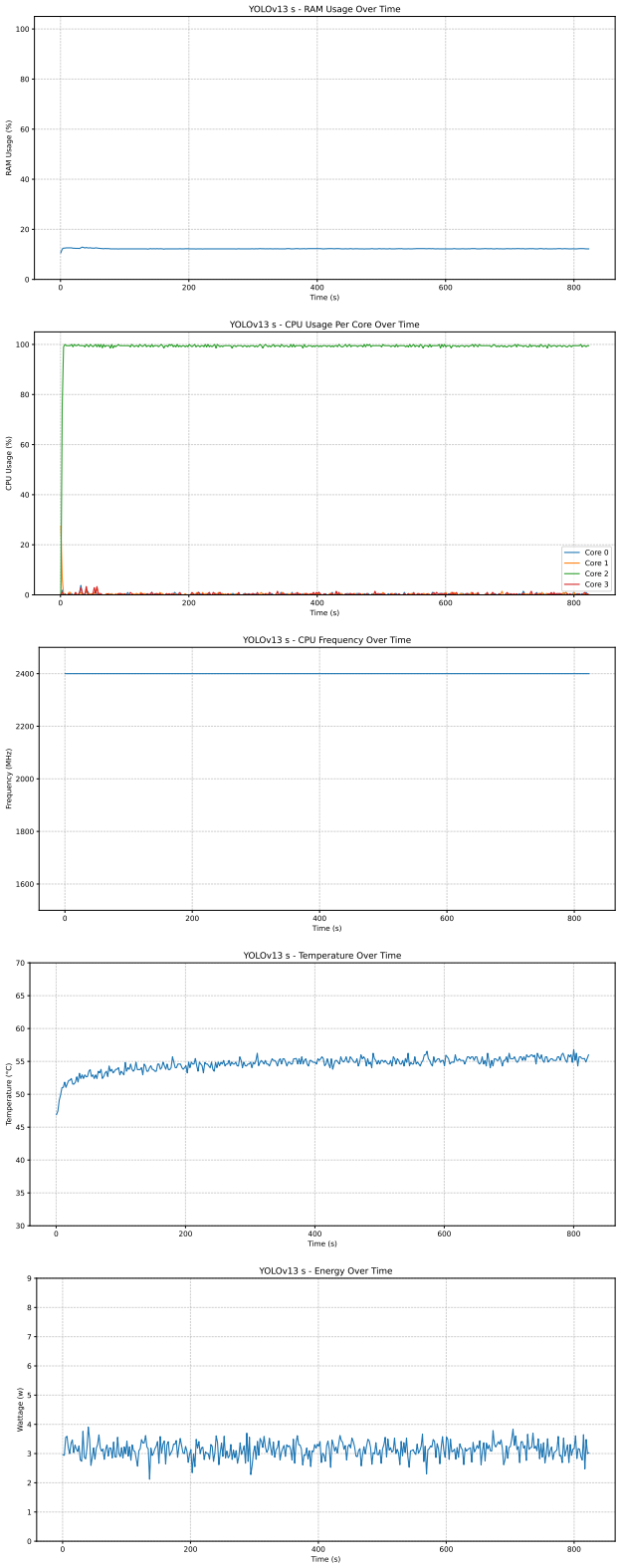



Figure A.5: YOLOv13 s - edge device.

The following Figure A.6, presents respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for YOLOv13 l on the evaluated edge device.

