UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

# Stream Based Active Distillation and Holonic Architecture for Scalable Camera Systems

Dani Manjah

Thesis submitted in partial fulfillment
of the requirements for the degree of
*Docteur en sciences de l'ingénieur*

Dissertation committee:

Prof. Benoit Macq (UCLouvain, advisor)
Prof. Stéphane Galland (UTBM, advisor)
Prof. Christophe Devleeschouwer (UCLouvain)
Prof. Manuel Kolp (UCLouvain)
Prof. Frédéric Dufaux (Supélec, France)
Prof. Marc Van Droogenbroeck (ULiège, Belgium)
Prof. David Bol (UCLouvain, chair)

Version of November 8, 2024.

**Acknowledgements**

The beginning of this thesis finds its roots in 2019. Back then, I was in consultancy and had seen a job ad about a research project in *Multi-Agent Systems*. Honestly, I had no idea what an “Agent" was. However, a strong curiosity drove me to understand how Artificial Intelligence Agents can interact with each other. Little did I know that applying would set me on a five-year journey at the Pixels and Interactions Lab. At first, I was a researcher and wanted to gain some experience and sharpen my knowledge in artificial intelligence before returning to industry. Yet, two years later, my supervisor, Prof. Benoît Macq, suggested the idea of a thesis. He was the first person who believed in this journey, and I would like to express my deepest gratitude to him. He not only initially took a chance on me but also supported me through the highs and lows. He gave me the freedom, the space, and the unfailing mental and financial support I needed.

As my thesis was at the intersection of computer vision and multi-agent systems, Prof. Stéphane Galland, an expert in agents, joined in and became my second supervisor. Stéphane not only brought his top expertise but truly inspired me with his diligent passion for computer science and research. I would like to thank him also for the two times he hosted me at his lab in Franche-Comté. I am also very grateful to Prof. Christophe Devleeschouwer, an academic mentor who not only provided me with technical expertise but also instilled proper reasoning. He helped me correct flaws in my thinking and taught me valuable lessons that will surely help me in my career. I was also blessed with a supervision committee of highly competent yet benevolent Professors. I would like to thank Manuel Kolp, David Bol, Frédéric Dufaux, and Marc Van Droogenbroek for the numerous discussions we had. Needless to say, I left my private defense more knowledgeable thanks to them.

A thesis can feel pretty lonely and comes with a lot of self-doubt. Of course, no one understands this rollercoaster like your fellow PhD students, who share your turmoil and offer support. I was lucky to get to know Karim Elkhoury, who became not only a PhD buddy but a trusted friend and a supportive peer when I needed it the most. I also had the chance to befriend another colleague, Charles Wiame, whom I would like to thank for his numerous personal and professional pieces of advice. I would also like to thank my friend Nicolas Leblanc, who planted the seed for this journey by posting the aforementioned job ad, as well as Simon Demey, Thomas Pairon, and Jean Cavillot for the many interesting discus-

sions over delicious pasta lunches at “Dude". I was also fortunate to have engaging discussions on the Distributed Kalman Filter with Dr. Antoine Aspeel and Dr. Charles Monnoyer de Galland de Carnières, who, despite the lack of conclusive results, continued to engage and support me in this exploration. My co-author on two papers, Davide Cacciarelli, deserves thanks as well, not only for his ability to motivate with optimism but also for his talent for research. I am grateful to all my colleagues at PiLab with whom I shared so much throughout these years.

There is a saying that it takes a village to raise a child, and I seriously believe this metaphor applies to a PhD project and the PhD student. Therefore, I would like to thank my family, especially my mother, Mintaha Manjah, and my siblings, Sami and Sarah Manjah, who provided unconditional love and support. I am also thankful for my cousin, Yousef Hanon, with whom I shared so many enthusiastic conversations about AI. My uncle, friend, and career mentor, Fouad Manjah, has also been a vital support. Of course, friends are of utmost importance. Special thanks go to Sabrina Vandezande, who supported me relentlessly, not only emotionally but also by carefully reviewing my papers, and to Sandrine Imbreckx for her photography to help me market myself. I would also like to thank Arnaud Mary and Tristan Berghmans for their overall personal support.

Finally, my gratitude goes to my field hockey team, **Misfits Legends**. With them, I could not only practice my sport but also enjoy the company of a joyful group that made every training session, match, and "troisième" a perfect outlet for maintaining both a balanced body and mind.

Was it worth it? As I write these acknowledgments, I cannot help but realize how transformative this journey has been. In my personal opinion, engaging in a doctorate is not primarily about developing technical skills—those can, I believe, be achieved in an industry role. Rather, it’s the solitary nature of the project, the hardships, the moments of self-doubt, and the reality that no one truly comes to your rescue that challenge you and help you grow into a better person. So, it is a big YES!

**Abstract**

Image processing is essential in camera systems for applications like traffic analysis. However, deploying video analytics solutions poses challenges such as:

1. Continuous adaptation to the operational environment,
2. The system's expansion with new units, and
3. The need for context-specific solutions.

This thesis aims to address these challenges by ensuring economical management of retraining and system expansion without compromising performance.

Firstly, we reduce annotation and training costs per node by employing lightweight deep neural networks trained on selected samples from camera streams. We demonstrate that selecting frames with the highest confidence scores from local models produces the most accurate models. These samples are annotated using a large general-purpose model, minimizing the need for manual labeling. We also analyze potential biases introduced by model-based pseudo-annotations in comparison to human annotations.

To further reduce costs, we cluster cameras based on their similarity to train group-specific models. This decreases the number of required models and enhances local performance through a specificity-diversity trade-off. Indeed, groups can fine-tune models to capture the nuances of individual streams while leveraging broader data for better generalization. We also introduce an automatic clustering strategy based on the premise that models fine-tuned on specific camera domains can effectively transfer to similar domains.

Lastly, we design a holonic architecture for multi-camera and multi-method systems. By exploiting its self-contained, multi-scale, and hierarchical characteristics, we develop two key applications: (1) a multi-method framework that integrates deep neural networks with traditional image processing techniques to reduce processing costs without sacrificing quality, and (2) robust design principles that facilitate the expansion and contraction of camera systems.

Through these methodologies, this thesis advances the automation and adaptability of image processing in multi-camera systems, contributing to their scalability.

# Contents

# List of Figures

# List of Tables

# List of symbols

### C-SBAD Framework

$\mathcal{E}_k$ — The $k$-th cluster of training sets (p. 51)

map$(i)$ — Function mapping the $i$-th stream to its cluster (p. 51)

X — Video Stream (p. 36)

$X_i$ — The $i$-th video stream out of the total $N$ streams (p. 51)

$N$ — Total number of video streams (p. 51)

$\theta_i$ — Model associated to a stream $X_i$ (p. 51)

$B$ — Frame budget per *Student* (p. 37)

$K$ — Number of clusters or grouped training sets (p. 51)

$\mathcal{T}$ — Training data collected from the stream X (p. 37)

$\mathcal{T}_i$ — Training data collected from the $i$-th stream (p. 51)

$\Theta$ — *Teacher* model used for pseudo-labeling images (p. 37)

$\theta$ — Model associated to a stream (p. 36)

### `SELECT` Sampling Strategy

$\tilde{C}_t$ — Set of confidence scores associated to $\tilde{P}_t$ (p. 38)

$I_t$ — Image at timestamp $t$ (p. 36)

$\alpha$ — Selection rate for `SELECT` (p. 38)

$\Delta$ ____ Confidence threshold for a confidence-based `SELECT` (p. 38)

$\tilde{P}_t$ ____ Set of bounding boxes for an image $I_t$ (p. 37)

$w$ ____ Warm-up period in frames, for confidence-based `SELECT` (p. 38)

**Holonic Architecture**

$\hbar_i^l$ ____ The $ith$ holonic agent at a layer $l$ of an holarchy $H^L$ (p. 66)

$L$ ____ The number of layers in an holarchy $H^L$ (p. 66)

$l$ ____ Layer $\in [0, L]$ of an holarchy $H^L$ (p. 66)

$\hbar_+$ ____ Incoming holon (p. 74)

$\mathrm{X}_i^l$ ____ Set of operating data streams of a holon $\hbar_i^l$ (p. 66)

$\mathcal{H}_{\boldsymbol{O}}^L$ ____ Holarchy of $L$ vertical layers instantiating an organization $O$ (p. 66)

$\mathcal{T}_i^l$ ____ Training set of a holon $\hbar_i^l$ (p. 66)

$\mathcal{T}_+$ ____ Training set of the incoming holon $\hbar_+$ (p. 74)

$\mathcal{V}_i^l$ ____ Validation set of a holon $\hbar_i^l$ (p. 66)

$\mathcal{V}_+$ ____ Validation set of the incoming holon $\hbar_+$ (p. 74)

$\mathrm{SUB}_i^l$ ____ Inner members corresponding to layer $l-1$ of a holon $\hbar_i^l$ (p. 66)

$\theta_i^l$ ____ Processing model of a holon $\hbar_i^l$ (p. 66)

$\boldsymbol{O}$ ____ An organization of holons (p. 66)

$N_+$ ____ Number of new units (p. 76)

# Acronyms

**AL** Active Learning

**CSBAD** Clustered Stream-Based Active Distillation

**DNN** Deep Neural Network

**HMAS** Holonic Multi-Agent System

**HoL** Holonic Learning

**KD** Knowledge Distillation

**MAS** Multi-Agent System

**ML** Machine Learning

**SBAD** Stream-Based Active Distillation

**SBAL** Stream-Based Active Learning

**TTA** Test-Time Adaptation

**WALT** Watch and Learn Time-lapse

# Glossary

**batch** in machine learning, a batch is a subset of the training data. 13

**camera domain** characteristics of the data collected by the camera, including the underlying distribution of features that might change due to different conditions (*e.g.*, locations, lighting conditions, viewpoint, camera type,etc.). 49, 52

**catastrophic forgetting** refers to the tendency of an artificial neural network to abruptly and drastically forget previously learned information upon learning new information. 16

**complex system** systems characterized by a high number of components with strong interdependence among parts and with rich and diverse information. Due to these factors, complex systems may be difficult to analyze, model, or determine their behaviors and outcomes with certainty. 17

**confirmation bias** refers in semi-supervised or unsupervised learning to noise accumulation when the model is trained using incorrect predictions. 15

**epoch** a full pass of a machine learning model through the entire training dataset. 14

**fine-tuning** involves adjusting a model trained on a large dataset to perform well on a smaller, task-specific dataset. 14

**gradient descent** an optimization algorithm driving the learning process by updating the model's parameters in the direction that minimizes the loss, iteratively improving the model's performance. 13

**holarchy** is defined as a hierarchy of self-regulating holons. 23

**holon** agents with self-nested structure that can be seen, depending on the level of observation, either as an autonomous whole entity or as an organization of other holons. 23

**holonification** the process of grouping a scattered agent population into a holarchy. 24

**iteration** involves processing a batch of data, calculating the loss, and updating the model's weights. 13

**open system** a system subject to a dynamic composition with continuous addition or removal of units. 18

**organization** is a subsystem where agents play roles and interact together to achieve a shared goal in the context of this organization. 22

**role** is both an expected behavior to fulfill (part of) a requirement and a status to the role's agent in the organization. 22

**student** refers to a (lightweight) deep neural network which aims at mimicking the pattern recognition mechanisms of a larger *Teacher* model. 27

**teacher** refers to a large and general purpose deep neural network that transfers its learned knowledge to a smaller, more compact *Student* model by guiding it to mimic its output. 27

**technical debt** is the implied cost of future reworking because a solution prioritizes expedience over long-term design. 17

# 1 Introduction

## 1.1 Scalability Issues in Camera Systems

CAMERA systems provide a visual aid to support decision-making. These tasks might involve detecting objects (*e.g.*, unattended items, vehicles) or classifying image content (*e.g.*, assessing the extent of damage in disaster areas). To automatically extract meaningful features from visual data, camera systems rely on image processing methods such as Deep Neural Networks (DNNs). Previously conceived in closed settings, video analytics systems are expected to expand into vast networks, broadening their coverage and increasing their survivability against device defects or occlusions [8, 9]. Furthermore, practitioners resort to edge devices for processing, which are more portable and economical but have limited computational power. To mitigate expansion costs, this thesis proposes methodologies to streamline DNNs deployment and design recommendations for managing their growth.

DNNs' predictive power (*i.e.*, extracting features from an image) correlates with their size (*i.e.*, number of parameters) [10] and a large and diverse training database [11]. However, this means a significant investment of money and time. First, current practices rely on human workforce for data annotation, costing \$0.25 to \$7 per annotated image [12]. Nevertheless, this proves unsatisfactory as a single investment in a dataset is

unlikely to cover future operational contexts in case of additional cameras deployment [13]. Moreover, in practice, DNNs require periodic updates or adjustments when underperforming due to changes in the operating conditions [14, 15]. Second, large models require sophisticated hardware for training and inference, which increases hardware costs and limits their portability to edge devices.

> **Scalability Objective 1:** Diminish the annotation and training costs as well as model size. This streamlines (re)training, facilitating recurrent maintenance of DNNs as well as reducing the marginal cost of adding a camera.

Maintaining and monitoring multiple DNN models in large-scale networks can be challenging without careful system design [14]. Indeed, these models are subject to feedback loops, where data and interactions with the external world influence their behavior in unintended ways [14]. At the system level, third-party subsystems [9] can join or leave unpredictably, creating challenges in controlling the system's behavior. Systems also tend to be integrated as subsystems in a broader system, creating intricacies that could trigger disastrous chain effects [16]. Traditional software logic may be subtly inept because the desired evolving behavior of systems cannot be effectively expressed in scripted logic. Furthermore, we cannot assume the real world fits into tidy encapsulation, nor can we conceive single-method, single-purpose systems without creating a high technical debt [14, 17]. In other words, an expedient design would imply a future high cost to maintain and scale systems.

> **Scalability Objective 2:** Reduce refactoring efforts and facilitate isolated changes. The purpose is to simplify the addition of nodes and improve control over expanding multi-camera, multi-model systems. This is essential not only to prevent system collapse but also to create more flexible systems capable of seamlessly adapting to new configurations.

In this thesis, vehicle detection for traffic analysis serves as a practical use case for the proposed methodologies (illustration in Fig. 1.1). The remainder of the introduction presents our approaches.

**Fig. 1.1** Cameras at intersections tracking vehicles help traffic engineers in understanding journey times, optimizing signal timing, and reducing congestion [1]. This example and figure highlight the diversity in real-world contexts and expansion scope of camera networks.

## 1.2 Scalable Deep Neural Networks Deployment

To reduce training and annotation costs, we propose a system where cameras use lightweight DNNs suited to the visual specificity of their video streams. In this system, illustrated in Fig. 1.2, the central unit collects the images from the nodes, annotates them, and trains the DNNs to be deployed on the nodes. Since automatic management of the system is desired [18], annotation of the collected data is implemented using a large, general purpose *Teacher* model instead of human annotators. In practice, the *Teacher* may be inaccurate, potentially causing *Student* DNNs to overfit to incorrect pseudo-labels. This phenomenon is generally referred to as *confirmation bias*.

Obviously, the amount of data collected per stream and the number of *Students* directly affect maintenance costs and system scalability, raising two key questions:

**RQ1.** How can we select images from each stream to maximize the resulting model's accuracy?

**RQ2.** To what extent does the quality of pseudo-labels affect downstream models' performance? How can we mitigate the impact of inaccurate pseudo-labels and reduce the risk of confirmation bias?

To mitigate training complexity in large-scale systems, we propose to cluster streams based on their similarities and to train a single model for each cluster.

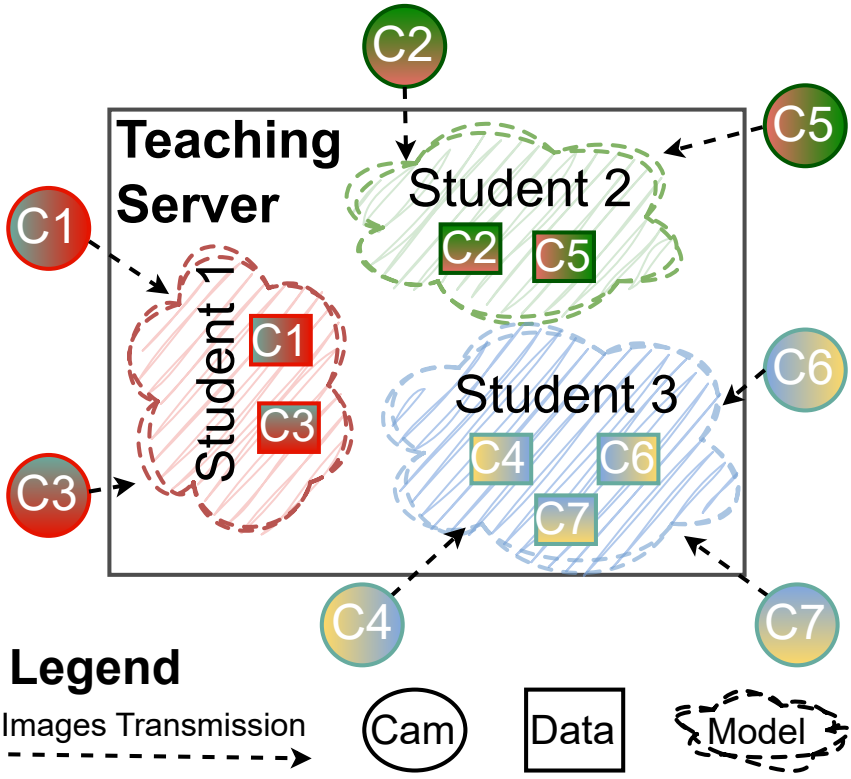



**Fig. 1.2** Updates of video analytics models: Cameras (C1 to C7) send selected image data to a central server, which pseudo-labels, trains, and updates specialized *Student* models for groups of similar cameras. These updated models are then sent back to their associated cameras. This thesis's contribution lies partly in the way each individual camera selects the images sent to the central server (see Section 4.2), and in the strategy adopted by the central server to aggregate the cameras into a set of clusters, each cluster leading to a specialized *Student* (see Section 5.2)

Choosing the right number of clusters is not trivial. Indeed,

- Fewer but larger clusters allow models to train on a broader diversity, enhancing generalization ability [11]. Moreover, they diminish the number of models to manage in the system.
- Conversely, numerous but smaller clusters fine-tune models to the nuances of individual streams, accounting for the **limited knowledge representation capacity** of lightweight DNNs [19].

Given a fixed number of training images per stream, the partitioning in clusters also calls for reevaluation of the training management. Indeed, in DNN training, the number of times a model updates its weights in one epoch (*i.e.*, a full pass of the model on the training set) depends on the amount of training images. Hence, with a fixed number of training epochs, models trained over smaller clusters update their weights less frequently. This disadvantages those models compared with those trained on larger clusters. Therefore, in scenarios where computational power is abundant, we could investigate increasing the number of epochs to reach the model's peak accuracy even when the training set gets smaller. In this context, we add two research questions:

**RQ3.** Given a constant number of epochs per model, thereby keeping the overall training complexity constant, which partitioning results in the highest model performance?

**RQ4.** Given a constant number of iterations (gradient descents) per model, which partitioning results in the highest model performance?

## 1.3 Multi-Camera and Multi-Method Systems

The previous section proposed methodologies that mitigate costs at the camera level and the system level, bringing a learning dimension to camera systems. To further push scalability, we consider here the design of a multi-camera and multi-method processing and learning system, illustrated in Fig. 1.3 and Fig. 1.4.

A proper design can reduce refactoring efforts and facilitate isolated changes, thereby streamlining the integration of a novel camera or model, and more broadly a subsystem of cameras and models [14, 17].

Based on the literature, we derived four recommendations to **design a scalable multi-method signal processing and learning system**:

1. Establish standardized interaction protocols, aggregation strategies, commitment, and communication patterns within components. This facilitates the integration of new units, as they only need to understand their role and communication methods within the system, irrespective of their operating mode [20, 21].
2. Render a method, a sensor, or by construction a subsystem as independent and self-contained, limiting the intricacy between units. This aims to simplify a local update or maintenance [21, 22, 23].
3. Recursively divide a system into subsystems based on a key criterion. This prevents the propagation of disruptions [20]. Furthermore, the integration of a new component requires less communication as it requires only coordination with upper system layers instead of with each subsystem [16].
4. Place units at certain levels of the hierarchy and provide representations of how other levels can contribute "information" or "models". This division simplifies programming complexity, allowing designers to focus on each module and facilitating reuse across different systems [16].

Holonic Multi-Agent System (HMAS) provides modelling tools to design a system with high-level **abstraction, autonomy, hierarchical organization and multi-scale modelling**. The core of HMAS is the "holon", a structure that is both self-contained and part of a larger system [24][1]. Applying such a whole-part relationship to different entities of a system creates a multi-level interconnected structure of holons, where the behavioral variations of **each holon at any level are characterized by operational variations of its subordinate holons at a lower level**. This recursive and autonomous nature is capable of describing complex hierarchical systems with various granularity levels [25, 26]. This thesis then proposes to address:

**RQ5.** Through the holonic paradigm, how can we design a scalable multi-method, multi-camera processing and learning system?

*Remark* 1.1. Section 2.3 provides a deeper introduction to holonic agents.

[1] For instance, a cell exists independently but also forms part of a human body, just as a human is independent but can be part of a work team.

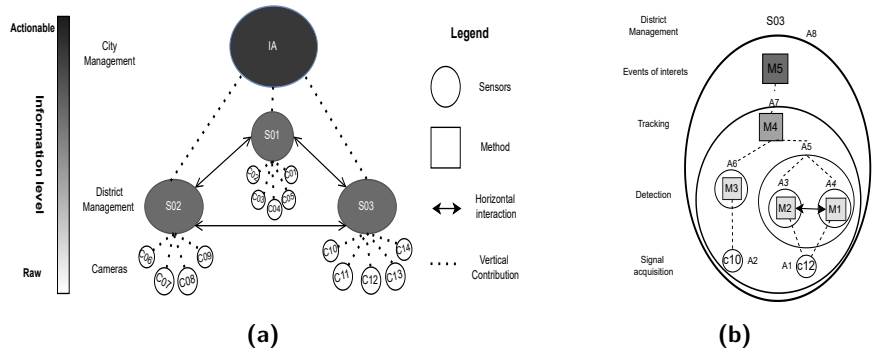



**Fig. 1.3** (Left) Fig. 1.3a illustrates a multi-level camera system. (Right) Fig. 1.3b depicts multi-layer image processing. These figures juxtapose the complex dynamics of multi-camera and multi-method systems and draw attention to the need for proper design to capture the complexity of these systems.

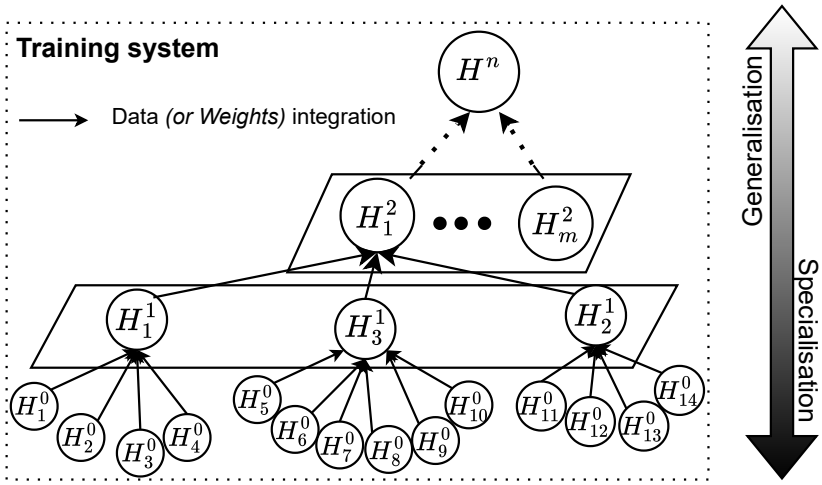



**Fig. 1.4** A large-scale training system is considered, where nodes consist of DNNs trained on specific datasets. Lower nodes are typically tailored to their task (*e.g.*, analytics on a sensor) and operationally more efficient. Higher nodes, trained over vaster and more diverse data, provide generalization ability. This system can scale up or down, causing integration and adaptability challenges.

## 1.4 Thesis Outline

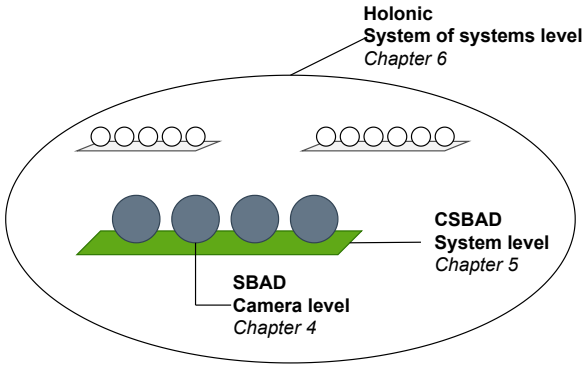



**Fig. 1.5** Each contribution chapter addresses a specific granularity level: Chapter 4 focuses on the camera level, Chapter 5 examines the system level, and Chapter 6 considers a holistic view of a system of systems.