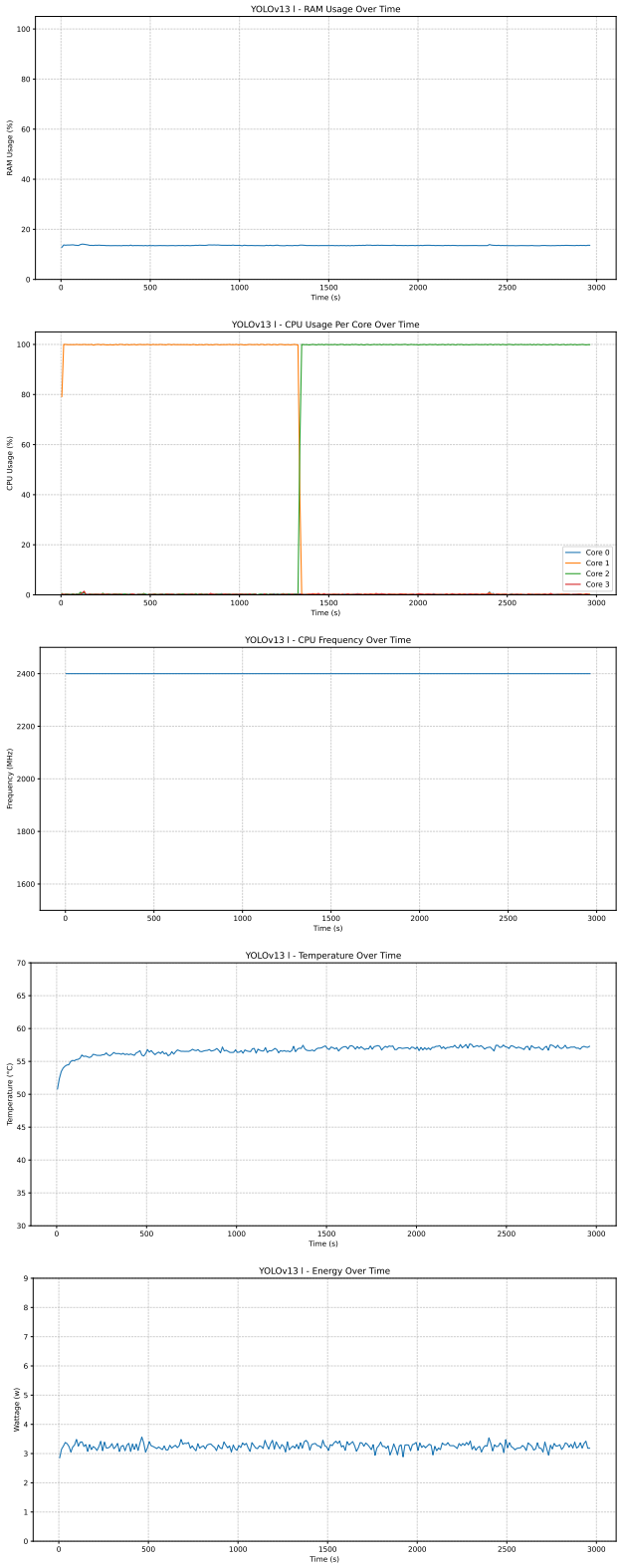



Figure A.6: YOLOv13 l - edge device.

The following Figure A.7, presents respectively, RAM usage, CPU usage, CPU frequency, device temperature, and energy consumption for YOLOv13 x on the evaluated edge device.