This thesis is organized as follows:

- **Chapter 2** introduces the key concepts and methods central to this thesis. It provides an in-depth analysis of holonic agents, explores the challenges in system design, and covers basic topics in image processing and machine learning.

- **Chapter 3** offers a comprehensive literature review and identifies the gaps that our methodologies aim to address.

- **Chapter 4** aims to reduce annotation and training costs per camera while developing models tailored to specific visual contexts. It introduces a methodology for collecting local images, annotating them with a large general-purpose model, and then deploying these models. The methodology is validated using a multi-camera setup. We demonstrate that selecting frames with the highest confidence scores from local models yields the most accurate DNN, addressing RQ1. Additionally, this chapter analyzes the potential biases introduced by model-based pseudo-annotations compared to human annotations, specifically in the context of *Students*' accuracy, addressing RQ2.

- **Chapter 5** extends the methodology from Chapter 4 by introducing a camera clustering approach to reduce the number of models that need to be trained and monitored in multi-camera systems. To identify similar cameras, it introduces a novel distance metric based on cross-performance of models, followed by applying Hierarchical Clustering based on this criterion. Results show that lightweight models can achieve optimal accuracy when trained on camera clusters that balance similarity within the camera domain and data diversity. This chapter addresses research questions RQ3 and RQ4.
- **Chapter 6** approaches scalability within a system of systems, addressing RQ5. This chapter defines the abstractions, commitments, and strategies necessary to build holonic processing and learning. It then presents two applications of the holonic architecture: first, demonstrating the architecture's capability to manage multi-method, multi-camera processing; and second, showcasing its ability to handle the integration and removal of sensors in a learning system.
- The final **Chapter 7** offers a comprehensive overview of the scalability challenges tackled and the solutions proposed throughout the thesis.

A recommended flow of reading is provided in Fig. 1.6.

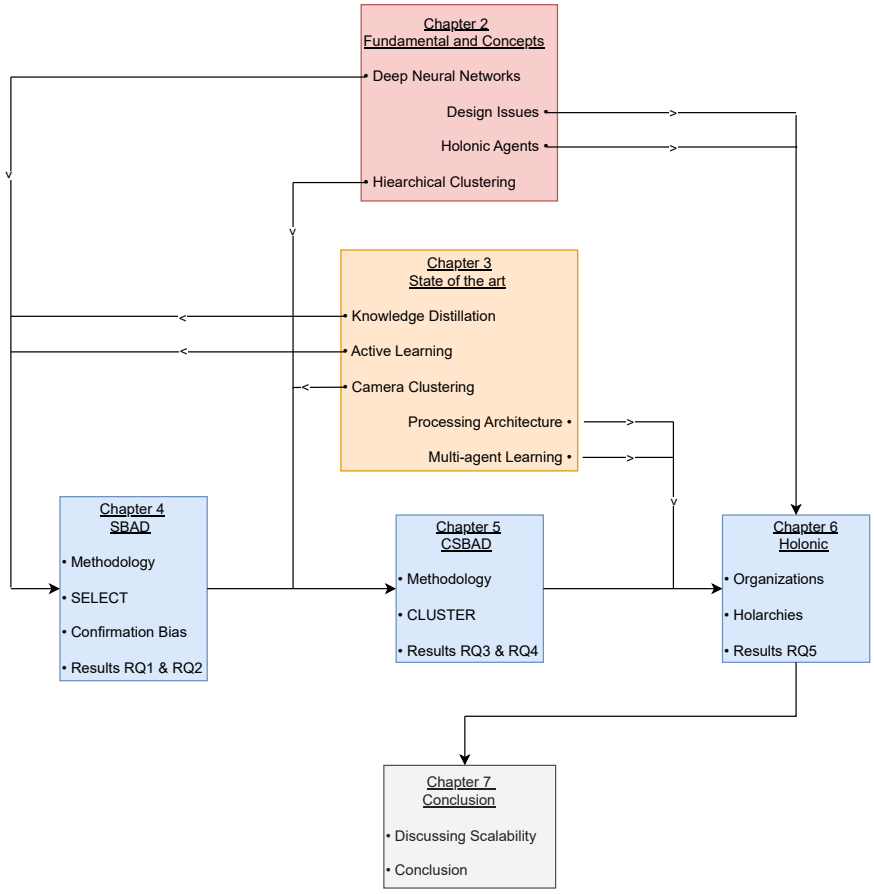



**Fig. 1.6** Recommended flow of reading.

# 2

# Fundamentals and Concepts

This chapter provides a tailored introduction to the key concepts and methods utilized in this thesis.

Section 2.1 introduces essential concepts in image processing with machine learning, underpinning the discussions in Chapters 4 and 5. To introduce Chapter 6, Section 2.2 lists issues in systems design, while Section 2.3 includes an introduction to agent theory and holonic agents. Finally, Section 2.4 offers a pedagogical example of hierarchical clustering, facilitating comprehension of the methodology discussed in Chapter 5.

*Remark* 2.1. For a quick reference, key concepts are defined at the end of each section and listed in the glossary. Each section begins with the necessary foundational concepts to ensure a clear understanding of the content.

## 2.1 Deep Neural Networks for Image Processing

This section provides a customized introduction[1] to DNNs for object detection in images. It begins with an overview of image processing in Section 2.1.1, followed by a discussion on the fundamentals of DNNs in Section 2.1.2, and their training in Section 2.1.3. Next, Section 2.1.4 explores

[1] For a deeper understanding of deep learning, readers may refer to the book *Deep Learning* by Ian Goodfellow, Yoshua Bengio, and Aaron Courville [19].

the You Only Look Once (YOLO) framework, which is widely used for object detection due to its effectiveness in real-time processing scenarios. Section 2.1.5 introduces the evaluation metric for object detection. Finally, Sections 2.1.6 and 2.1.7 address key challenges in the automatic pseudo-labeling and continuous learning of DNNs, specifically *confirmation bias* and *catastrophic forgetting*.

### 2.1.1 Image Processing

Digital image processing is fundamental to various professional activities, such as industrial automation, smart mobility, and medical diagnostics. The primary goal of image processing is to automatically extract meaningful features from visual data, supporting decision-making processes. These tasks may include detecting objects (*e.g.*, vehicles, unattended items), segmenting regions of interest (*e.g.*, tumors in medical images), or classifying image content (*e.g.*, assessing damage in disaster areas). Image processing approaches are typically divided into two main categories: **signal-based** and **machine learning-based** approaches.

**Signal-Based** methods use predefined visual features (such as intensity or texture) and apply thresholding techniques to map these features to specific objects or regions. These methods are generally more interpretable and simpler to implement, making them suitable for well-defined, controlled environments. However, their applicability is often limited to specific scenarios.

**Machine Learning-Based** methods learn directly from large datasets, generalizing patterns and distributions of events within the data. When presented with a new instance, these models estimate the likelihood of various outcomes based on learned patterns.

*Remark* 2.2. In Section 6.3, we combine signal-based and machine learning to create a multi-method application. Specifically, we employed background subtraction, which begins with a background model and calculates the difference between consecutive frames to detect motion, determining the presence of an object if the change exceeds a certain threshold. The machine learning model is YOLO.

### 2.1.2 Deep Neural Networks

DNNs are a class of machine learning models that have demonstrated remarkable capabilities. DNNs consist of multiple layers of interconnected neurons that collaboratively transform raw input data, such as images, into increasingly abstract and informative representations. This ability to automatically learn and extract features makes DNNs particularly powerful for tasks like image classification, object detection, and pattern recognition.

A typical DNN comprises the following layers:

- **Input Layer:** Receives the raw input data.
- **Hidden Layers:** A series of layers where neurons apply non-linear transformations, enabling the network to learn complex patterns.
- **Output Layer:** Produces the final output, such as class labels in classification tasks.

The depth of a DNN, determined by the number of hidden layers, allows the modeling of intricate, non-linear relationships within the data. While deeper networks can learn more complex patterns [19], they also increase computational demands.

### 2.1.3 ABC of DNN Training

The learning process is driven by backpropagation, iteratively adjusting the network's weights to minimize the loss (*i.e.*, prediction errors) on the training set.

For object detection, the loss function typically combines two components:

- **Classification Loss**: Measures the accuracy of the predicted classes for each object.
- **Localization Loss**: Measures how well the predicted bounding boxes match the ground truth boxes.

Once the loss is defined, the optimization algorithm gradient descent updates the model's parameters to minimize it. Given the large size of training datasets, data is partitioned into batches. Each iteration involves: 1) processing a batch of data, 2) calculating the loss, and 3) updating the model's weights based on the gradient of the loss.

After all batches are processed, one epoch is completed. Training DNNs requires several epochs—multiple full passes through the entire training dataset.

Instead of training from scratch, fine-tuning offers a viable option to reduce computational effort. Fine-tuning involves adjusting a model trained on a large dataset, such as COCO [27], to perform well on a smaller, task-specific dataset. This approach is particularly valuable in scenarios with limited labeled data, as it leverages the general features learned during pre-training while adapting to the specific requirements of the new task. It is a cornerstone of the approach discussed in Chapter 4.

### 2.1.4 You Only Look Once (YOLO) Object Detection Framework

YOLO frames object detection as a single regression problem, simultaneously predicting bounding boxes and class probabilities, often using regression loss functions such as smooth L1 or IoU (Intersection over Union). This approach makes YOLO exceptionally fast and suitable for real-time applications. Its architecture is scalable, offering various model sizes to accommodate different computational resources and accuracy requirements.

This is practical in our *Teacher-Student* framework, where we typically use the large YOLOv8**x6** as the *Teacher* and the smaller YOLOv8**n** as the *Student*. YOLO is also well-suited for fine-tuning, with numerous studies demonstrating its effectiveness [28, 29, 30].

### 2.1.5 Evaluating Model Performance in Object Detection

Mean Average Precision (mAP), especially mAP50-95, provides a comprehensive assessment by averaging precision scores across various Intersection over Union (IoU) thresholds. **mAP50-95** represents the average of mAP scores computed at IoU thresholds typically ranging from 0.50 to 0.95 in increments of 0.05. This metric is crucial as it assesses not only the model's ability to detect objects but also its precision in localizing them.

### 2.1.6 Confirmation Bias in Self-Supervised Learning

Confirmation bias in semi-supervised or unsupervised learning can lead to significant errors by reinforcing incorrect predictions, thereby misleading

learning [31]. This can be caused by a difficult image or hallucination, as illustrated in Fig. 2.1 and Fig. 2.2.

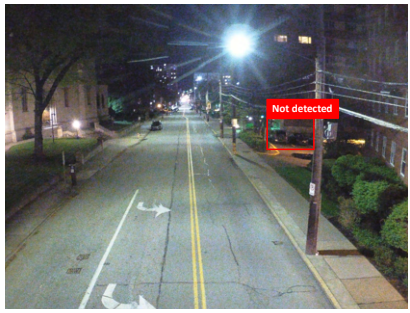



**Fig. 2.1** Hard-to-label example for the *Teacher* (YOLOv8x6).

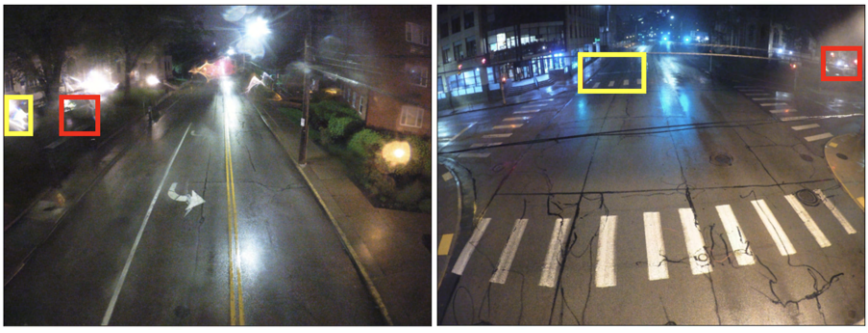


**Fig. 2.2** Hallucination of DNNs. In yellow, the prediction of the *Student* (YOLOv8n) and in red the pseudo-labels of the *Teacher* (YOLOv8x6). Both models predicted vehicles where there were none.

**Definition 2.3** (confirmation bias)**.** It refers to noise accumulation in learning when the model is trained using incorrect predictions over and over.

### 2.1.7 Catastrophic Forgetting

When artificial networks are continuously trained, we could observe a decline in performance on previously encountered data, especially if the new data involves new or varied data classes [32]. The causes behind catastrophic forgetting include **network drift** and **inter-class confusion**. In the

first case, the neural network drifts from the feature space learned from the old class training data to that of the new class data. In the second case, the boundaries between new and old classes are not well defined because they have never been trained together [32].

**Definition 2.4** (catastrophic forgetting)**.** It refers to the tendency of an artificial neural network to abruptly and drastically forget previously learned information upon learning new information.