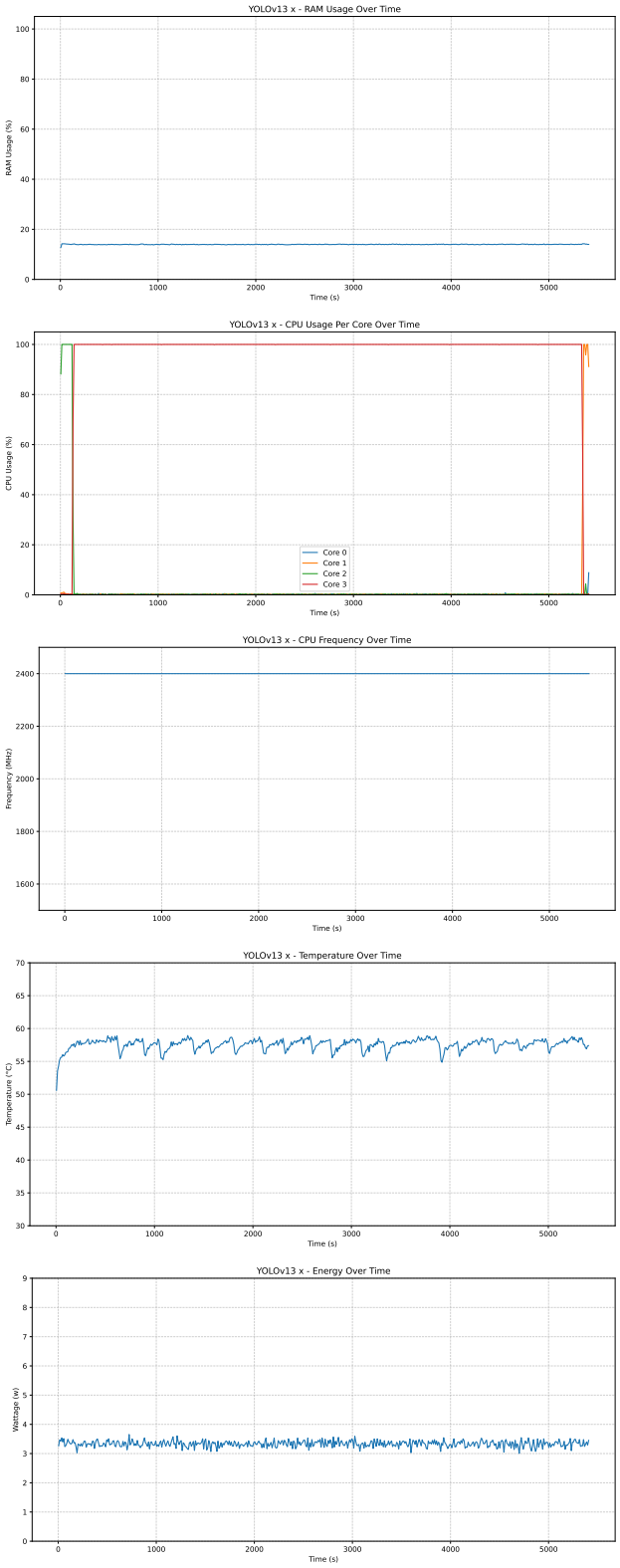



Figure A.7: YOLOv13 x - edge device.

## A.2 Edge-Device Active Cooling Results

The following Figure A.8, presents the same previously analyzed metrics for Perceptual Hashing on the evaluated edge device, when not leveraging active cooling.