## 2.2 System Design Concepts

Let us take the example of a video surveillance company that secured a new contract to solve problem $A$. The engineering team, under pressure to deliver a high-performing solution at the lowest cost, quickly develops a specific module, $m_A$, and tests it on a subset of $N$ cameras of type $X$ in the customer's environment.

The remainder of this section starts from this context and considers several scenarios. Section 2.2.1 considers the expansion of the system, reaching a new scale. Section 2.2.2 approaches the issues of entangled modules. Section 2.2.3 discusses how modern systems are growing towards more integrated and complex systems. Section 2.2.4 considers the complications in controlling units joining and leaving the network. Finally, Section 2.2.5 discusses the evolving nature of deployment contexts and their impact.

### 2.2.1 Expansion

The customer decides to expand the system, increasing the initial scale to the power of two; the system size is now $N^2$. To accommodate this growth, the team plans to increase the processing power of module $m_A$. However, given that current hardware has limited resources (such as memory and speed), the team opts for distribution and parallelization.

System designers often rely on the assumption that "what worked in the past will continue to work in the future" when planning expansions. However, **expansion can introduce unexpected stresses on the system.**

A practical example can be found in the training of large models. Due to the computational demands, practitioners need to distribute the training among several servers. As the models become larger, not only does

the number of training servers increase, but so does the communication overhead among them. Consequently, the training is slowed down and further scaling could be compromised [33]. In general, **each scale brings new challenges.**

### 2.2.2 Technical Debt of Single Purpose Design

The new sensors, $X'$, differ slightly from the originals, $X$. The customer now requires a solution for a modified scenario, $A'$. A common strategy is to develop a new module, $m'_A$, which uses $m_A$ as an input and applies a minor correction.

This approach is often favored under deadline pressure because it allows for quick adjustments without a complete system redesign. However, this can result in what is known as a correction cascade, where each subsequent scenario, like $A''$, leads to further iterations, resulting in $m''_A$ that builds upon the previous modules.

This cascading approach can create an improvement deadlock, establishing a new system dependency on $m_A$, which makes future improvements significantly more costly to analyze. As the system grows more complex, improving the accuracy of any individual component, such as $m_A$, $m'_A$, or $m''_A$, can inadvertently degrade overall performance. This occurs because each model in the cascade is tightly coupled with the others, making the system fragile and resistant to change. This approach introduces significant technical debt, rendering the system increasingly difficult to maintain, debug, and extend [14].

### 2.2.3 Towards Complex and Integrated Systems

Cameras are not only spatially distributed but can observe the same scene and compensate for a malfunctioning camera (due to occlusion or fault) by reconstructing missed detections [34]. Similarly, we could develop a portfolio of modules, $\mathbf{m}_A := \{m_{1A}, m_{2A}, \cdots\}$ from different paradigms but all solving Problem $A$ to improve the solution.

*Remark* 2.5. This has been demonstrated in Section 6.3.

Another consideration is the possibility of undeclared future clients. In other words, the system might become part of a larger ecosystem [16, 35].

In short, **multi-camera, multi-method systems** are **inherently complex systems**. According to Herbert Simon in 1976 [36], complexity involves

systems characterized by a high number of components with strong interdependence among parts and with rich and diverse information. Thus, they are harder to analyze and predict their behaviors.

For further reference to what complexity means, we provide Definition 2.6.

**Definition 2.6** (Complex System)**.** A complex system is defined as a system with the following attributes:

- **Number of Components**: A system with many components is often more complex than one with fewer components.
- **Interdependence**: The greater the interdependence among the components, the more complex the system.
- **Information Content**: The complexity of a system can also be measured by the amount of information it contains, especially if it involves diverse, non-repeating components.
- **Decidability and Computational Complexity**: Systems that are undecidable or require significant computational resources to understand or navigate are considered complex.

### 2.2.4 Towards Open Systems

The massive *Internet of Things (IoT)* enables cities to become urban sensing platforms [9][2]. Practitioners are resorting to large numbers of inexpensive miniaturized devices (Jetson Nano, Raspberry Pi, etc.) to reduce expansion costs. This increases the likelihood of devices or methods joining, leaving, or failing during operation.

Consequently, the systems are more open, facing non-linear processes. Correction mechanisms designed for minor sensor changes, or worse, ignoring the open nature, may result in failure or become counterproductive [37, 38].

**Definition 2.7** (open system)**.** A system subject to a dynamic composition with continuous addition or removal of units.

[2]For example, a Belgian initiative, Telraam, proposes that citizens set up an edge camera and a Raspberry Pi on their window to help monitor traffic (see `https://telraam.net/`).

### 2.2.5 Stochastic and Evolving Context

In 2012, Place Debrouckère in Brussels was primarily used by cars. Therefore, a surveillance system designed at that time would have been optimized for traffic monitoring, focusing on vehicle detection and flow analysis. By 2021, however, the area had been transformed into a mostly pedestrian zone, drastically altering the operational environment for the surveillance system, as illustrated in Fig. 2.3. This shift in distribution can cause a model deployed in 2012 to experience a performance drop by 2021 [15].

The design of machine learning systems should hence take into account that their behavior evolves with environmental data and user interactions [14]. Enforcing strict abstraction boundaries by prescribing specific intended behavior is often ineffective [14].


**Fig. 2.3** Place Debrouckère, Brussels, Belgium in 2012 and 2021. This figure depicts ever-evolving cities. The deployment of machine learning models should take into consideration continuous evolution.

### 2.2.6 Conclusion

Modern systems will face growing engineering challenges, exacerbated by: (1) the increasing diversity of subsystems, (2) the unpredictable nature of third-party subsystems joining and leaving open environments, and (3) continuous (self-)integration. While single-purpose design expedites a solution, increasing profit and customer satisfaction, it leads to significant technical debt. This can be costly to address during system refactoring and impede expansion opportunities.

## 2.3 Introduction to Holonic Agents

Before introducing the holon in Section 2.3.3, we begin with the definition of an intelligent agent in Section 2.3.1. Section 2.3.2 then discusses organizations in Multi-Agent System (MAS).

### 2.3.1 What is an Agent?

The term “agent" has diverse interpretations across fields such as biology, computer vision, and software engineering. In artificial intelligence, Wooldridge’s 1995 work [39] offers a broad definition of intelligent agents, describing them as hardware or software-based systems with the following properties:

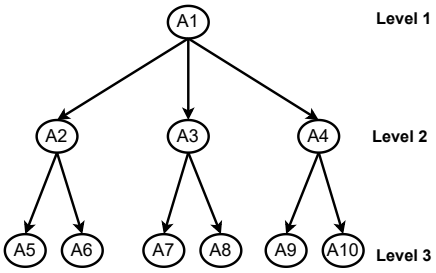
1. **Autonomy:** Agents operate without direct intervention from humans or others and have control over their actions and internal state.
2. **Social ability:** Agents interact with other agents (and possibly humans) via some form of communication language.
3. **Reactivity:** Agents perceive their environment and respond to any change in a timely fashion.
4. **Pro-activeness:** Agents do not simply act in response to their environment; they can exhibit goal-directed behavior by taking the initiative.

*Remark* 2.8. Although this definition has been further extended in recent literature, it is considered satisfactory for this thesis.

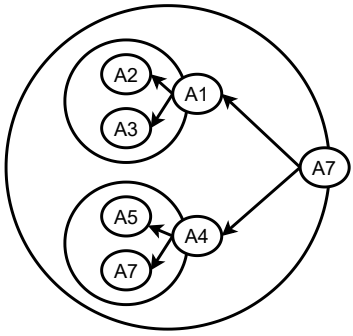
When multiple agents interact in a shared environment, they form a MAS, a system capable of solving complex, distributed problems that exceed the abilities of a single agent. Within a MAS, agents may cooperate or compete, depending on the system’s design and their individual goals. The MAS framework offers other advantages when the amount of data cannot be transferred on the same computer [40], or when potentially private information should not be shared [41].

### 2.3.2 Organizational MAS

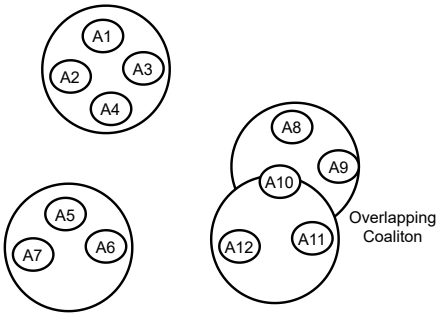
As the fundamental concepts of MASs **are more intentional and social than implementation-oriented**, MASs are best viewed as organizational



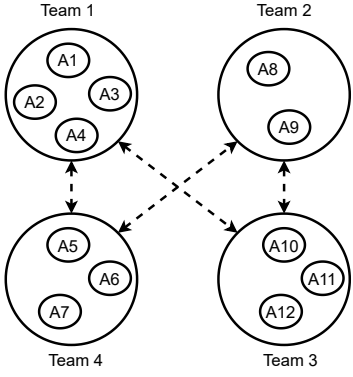

**(a) Hierarchical.** Agents are arranged in a hierarchy, with control flowing from higher to lower levels.




**(b) Holonic.** A holon is a composite agent formed by a group of subagents.




**(c) Coalition.** Agents form temporary coalitions to achieve specific objectives, dissolving once the goal is met.




**(d) Team-Based.** Long-term collaboration among agents within a team.

**Fig. 2.4** Some organizational structures in MAS. Each structure can then be derived to better achieve MAS' goals. For example, [2] introduces a "Vertical Specialization" model, where a hierarchical task delegation strategy assigns simpler tasks to less capable agents, escalating more complex tasks to more capable agents to balance cost efficiency with operational effectiveness. Similarly, [3, 4] propose a "Specialized Intelligent Communities" model that adopts a team-based structure, leveraging labor division and specialization to achieve complex goals efficiently.

structures of autonomous, proactive agents interacting to achieve shared or individual goals [42]. Similar to large, decentralized human organizations, MASs require strategies to achieve long-term objectives [43]. Both must adapt to changing environments, leverage diverse strengths, and mitigate weaknesses. Both are composed of individual “agents" that provide services, collaborate, compete, and operate in dynamic environments with agent turnover.

Organizational theory from social science inspired software designers who developed methodologies for the development of MAS to break down design complexity via 1) the use of metaphors that are more accessible to software engineers and 2) a focus on high levels of abstraction to enable the integration of new agents, even when they differ significantly from existing ones [42, 43, 44].

No single organizational structure fits all scenarios. The choice depends on factors like structuring criteria (*e.g.*, function, division, territory, product/service, client, process), types of nodes and edges (*e.g.*, power, communication, collaboration), dynamics (*e.g.*, static or dynamic), environment (*e.g.*, stable, unstable, complex), type and source of change (*e.g.*, structure, culture, process, strategy), and the number of hierarchies [43].

The possibilities are numerous, and compounded organizational structures further expand these options. For instance, an organization can be both hierarchical and team-based, though this complexity may come at the cost of a more intricate design.

Many organizational agent-oriented methodologies have been developed such as ADELFE [22, 45], ASPECS [6], Gaia [46], INGENIAS [47], SODA [48] or Tropos [49]. Each has its own focus: ADELFE on adaptive systems and cooperative agents, ASPECS on holonic multi-agent systems, Gaia on static organization and roles, and SODA on the environment.

Given our choice of the holonic paradigm, we will adhere to the definitions of ***Organization*** and ***Role*** provided by ASPECS [6]:

**Definition 2.9** (organization)**.** An ***Organization*** is a subsystem where components play roles and interact together to achieve a shared goal in the context of this organization.

**Definition 2.10** (role)**.** A ***Role*** is both an expected behavior to fulfill (part of) a requirement and a status to the role’s agent in the organization.

### 2.3.3 Holonic Agent

Arthur Koestler introduced the term “holon" in 1967, a philosophical concept describing entities that are both self-contained and parts of larger systems [24]. For instance, a cell exists independently but also forms part of a human body, just as a human is independent but can be part of a work team.

The power of the holon lies in its dual nature, similar to the two-faced Janus: one face represents the self and looks inward, while the other represents the whole and looks outward. A holon is simultaneously independent with its own characteristics and functions, yet it is also a component of larger systems.

A holon constitutes a way to gather local and global view [24, 50]. This duality enables the holonic agent to exhibit individual behaviors as regular atomic agents or participate in emergent collective behaviors built up from their interaction. For example, in [51], one agent was programmed to flee upon detecting another agent, while the other was programmed to approach and join any agent it encountered. The **interplay of** these simple, **individual behaviors led to** the spontaneous **emergence of a complex** hunting dynamic within the system.

Applying such a whole-part relationship to different entities of a system creates a multi-level interconnected structure of holons, where the behavioral variations of **each holon at any level are characterized by operational variation of its subordinate holons in a lower level**. Such structure is called a “holarchy" illustrated in Fig 2.4b. This makes them intrinsically recursive and capable of naturally describing complex systems of hierarchical nature [20], with various granularity levels [35, 52].

In the holonic terminology, holonification is the process of grouping agents into a holarchy, resembling complex clustering based on criteria like capabilities and resource access. A well-planned multi-agent holonic system design can significantly improve efficiency and manage complexity [53, 54]. For concise definitions of these terms, see Definitions 2.11, 2.12 and 2.13.

**Definition 2.11** (holon)**.** A holonic agent is a self-nested structure that can be seen, depending on the level of observation, either as an autonomous whole entity or as an organization of other holons.

**Definition 2.12** (holarchy)**.** A hierarchy of self-regulating holons.

**Definition 2.13** (holonification)**.** The process of grouping a scattered agent population into a holarchy.

## 2.4 Hierarchical Clustering

Hierarchical clustering is a method used to group similar objects into clusters. One popular approach is agglomerative clustering, which starts by treating each object as its own cluster and then successively merges the closest clusters until all objects are in a single cluster. The process relies on a distance measure and a linkage criterion to determine how distances between clusters are computed after each merge.

### Step-by-Step Example with Letters

Consider the letters A, C, D, L, and O. We define the distance between any two letters as the number of letters separating them plus one. Here is our initial distance matrix:

| | $A$ | $C$ | $D$ | $L$ | $O$ |
|---|---|---|---|---|---|
| $A$ | 0 | 2 | 3 | 11 | 14 |
| $C$ | 2 | 0 | 1 | 9 | 12 |
| $D$ | 3 | 1 | 0 | 8 | 11 |
| $L$ | 11 | 9 | 8 | 0 | 3 |
| $O$ | 14 | 12 | 11 | 3 | 0 |

### 2.4.1 Initial Clustering and Distance Calculation

Initially, each letter is its own cluster:

- Clusters: {A}, {C}, {D}, {L}, {O}

The first step is to merge the two clusters with the smallest distance. Here, C and D have the smallest distance of 1. We merge them into a new cluster {C, D}. Now, we need to update the distance matrix using a linkage criterion to determine the distance between the new cluster {C, D} and the other clusters.

### 2.4.2 Linkage Criteria

Let's compute the distances using different linkage criteria:

1. **Min Linkage:**

$$\text{Distance to A: } \min(\text{dist}(A,C), \text{dist}(A,D)) = \min(2,3) = 2$$
$$\text{Distance to L: } \min(\text{dist}(L,C), \text{dist}(L,D)) = \min(9,8) = 8$$
$$\text{Distance to O: } \min(\text{dist}(O,C), \text{dist}(O,D)) = \min(12,11) = 11$$

2. **Max Linkage:**

$$\text{Distance to A: } \max(\text{dist}(A,C), \text{dist}(A,D)) = \max(2,3) = 3$$
$$\text{Distance to L: } \max(\text{dist}(L,C), \text{dist}(L,D)) = \max(9,8) = 9$$
$$\text{Distance to O: } \max(\text{dist}(O,C), \text{dist}(O,D)) = \max(12,11) = 12$$

The updated distance matrices are:
**Min Linkage:**

| | $A$ | $L$ | $O$ | $\{C,D\}$ |
|---|---|---|---|---|
| $A$ | 0 | 11 | 14 | 2 |
| $L$ | 11 | 0 | 3 | 8 |
| $O$ | 14 | 3 | 0 | 11 |
| $\{C,D\}$ | 2 | 8 | 11 | 0 |

**Max Linkage:**

| | $A$ | $L$ | $O$ | $\{C,D\}$ |
|---|---|---|---|---|
| $A$ | 0 | 11 | 14 | 3 |
| $L$ | 11 | 0 | 3 | 9 |
| $O$ | 14 | 3 | 0 | 12 |
| $\{C,D\}$ | 3 | 9 | 12 | 0 |

### 2.4.3 Further Merging

Next, we find the smallest distance in the updated matrix. For Min Linkage, the smallest distance is 2 (between A and {C, D}). We merge them into {A, C, D}.

Continuing this process, we eventually reach the final clusters. Here are the steps for Min Linkage:

1. Merge {A} and {C, D} to form {A, C, D}.
2. Next, {L} and {O} merge because their distance is 3.

The final clusters are {A, C, D} and {L, O}. If we merge these clusters, we form the final cluster {A, C, D, L, O}.

The final distance matrix (if continued to complete linkage) would look like:

| | $L$ | $O$ | $\{A, C, D\}$ |
|---|---|---|---|
| $L$ | 0 | 3 | 8 |
| $O$ | 3 | 0 | 11 |
| $\{A, C, D\}$ | 8 | 11 | 0 |

# 3

# State of the art

This section provides the foundational research framework and related work for this thesis. First, Section 3.1 discusses Knowledge Distillation (KD), a research area focusing on how to train compact models from large models. Next, Section 3.2 covers Active Learning (AL), a framework for selecting the most valuable data to label for training a model. To diminish the number of models to be trained, we propose to cluster cameras, and Section 3.3 reviews related work on multi-camera aggregation. Section 3.4 explores the architectures used for sensor processing. Finally, Section 3.5 discusses the landscape of learning in a Multi-Agent System (MAS).

## 3.1 Knowledge Distillation

KD is a process where a compact *Student* model learns from a more complex *Teacher* model [55], illustrated in Fig. 3.1. It is used to create compact models that are suited to limited storage and computational capacities [56], as encountered on edge devices [57, 58]. Through regular updates, KD ensures that the *Student* model continues to perform effectively even when operating on changing data distributions [59]. In that sense, KD can be framed as a **source-free** Test-Time Adaptation (TTA) [60]. In comparison with source-based TTA such as [61], we do not use the model nor the data

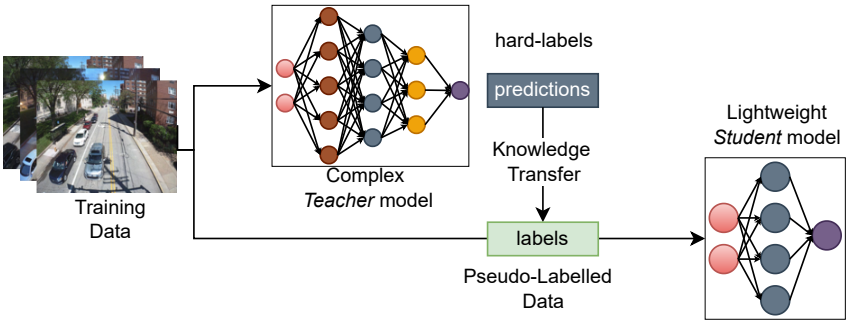



**Fig. 3.1** The essence of KD lies in transferring the knowledge of a large, well-trained model to a smaller, more efficient one. This transfer makes it feasible to deploy complex models in resource-constrained environments. In this thesis, we use "data distillation" proposed by [5], which consists of the *Teacher* generating (pseudo-)labels instead of manual labels for the *Student*'s training set.

of a *Student* to train a new one. Instead, we employ a large *Teacher* model to generate pseudo-labels directly in the **target environment**[1].

Recently, KD has supported innovative applications in video analytics, particularly through *Online Distillation* [63, 64], where a compact model's weights are updated in real time to mimic the output of a larger, pre-existing model. Other notable applications include *Context- and Group-Aware Distillation* [65, 66, 67], which accounts for context and/or sub-population shifts when delivering *Student* models.

The recourse to another model for labeling could lead to noise accumulation where the *Student* is trained using incorrect predictions. To tackle this, *Active Distillation* [68] combines AL with KD to train *Students* with a reduced number of appropriately selected samples to be robust to a *Teacher*'s inaccuracy, while *Robust Distillation* [69] explores the transfer of adversarial robustness from *Teacher* to *Student*.

With automatic labeling, it is tempting to train a *Student* for each device, multiplying the number of models to be trained. This "rebound effect" could lead to significant training and inference costs as we scale. To mitigate the associated costs (*e.g.*, hourly usage fees of a training cloud, data

[1]TTA is a powerful paradigm designed to adapt pre-trained models to distribution shifts between source domain (*i.e.*, model training data) and target domain (*i.e.*, the deployment domain), just before making predictions [60]. This approach is particularly relevant in real-world scenarios where machine learning systems are deployed in the wild, such as when images are captured by different cameras [62].

transfer), we could apply AL, which aims to identify the most informative examples for labeling. This is discussed in the next section.

## 3.2 Active Learning

AL aims to select and label the most informative data points to reduce the number of samples required to train a model [70]. AL can be implemented offline (Pool-Based AL) or on-the-fly (Stream-Based AL), from continuous streams (illustrated in Fig. 3.2).

Stream-Based Active Learning (SBAL) is the most relevant for camera-based systems, where vast amounts of unlabeled images arrive continuously, **making it impractical to maintain all of the data in a candidate pool**[2]. However, most of the AL methods developed for DNNs have been focused on the pool-based scenario [71, 72, 73, 74, 75, 76, 77, 78]. SBAL has mostly been studied in conjunction with classification or regression models [79, 80, 81], with traditional statistical models [82, 83]. Those methods do not generalize to DNNs. In contrast, our work considers AL to train a *Student* with images labeled by a *Teacher* model (and not a human).

By leveraging KD, our method diverges from traditional AL approaches due to the presence of inaccurate labels. The importance of developing sampling methods that are query-efficient and robust to labeling inaccuracies due to *Teacher* imperfection was first introduced in Active Distillation in [68]. However, it was designed for a pool-based setting, meaning that we need to store all images before selection. We introduce in this thesis Stream-Based Active Distillation (SBAD), a stream-based version of Active Distillation, which requires sampling methods to select images on the fly, thus avoiding their storage.

[2]Video streaming yields vast quantities of data. For instance, a $2000 \times 1500$ resolution camera, capturing at 10 frames per second, results in around 900 MB of data per minute, or 9,072 TB weekly.

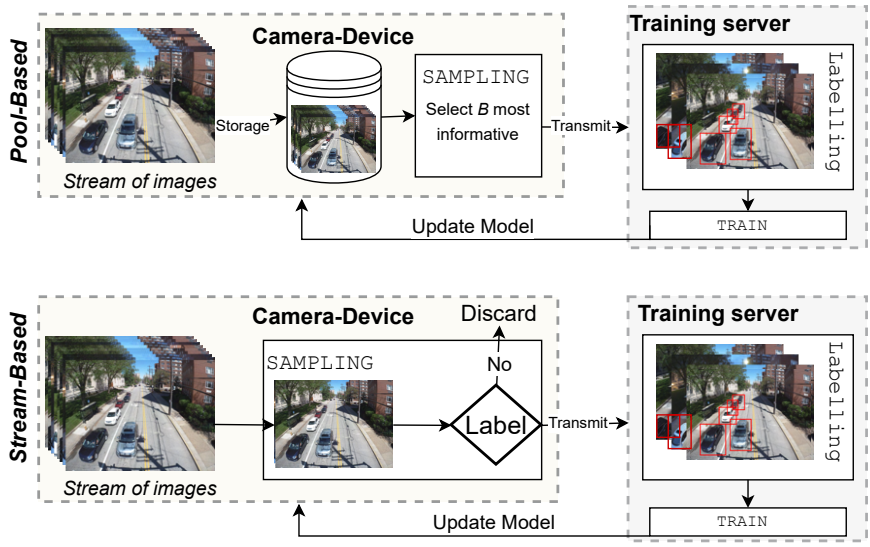



**Fig. 3.2** Active-Learning Categories. In a Pool-Based approach, all data is collected before sampling. In contrast, a Stream-Based setting processes each image on-the-fly, deciding whether to keep or discard it.

### 3.2.1 Key Frame Extraction in Video Analysis

Key frame extraction aims to select a subset of the most discriminative frames from a video [84]. This process is crucial for applications such as action recognition [85] and video summarization [86], enhancing video browsing, retrieval, and overall user experience. Typical challenges include identifying important frames or shots, detecting significant events, and maintaining overall coherence between selected frames [84].

A comparison with this literature is not straightforward, as these methods do not aim to select the most informative images for training *Students*. However, since they handle videos, their techniques can inform the design of SBAL strategies for cameras. Specifically, key frame extraction focuses on selecting representative frames from videos with diverse visual information, accounting for variations introduced by different camera viewpoints, lighting conditions, and motion. Furthermore, key frame extraction methods strive to reduce inter-frame redundancies [87], aligning with the need for diverse samples in DNNs to enhance learning of the camera distribution.