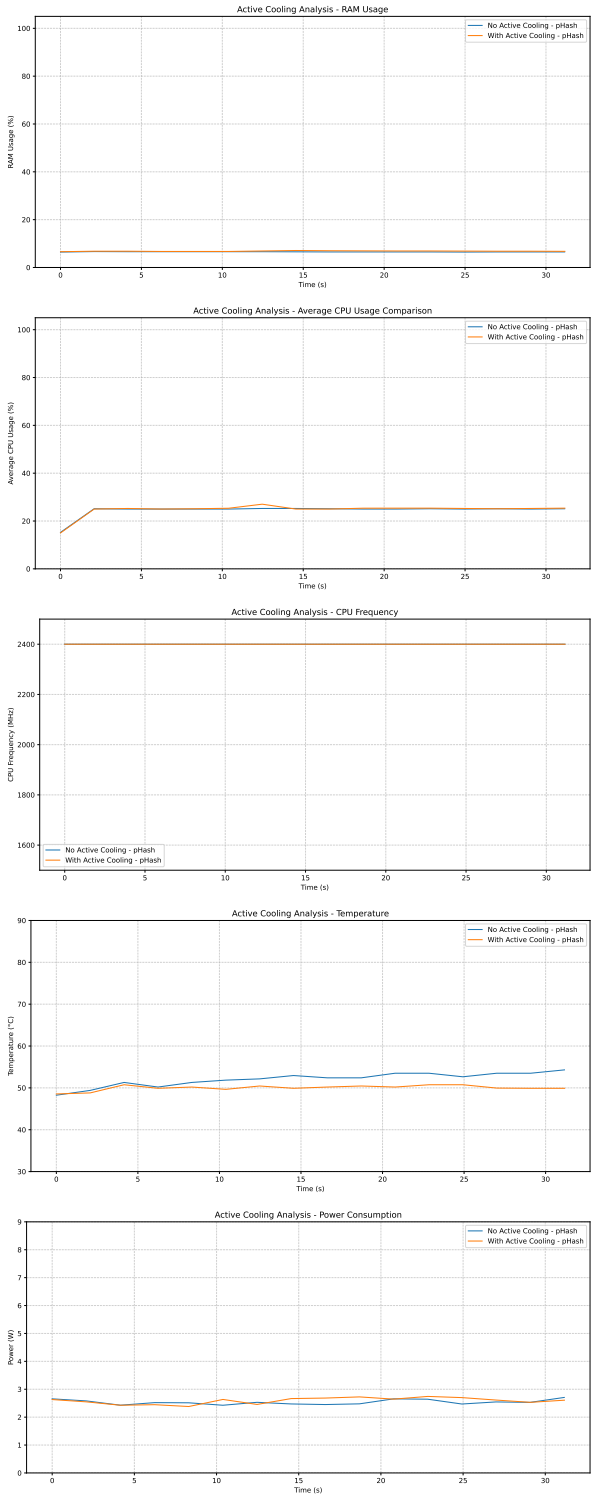



Figure A.8: Perceptual Hashing active cooling - edge device.

The following Figure A.9, presents the same previously analyzed metrics for CLIP embeddings on the evaluated edge device, when not leveraging active cooling.