## 3.3 Camera Clustering

In multi-camera systems, the technique of scene clustering plays a crucial role in aggregating and synthesizing information from diverse sources effectively. This technique supports video summarization [88], anomaly detection [89], and content recommendation [90], with hierarchical methods addressing redundancy in tracking and overlapping scenes [91, 92, 93, 94, 95, 96]. Despite advances in techniques such as multiview clustering [97, 98], a significant challenge persists in **selecting representative frames for each video**, and, to the best of our knowledge, no previous work has studied the aggregation of views to reduce the number and improve the accuracy of *Student* models serving a multi-camera system.

## 3.4 Multi-Sensor Processing Architectures

Surveillance systems are commonly designed as centralized systems where sensors connect to a central node. However, this creates a bottleneck at the central node, potentially leading to system collapse as the system expands and demand increases. To enhance scalability, researchers have explored multi-layer sensor management [99]. In this hierarchical approach, the central node decomposes the processing loads into smaller assignments and reduces communication costs since only results are transmitted to upper layers.

Conventional software engineering techniques (*e.g.*, object-oriented) do not have the apparatus to handle this rising complexity as they have been inspired by programming concepts (*e.g.*, procedural scripting) [23]. A solution from research in software engineering is to be inspired by living systems and their organization [23, 100]. In this approach, software components are endowed with autonomy and interaction skills, referred to as “agents".

Holonic Multi-Agent System (HMAS) derives from MASs by using holons to represent whole-part relationships, allowing for hierarchical, modular and detailed system modeling through the aggregation of agent achievements. This approach has been successfully developed for independent multi-sensor platforms [35]. Further, in [101], the author proposed a holonic federated sensor management framework for pervasive surveillance

systems and provided intelligent agents (autonomous, social, proactive, and reactive) using the Belief-Desire-Intention framework [39].

Despite its promising potential, HMAS remains scarcely discussed and applied in sensor management. Most existing research focuses on multi-sensor management, treating signal processing as a single, static method. However, factors such as camera types and inter-camera coordination directly influence how processing models should function. By **not considering multi-method approaches in multi-sensor systems**, we **limit the flexibility** of these systems.

## 3.5 Multi-Agent Learning

Adaptive MAS networks leverage online learning strategies to dynamically respond to environmental changes, highlighting the importance of distributed and collaborative learning [102].

[103] introduce a system utilizing reinforcement learning to align agent actions with collective goals, minimizing human oversight. Agents are organized into "Subworlds" for focused collaboration, yet application of reinforcement learning in complex scenarios with varied sensors and methods encounters obstacles like unclear rewards and limited exploration hindering the required diversity of learning [103, 104, 105].

Organizational learning considers agents evolving through both personal and collective learning efforts, enhancing agents' abilities in MAS through management-mediated interactions and task alignment to boost system efficiency [106, 107]. This model emphasizes the role of knowledge sharing in improving workflows and establishing structural knowledge, crucial for system resilience. Social science [108] research reflects on the applicability of this framework to understand the impact of staff turnover on management, analogous to agent dynamics in open MAS.

Hierarchical learning [109] uses hierarchical MAS to streamline Machine Learning (ML) training in various geographical locations. By modeling challenges as a hypergraph, the system organizes agents, each with unique skills and knowledge, into a structured, multi-tiered network. This design not only facilitates the decentralized handling of ML algorithms and data but also significantly enhances the efficiency and scalability of processing large, distributed datasets.

In the context of distributed ML, [110] pioneered Federated Learning

(FL) to train neural networks across distributed datasets, prioritizing data privacy and computational efficiency [110]. However, FL faces hurdles in communication and training reliability. Hierarchical FL addresses these by grouping users to improve FL security and efficiency through group-specific updates [111]. Personalized FL [112] methods aim to produce personalized models for different users or clusters of users [113] to keep track of their individualized requirements. Hierarchy has also been instrumental in Fog Learning. Unlike FL, which is based on a star topology of device-server interactions, Fog Learning explicitly considers the network and topology structures among the devices and enables intelligent device collaborations through data and parameter offloading [114].

[26] abstract FL with Holonic Learning (HoL), applying holonic principles to a collaborative learning framework. In that sense, FL can be seen as a first-order HoL. HoL enhances model cooperation with specific strategies for aggregation, communication, and commitment within holons, facilitating complex yet intuitive collaboration of nodes in comparison with Fog Learning. In this balance between local autonomy and coordinated decision making, holonic systems are better equipped to tackle challenges such as adaptability, fairness, and scalability [26].

In conclusion, recent advances have highlighted the efficacy of HoL in improving system adaptability and scalability, particularly in applications that require autoscaling and autotuning [26, 115]. However, the field must advance in providing reliable design, tuning, and debugging to ensure control in uncertain environments and mitigate costs associated with failure [115]. This emphasizes the importance of further engineering and research towards *reliable open systems*.

## 3.6 Conclusion

This thesis proposes SBAD, a novel framework that harnesses KD and SBAL for sampling unlabeled streaming data, correcting labeling inaccuracies, and constructing the smallest possible training set

We extend this approach with Clustered Stream-Based Active Distillation (CSBAD), which clusters camera data to balance training complexity and enhance model accuracy.

To design such a system, we consider a multi-sensor and multi-method approach that encompasses sensor heterogeneity, environmental factors,

the dynamics of open systems, and the mutual influence between cameras and processing behaviors. Building upon existing research in holonic multi-sensor architectures and holonic multi-agent learning, our design considers sensors and methods jointly as integrated learning and processing units.

# 4

# Stream-Based Active Distillation

*Remark* 4.1. This chapter is a reformulation of the author's works [30] and [116].

*Remark* 4.2. Before proceeding, readers are encouraged to familiarize themselves with machine learning concepts such as *confirmation bias* (see Section 2.1.6) and *catastrophic forgetting* (see Section 2.1.7).

STREAM-BASED active distillation (SBAD) is a technique designed to craft a lightweight DNN *Student* model tailored specifically for a camera's unique environment. SBAD operates by gathering local samples directly from the camera, which are then pseudo-labeled by a large, general-purpose model referred to as the *Teacher*. This approach significantly reduces labeling costs compared to using human annotators.

A core component of SBAD is its sampling strategy, which is focused on selecting the smallest and most informative set of local samples. This reduces both the training costs for the *Student* and the number of queries made to the *Teacher*. To further optimize efficiency, the decision to label an image is made in real-time, avoiding the need to store or transmit the continuous stream of images. Fig. 4.1 illustrates the proposed sampling process.

The chapter is structured as follows. Section 4.1 introduces the general SBAD framework. Section 4.2 details the sampling strategy, which

selects samples where the *Student* model exhibits the highest confidence. Section 4.3 elaborates on the datasets, models, sampling methods baseline and evaluation metrics employed in our study. Section 4.4 evaluates the effectiveness of various SELECT sampling strategies and explores how the number of images per stream impacts the performance of individual *Student* models, addressing RQ1. Finally, Section 4.5 analyzes the correlation between the complexity of *Teacher* models and their propensity to induce *confirmation bias*, answering to RQ2.

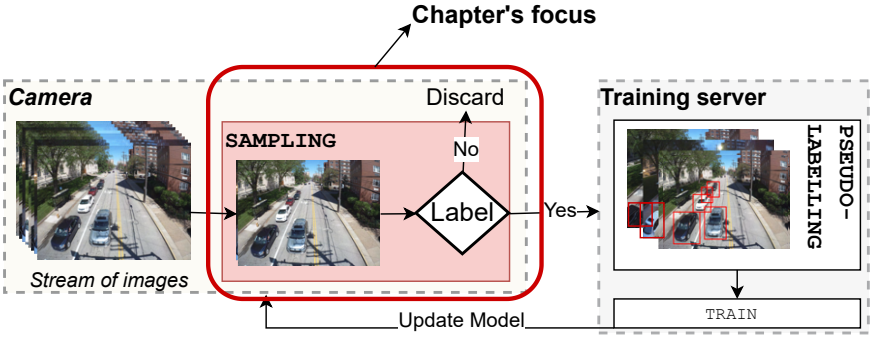



**Fig. 4.1** This chapter examines the real-time selection of images for training a lightweight model, using a larger model as the label provider. The sampling process aims to minimize both training and query costs while ensuring robustness against labeling inaccuracies.

## 4.1 SBAD Framework

Consider a video stream X , linked to **a lightweight and pre-trained** *Student* model, $\theta$ for object detection. To maintain the *Student* up-to-date, models are fine-tuned on a central server using selected samples from the videos. The two-level system, encompassing *Student* and a *Teaching Server*, is detailed below.

### 4.1.1 Student-level

*Student* operates on cost-effective hardware and uses a SELECT $(I_t)$ function to decide whether an image $I_t$ , at a timestamp $t$, is informative to train the

model. SELECT functions must be *efficient*. Efficiency is crucial; if SELECT takes longer than the frame interval to make decisions, a buffer must be used to store incoming images. This increases the demand for data storage, thus conflicting with scalability goals. Additionally, to reduce training cost, the set of selected frames and their associated pseudo-labels, which constitute the training set $\mathcal{T}$, must not exceed a frame budget $B$ per *Student*, hence, $|\mathcal{T}| \leq B$.

### 4.1.2 Teaching Server

To moderate annotation costs, the images are pseudo-labeled by a universal but imperfect model $\Theta$ referred to as *Teacher*. Upon receiving an image $I$ from a stream X, $\Theta$ pseudo-labels it by generating a set of bounding boxes $\tilde{P}_t$ and then adds the pair $(I_t, \tilde{P}_t)$ to the image stream's database $\mathcal{T}$.

**Algorithm 1** SBAD Framework

**Require:** pre-trained *Student model* $\theta$, a general purpose *Teacher* model $\Theta$, a training frame budget $B$, a SELECT strategy, video stream X.
**Ensure:** $B \geq 1$.

▷ INITIALIZATION
$\mathcal{T} \leftarrow \emptyset$
$t \leftarrow 0$ ▷ Timestamp
▷ SAMPLING
**while** $|\mathcal{T}| \leq B$ **do**
  Observe current frame $I_t$
  **if** SELECT$(I_t)$ = TRUE **then**
    $\tilde{P}_t \leftarrow \Theta(I_t)$ ▷ Infer pseudo-labels
    $\mathcal{T} \leftarrow \mathcal{T} \cup (I_t, \tilde{P}_t)$
  **end if**
  $t \leftarrow t + 1$
**end while**
▷ FINE-TUNING AND UPDATE
$\theta \leftarrow train(\theta, \mathcal{T})$

## 4.2 Top-Confidence Strategy

Our investigation into video sampling strategies underscored the superiority of the TOP-CONFIDENCE approach. In this method, the local DNN *Student* $\theta$ begins with a warm-up phase by inferring on $w$ frames, without selecting any for training. For each frame $I_t$, where $t \in \{1, \cdots, w\}$, it produces a set of $L$ bounding boxes $\tilde{P}_t = \{\tilde{p}_{1t}, \cdots, \tilde{p}_{Lt}\}$ along with their associated confidence score $\tilde{C}_t = \{c_{1t}, \cdots, c_{Lt}\}$, computed as proposed in [117]. The highest confidence score from each frame, $\max_l c_{lt}$, where $l \in \{1, ..., L\}$, is collected during this phase for threshold determination. More precisely, this process establishes a threshold $\Delta$ as the $(1-\alpha)$-upper percentile of maximum scores, aiming for an $\alpha$ selection rate, meaning that $\mathbb{P}(\max(\tilde{C}_t) \geq \Delta) = \alpha$. Subsequently, for $t > w$, a frame $I_t$ is forwarded to the *Teacher* if $\max(\tilde{C}_t) \geq \Delta$. The pseudo-code is presented in Algorithm 2.

**Algorithm 2** TOP-CONFIDENCE Thresholding for Object Detection

**Require:** model $\theta$, budget $B$, warm-up period $w$, selection rate $\alpha$.

Initialize list **MaxScores** to empty
**for** $t = 1$ to $w$ **do**
    $\tilde{P}_t, \tilde{C}_t \leftarrow \theta(I_t)$ ▷ Infer boxes and scores
    Append $\max(\tilde{C}_t)$ to **MaxScores** ▷ Collect max score for $I_t$
**end for**
▷ Compute selection threshold $\Delta$
$\Delta \leftarrow \text{percentile}(\textbf{MaxScores}, 1-\alpha)$
$i \leftarrow 0$ ▷ Selected frames counter
**while** $i < B$ and more frames available **do**
    $\tilde{P}_t, \tilde{C}_t \leftarrow \theta(I_t)$ ▷ Infer boxes and scores
    **if** $\max(\tilde{C}_t) \geq \Delta$ **then** ▷ If meets $\Delta$, forward
        Forward $I_t$ to *Teacher*
        $i \leftarrow i + 1$
    **end if**
    $t \leftarrow t + 1$ ▷ Next frame
**end while**

Note that $\alpha$ can be carefully increased to allow more permissive selection, satisfying the image budget. This was applied in our experiments (see Section 4.3.2).

## 4.3 Materials and Methods

We present the dataset, the evaluation protocol, models and training approach and other sampling strategies that were evaluated and compared to our proposed method.

### 4.3.1 Watch and Learn Time-lapse Dataset

The Watch and Learn Time-lapse (WALT) dataset [118] features footage from nine cameras over several weeks, offering a variety of viewpoints, lighting and weather conditions, including snow and rain (samples are shown in Fig. 4.3). Data collection spans periods ranging from one to four weeks per camera, resulting in a weekly range of 5,000 to 40,000 samples due to asynchronous recordings and differing sampling rates. Table 4.3 includes weekly image counts and sizes for each camera. Notwithstanding, some instances of burst phenomena are observed, presenting cases of strong temporal redundancy.

*Test sets*

We had nine test sets, two from the original paper and seven that we annotated by selecting a week of footage from each camera, which we excluded from training. The 1850 images were meticulously labeled to identify "vehicle" instances, resulting in 12,577 annotated instances. These sets, now publicly available, facilitate further research. The details and links can be found in Table 4.4 in annex Section 4.9.

*Human Annotation For Sampled Sets*

In our experiment, in Section 4.5.1, "*Impact of Teacher Size*", we manually annotated 96 images sampled by a YOLOv8n$^{COCO}$ *Student* utilizing the `Least-Confidence` or `Top-Confidence` approach. Subsequently, we compared the performance of these *Students* against when they are trained using the *Teacher*'s pseudo-annotations or a human. In Table 4.5 in annex Section 4.9, we present the images sampled by each strategy, which we annotated, thereby creating the equivalent human annotations.

### 4.3.2 Other SELECT and Methods Parameters

Our investigation evaluates alongside `Top-Confidence` three other `SELECT` strategies:

- `Uniform-Random`: a frame is chosen if $s \sim U(0,1) \geq 1 - \alpha$. To ensure statistical reliability, we conducted six iterations with six different seeds. The variability in performance is quantified by the Margin of Error (MoE), calculated as $\text{MoE} = z \times \frac{\sigma}{\sqrt{n}}$, where $z$ represents the 1.96 z-score correlating with a 95% confidence level, $\sigma$ is the standard deviation of the scores, and $n$ is the effective sample size.

- `Least-Confidence`: targets frames with minimal prediction confidence to present the model with challenging instances. This method mirrors the `Top-Confidence` algorithm (Algorithm 2), albeit with a reversed selection criterion, shifting from $\max(\tilde{C}_t) \geq \Delta$ to $\min(\tilde{C}_t) \leq \Delta$.

- `N-First`: opts for the initial $B$ frames from the stream, establishing an unbiased baseline.

Standard parameters are set to $\alpha = 0.1$ and $w = 720$. Relaxation was done in high-budget scenarios for cameras 3 and 9 due to data constraints, increasing $\alpha$ to 0.55 for camera 9 and 0.25 for camera 3, thus facilitating expedited frame selection to fulfill our image budget.

### 4.3.3 Models, Training and Distillation Implementation

We utilize models pretrained on the COCO dataset [27] to ensure a broad foundational understanding of the object detection task. We employ as the *Teacher*, YOLOv8x6$^{\text{COCO}}$ and YOLOv8n$^{\text{COCO}}$ as the *Student*. The architectures have 68.2 and 3.2 million parameters, respectively. Both models serve as the evaluation benchmark. No preprocessing was done, but for post-processing, similar COCO object classes (bikes, cars, motorcycles, buses, and trucks) were grouped into a "vehicle" category. The *Student* is then fine-tuned by default on 100 epochs, which can vary according to the experiment. The batch size remains constant at 16. All other training parameters are defaults as configured in [10].

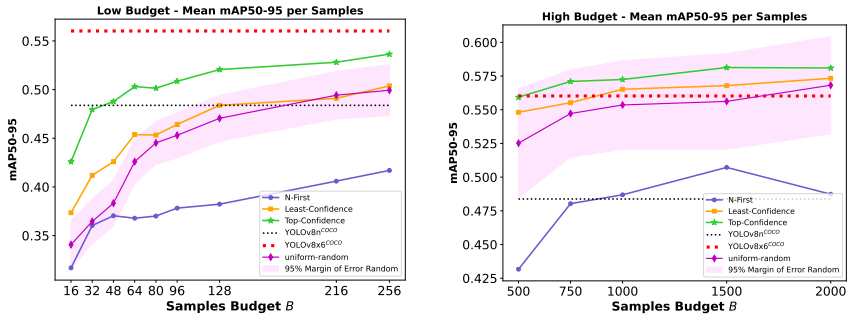
### 4.3.4 Evaluation

System performance was evaluated by the accuracy of *Student* models on their respective camera.

*Detection accuracy*

In object detection model evaluation, the mAP50-95 metric signifies the mean Average Precision (*mAP*) across various thresholds of intersection over union, spanning from 0.50 to 0.95 in increments of 0.05.

## 4.4 Impact of SELECT and Sample Budget




**(a)** *Low* Budget **(b)** *High* Budget

**Fig. 4.2** Average mAP50-95 scores for different training sample sizes, using four SELECT models. The number of epochs is 100. Key observations are: 1) `Top-Confidence` is the best sampling strategy and 2) fine-tuned compact models can outperform a *Teacher* YOLOv8x6$^{COCO}$.

A *Student* is fine-tuned using $B$ samples from a camera, selected on-the-fly according to the SELECT strategies (detailed in Section 4.3.2). The resulting model is then evaluated on the camera's test set. The low budget $B \leq 256$ experiments were conducted across all cameras for each week, resulting in eighteen pairs per experiment. For the high budget $B > 500$ scenario, we combined weekly data from each camera to reach the budget count, forming nine pairs per experiment.

Figure 4.2 presents the average mAP50-95 scores as a function of $B$ across different SELECT strategies, maintaining a consistent 100 epochs for

all tests. Key observations include the following:

1. `Top-Confidence` is the best sampling strategy. Conversely, `N-First` was catastrophic and performed worse than the *Student* when a small $B$ was considered.
2. Fine-tuned *Students* with `Top-Confidence` outperform unfine-tuned models YOLOv8n$^{COCO}$ and YOLOv8x6$^{COCO}$, with sample budgets exceeding 48 and 750, respectively.
3. As $B$ increases, the model quality improves but at a slower rate, suggesting the achievement of peak accuracy.

The findings emphasize the need for careful design of `SELECT`, particularly when the budget is limited. We observe that, when $B$ is small, a bad selection can severely disrupt learning. Methods such as `N-First` and `Random` are **ill-suited for the redundant nature of video streams and tend to select frames with little or no relevant activity, penalizing the learning process**. Conversely, the `Top-Confidence` approach **tends** to select frames containing objects. Furthermore, **opting for** higher-confidence frames **can enhance** the accuracy of pseudo-labels in a distillation scheme, while **selecting the** least confident frames **tends to amplify** the *Teacher*'s inaccuracies. Section 4.5 delves into this phenomenon, known as confirmation bias.

## 4.5 Confirmation Bias Analysis

To identify strategies mitigating *confirmation bias*, we conducted two experiments exploring the interplay of `Least/Top-Confidence` with varying i) *Teacher* and ii) *Student* sizes. Model size (Params) is quantified by the number of parameters in millions. In both experiments, models were trained for 100 epochs.

### 4.5.1 Impact of *Teacher* Size

In this experiment, **we manually annotated** 96 images sampled by a *Student* utilizing the `Least/Top-Confidence` approach and subsequently compared the performance of these *Students* against scenarios in which a model *Teacher* pseudo-annotated the same set of images. This experiment was

conducted across three camera streams[1], employing various model sizes for the *Teachers*.

**Table 4.1** mAP50-95 values, and percentage increase per annotator for `Least/Top-Confidence`. The *Student* is YOLOv8n$^{COCO}$ with $B = 96$ samples and 100 epochs.

| Annotator | Params (M) | Least | Top | % Increase |
|---|---|---|---|---|
| YOLOv8n | 3.2 | 0.28 | 0.45 | +56% |
| YOLOv8s | 11.2 | 0.35 | 0.49 | +37% |
| YOLOv8m | 25.9 | 0.37 | 0.52 | +40% |
| YOLOv8l | 43.7 | 0.40 | 0.51 | +30% |
| YOLOv8x6 | 68.2 | 0.40 | 0.52 | +31% |
| human | N/A | 0.40 | 0.52 | +27% |

The results, detailed in Table 4.1, reveal that less complex *Teachers* tend to yield inferior *Students*, a situation that the `Least-Confidence` sampling strategy exacerbates. Intriguingly, human annotations did not offer improvements. This suggests the `Least-Confidence`'s tendency to choose ambiguous or challenging samples, thereby reinforcing missed detections of the *Teacher*, as illustrated in Fig. 2.1 or incorrect detection as shown in Fig. 2.2.

Conversely, `Top-Confidence` significantly improves pseudo-label quality and, consequently, model performance, highlighting the critical need for selecting high-quality, clear images for training. Moreover, this finding reiterates the effectiveness of pseudo-labeling, indicating that it nearly matches the performance enhancements provided by human-generated labels.

### 4.5.2 Impact of *Student* Size

We explored the influence of *Student* model size on performance, utilizing $B = 256$ images from nine cameras, pseudo-labeled by YOLOv8x6$^{COCO}$.

Our observations, detailed in Table 4.2, yield two key insights, namely, 1) a general enhancement in performance with the increase in *Student* size, and 2) a growing disparity in the effectiveness of `Least-Confidence` and `Top-Confidence` strategies as the model size expands. These findings indicate that larger models are better equipped to regularize and generalize, a

[1]Specifically, Camera 1 from Week 1, Camera 2 from Week 1, and Camera 3 from Week 5. Links to the manual annotations can be found in the GitHub repository.

crucial aspect in correcting the inaccuracies inherent in pseudo-labels generated through the `Least-Confidence` strategy. Furthermore, the strategy `Top-Confidence` appears to leverage the advanced capabilities of bigger models more efficiently.

**Table 4.2** Comparison of mAP50-95 values and percentage increase per *Student* for Different Strategies. The *Teacher* is YOLOv8x6$^{COCO}$ and $B = 256$ samples and 100 epochs were used.

| *Student* | Params (M) | Least | Top | % Increase |
|---|---|---|---|---|
| YOLOv8n | 3.2 | 0.52 | 0.53 | +2.8% |
| YOLOv8s | 11.2 | 0.56 | 0.58 | +2.4% |
| YOLOv8m | 25.9 | 0.56 | 0.59 | +4.6% |
| YOLOv8l | 43.7 | 0.57 | 0.60 | +4.3% |
| YOLOv8x | 68.2 | 0.57 | 0.60 | +5.0% |

## 4.6 Discussion

Our analysis confirms the expected significance of $B$ in line with the data-hungry nature of DNNs. The epoch count, while relevant, proved to be of secondary importance (see Fig. 5.4). Increasing $B$ means more images are used for training, potentially boosting performance but requiring more resources. Naturally, `Top-Confidence`'s success highlights the importance of selecting clear, well-lit images with identifiable instances for training in order to minimize the budget.

Finally, the superior performance of compact fine-tuned *Students* over a large general-purpose *Teacher* demonstrates distillation's effectiveness, simultaneously reducing complexity and boosting domain-specific performance without the need for a human annotator. Regarding model size, we observe that larger *Teacher* or *Student* models generally yield better predictive accuracy.

## 4.7 Limitations and Perspectives

### 4.7.1 Continuous Deployment

Sustaining a budget $b$, where $b \leq B$, for each $\mathcal{T}$ through successive deployment cycles (*i.e.*, integration of a new device or regular updates), can have advantages. First, this strategy saves resources, as only $B - b$ new images and annotations are required in the next deployment cycle. Second, this approach could mitigate *catastrophic forgetting* (see Def. 2.4). As this requires further assumptions on the frequency of fine-tuning and the use case, we considered $b = 0$ in our experiments. The extension can be done by changing Line 1 in Algorithm 1 to $\mathcal{T} \leftarrow \texttt{RECYCLE}(\mathcal{T})$, where `RECYCLE` manages the storage of $\{\mathcal{T}\}$ for subsequent deployment.

## 4.8 Conclusion

We presented SBAD to bridge the gap between large-scale and affordable DNNs while adapting to changing environments. The framework evaluates the informativeness of each frame, accounting for *Teacher* imperfections in a KD scheme. Experiments demonstrate that traditional AL strategies may not be optimal for KD.

Future research could explore alternative sampling strategies and distillation mechanisms to improve performance.