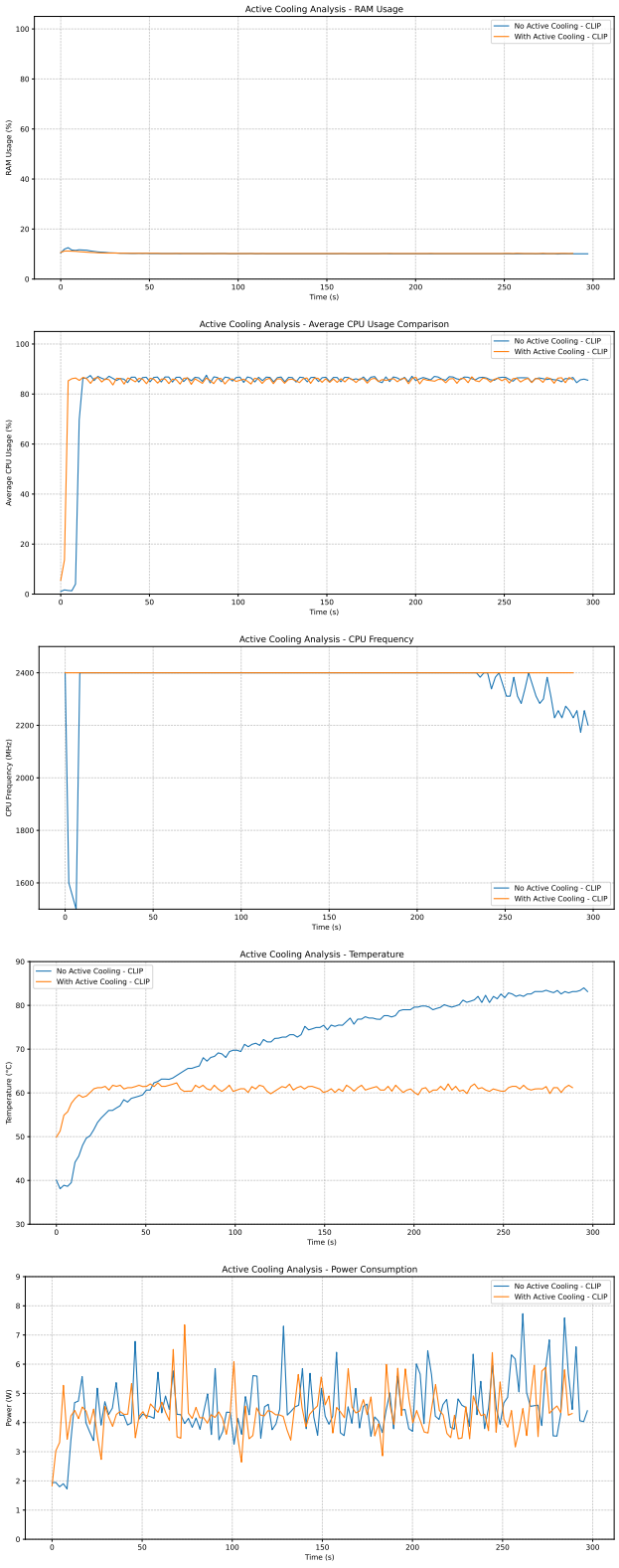



Figure A.9: CLIP active cooling - edge device.

The following Figure A.10, presents the same previously analyzed metrics for Florence 2 OD on the evaluated edge device, when not leveraging active cooling.

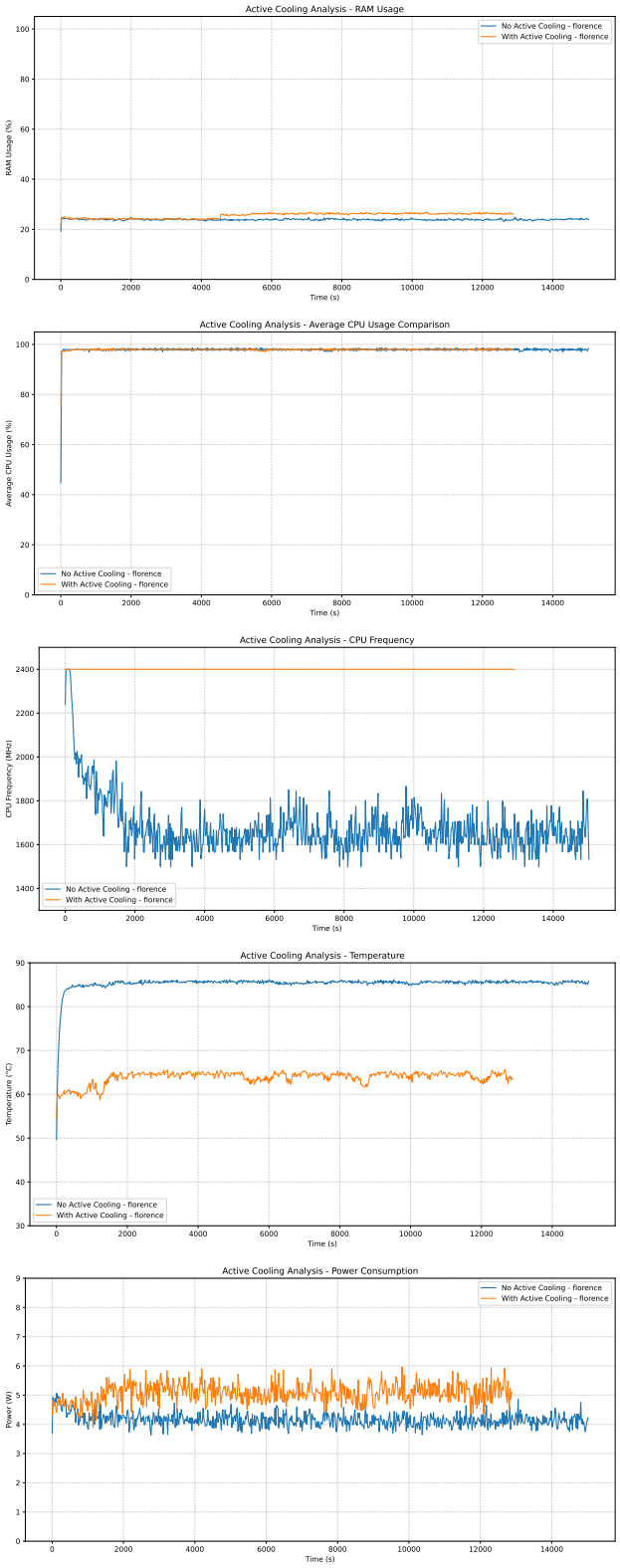



Figure A.10: Florence 2 OD active cooling - edge device.