## 4.9 Appendix of Chapter 4

### 4.9.1 Training Set Details

**Table 4.3** Detailed weekly image counts per camera. Missing values indicate weeks with no data collection.

| Cam | Week1 | Week2 | Week3 | Week4 | Week5 | Total |
|---|---|---|---|---|---|---|
| **1** | 5711 | 5640 | 13159 | 17641 | 4919 | **44070** |
| **2** | 5827 | 5748 | 14192 | 17865 | 525 | **45157** |
| **3** | - | - | 8209 | - | - | **8209** |
| **4** | - | 35192 | 30154 | - | - | **65346** |
| **5** | 4707 | 5019 | 42555 | 46799 | 5368 | **101448** |
| **6** | 16064 | 21711 | 24765 | 27184 | - | **89724** |
| **7** | - | - | 680 | 28914 | - | **29594** |
| **8** | - | - | 30427 | 29455 | - | **59882** |
| **9** | 6857 | 6055 | - | - | - | **12912** |

### 4.9.2 Test Set Details

**Table 4.4** Test set details summarizing the image count (Samp.), number of instances (Inst.), average annotation time (Time), and Roboflow Links for each camera. URLs are relative to the preamble: `https://universe.roboflow.com/sbad-dvvax/sbad_`.

| Cam | Samp. | Inst. | Time (s) | Link<br>*Append preamble* |
|---|---|---|---|---|
| **1** | 300 | 2133 | 7.33 | cam1_test_set/dataset/3 |
| **2** | 300 | 1252 | 4.2 | cam2_test_set/dataset/1 |
| **3** | 300 | 1475 | 4.9 | cam3_test_set/dataset/1 |
| **4** | 50 | 267 | 5.3 | cam4_test_set/dataset/1 |
| **5** | 300 | 210 | 0.7 | cam5_test_set/dataset/1 |
| **6** | 100 | 1027 | 10.3 | cam6_test_set/dataset/1 |
| **7** | 100 | 2709 | 27.1 | cam7_test_set/dataset/1 |
| **8** | 100 | 992 | 9.9 | cam8_test_set/dataset/2 |
| **9** | 300 | 2512 | 8.4 | cam9_test_set-akhti/dataset/1 |

### 4.9.3 Samples Illustration

**(a)** Camera 1 **(b)** Camera 2 **(c)** Camera 3

**(d)** Camera 4 **(e)** Camera 5 **(f)** Camera 6

**(g)** Camera 7 **(h)** Camera 8 **(i)** Camera 9

**Fig. 4.3** Images captured from the nine cameras in the WALT dataset .

### 4.9.4 Confirmation Bias Dataset

**Table 4.5** Roboflow links for the human annotation when a *Student* samples using `Top-Confidence` (Top) or `Least-Confidence` (Least) . URLs are relative to the preamble: `https://universe.roboflow.com/sbad-dvvax/`.

| **Cam** | **Week** | `SELECT` | **Link**<br>*Append to preamble* |
|---|---|---|---|
| 1 | 1 | Top | cam1_confbias_topconf96/dataset/3 |
| 1 | 1 | Least | cam1_confbias_leastconf96/dataset/3 |
| 2 | 1 | Top | cam2_confbias_topconf96/dataset/3 |
| 2 | 1 | Least | cam2_confbias_leastconf96/dataset/5 |
| 3 | 5 | Top | cam3_confbias_topconf96/dataset/2 |
| 3 | 5 | Least | cam3_confbias_leastconf96/dataset/3 |

# 5

# Camera Clustering for Scalable Model Deployment

*Remark* 5.1. This chapter is a reformulation of the author's works in [116].

With an automatic methodology such as SBAD, it is tempting to fine-tune a *Student* for each device, multiplying the number of models to be trained. To further reduce costs, we consider clustering camera nodes in a multi-camera environment to train one model per group. We set the premise that models fine-tuned on a specific camera domain are likely to transfer effectively to other domains with similar characteristics. In this work, a camera domain refers to the characteristics of the data collected by the camera, including the underlying distribution of features that might change due to different conditions (*e.g.*, locations, lighting conditions, viewpoint, camera type, etc.).

Our cluster-based training methodology (illustrated in Figure 1.2) kills two birds with one stone. First, building a dataset from multiple streams ensures a large and diverse training sample, which is essential for model training [11]. Second, restricting the clusters to streams that correspond to similar domains accounts for the limited expressivity (or representation capacity) of lightweight models [19]. In other words, our work enhances DNN accuracy by striking a balance between specificity (allocating a model for each camera) and universality (one model for all streams).

Moreover, limiting the number of clusters and thus the number of *Stu-*

*dent* models helps mitigate the increase in total training cost when the system expands. The key contributions of this chapter are the introduction of a novel camera clustering mechanism based on model cross-performance, including an in-depth analysis of how clustering impacts model accuracy. This dual contribution showcases model improvements without sacrificing accuracy and explores how various clustering strategies affect training complexity and model performance. An overview of the research question is provided in Fig. 5.1.

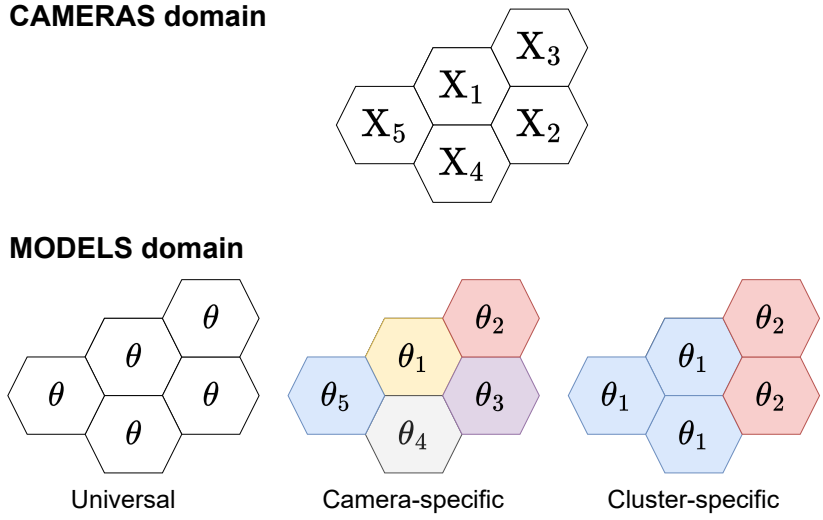



**Fig. 5.1** This chapter explores the optimal deployment strategy for camera models: whether to use a single universal model for all cameras, a specific model for each camera, or to cluster the domain into groups, each with a dedicated model.

This chapter is organized as follows. Section 5.1 extends the SBAD methodology in a multi-camera setup. In Section 5.2, we provide an automated method for clustering cameras. Section 5.4.1 considers the clustering of cameras. Section 5.3 adds some material information. Sections 5.4.2 and 5.4.3 investigate the influence of cluster count and training complexity on model performance, respectively answering RQ3 and RQ4. In Section 5.4.4, the transferability of fine-tuned models in incremental deployment scenarios is presented. Sections 5.5 and 5.6 provide discussions about this work and 5.7 concludes this chapter.

## 5.1 Clustered Stream-Based Active Distillation

Consider a set of $N$ video streams $\{X_1, \cdots, X_N\}$, with each $X_i$ linked to a lightweight *Student* model, $\theta_i$ for object detection. To maintain *Students* up-to-date, models are fine-tuned on a central server using selected samples from the videos. The two-level system encompasses *Students* and a *Teaching Server* as in SBAD in Section 4.1. Only the teaching server is augmented with a clustering methodology detailed below.

*Remark* 5.2. Note that each *Student* must comply with the `SELECT` defined in Section 4.1.1. Namely, the training set $\mathcal{T}_i$ must not exceed a frame budget $B$ per *Student*, hence $|\mathcal{T}_i| \leq B$, for all $i = 1, \cdots, N$.

### 5.1.1 Teaching Server

To moderate annotation costs, the images are pseudo-labeled by a universal but imperfect model $\Theta$ referred to as *Teacher* [5]. Upon receiving an image $I_t$ from a stream $X_i$, $\Theta$ pseudo-labels it by generating a set of bounding boxes $\tilde{P}_t$ and then adds the pair $(I_t, \tilde{P}_t)$ to the image stream's database $\mathcal{T}_i$. This process results in a collection of sets $\{\mathcal{T}_1, \cdots, \mathcal{T}_N\}$. Next, the server employs a `CLUSTER` method that partitions $\{\mathcal{T}_i\}_{i=1,\ldots,N}$ into $K \leq N$ training sets $\{\mathcal{E}_1, \ldots, \mathcal{E}_K\}$. The `CLUSTER` approach builds on the premise that models are transferable between similar streams and is detailed in Section 5.2. We also define a mapping function, $map(\cdot)$, linking *Students* to their clusters, thus enabling the appropriate updating of their weights post-training.

$K$ models are then trained on each $\mathcal{E}_k$, with $k = 1, \cdots, K$, and deployed on associated cameras. Algorithm 3 provides the pseudo-code of the methodology.

**Algorithm 3** CSBAD Framework

**Require:** *Student* model $\theta$, a general purpose *Teacher* model $\Theta$, a training frame budget $B$, a `SELECT` strategy, image streams $\{X_1, \cdots, X_N\}$, a mapping rule `CLUSTER`, K partitions.
**Ensure:** $B \geq 1, 1 \leq K \leq N$.

▷ **1. Sampling**
1: **for** each $X_i$ **do** ▷ $i = 1, \cdots, N$
2: $\mathcal{T}_i \leftarrow \emptyset$ ▷ Initialize empty training set
3: $t \leftarrow 0$ ▷ Frame number

```
    while |T_i| ≤ B do                                          ▷ Sampling
        Observe current frame I_t
        if SELECT(I_t) = TRUE then
            P̃_t ← Θ(I_t)                                        ▷ Infer pseudo-labels
            T_i ← T_i ∪ (I_t, P̃_t)                              ▷ Add to training data
        end if
        t ← t + 1                                               ▷ Move to next frame
    end while
end for
                                                                ▷ 2. Clustering
{E_1, ··· , E_K}, map() ← CLUSTER(T_1, ··· , T_N; K)
                                                                ▷ 3. Training
for k = 1, ··· , K do
    θ_k ← train(θ, E_k)                                         ▷ Students training
end for
                                                                ▷ 4. Updating streams models
for i ∈ 1, ··· , N do
    θ_i ← θ_map(i)                                              ▷ Update streams model
end for
return {θ_i}_{i=1,···,N}                                        ▷ Return updated models
```

This research aims to explore the trade-offs involved in choosing $K$. On one hand, a small $K$ implies models trained with a broad diversity and good applicability across various *Students*, potentially with improved generalization. On the other hand, a $K$ approaching $N$ may tailor models to the nuances of individual cameras.

## 5.2 Clustering

We propose a node clustering method for DNN training. Our method is based on the premise that a model fine-tuned on a particular camera domain will likely transfer effectively to other camera domains with similar features. Conversely, significant performance drops are expected across dissimilar camera domains.

CLUSTER uses a Hierarchical Clustering algorithm after the initial training of $N$ stream-specific *Student* models. This algorithm does not need a pre-specified number of clusters. Instead, the resulting dendrogram can be cut at the appropriate level to obtain an appropriate number of clusters.

Before introducing our method, we define the validation set $\mathcal{V}_i$ as the set of image and pseudo-label pairs $(I_{it}^{val}, \tilde{P}_{it})$ from a stream $X_i$. We also define the function $f(\theta, \mathcal{V})$ measuring the quality of a model $\theta$ on a set $\mathcal{V}$.

*Remark* 5.3. A reminder of how hierarchical clustering works is provided in Section 2.4.

**STEP 1 – Train $N$ Stream-specific Models:**
We fine-tune for each stream $i$ a specific model, denoted by $\theta_i$, $i = 1, \cdots, N$ [1]. They will subsequently be grouped according to their validation performance on the sets $\{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$. From an operational standpoint, the choice of $B$ and number of epochs for this step should align with your specific constraints and use case. Generally, a higher training budget $B$ allows for better capture of the stream's distribution, thus the transferability (or not) of models will be more pronounced, hence leading to more effective clustering.

**STEP 2 – Compute Cross-Domain Performance:**
We compute the cross-domain performance matrix $M \in \mathbb{R}^{N \times N}$, with each element $M_{ij}$ representing the performance metric $f\left(\theta_i; \mathcal{V}_j\right)$, which indicates how a model fine-tuned on domain $X_i$ performs on the validation set of $X_j$. The matrix is defined in Eq.5.1.

$$M := \begin{bmatrix} f\left(\theta_1; \mathcal{V}_1\right) & \cdots & f\left(\theta_1; \mathcal{V}_N\right) \\ \vdots & \ddots & \vdots \\ f\left(\theta_N; \mathcal{V}_1\right) & \cdots & f\left(\theta_N; \mathcal{V}_N\right) \end{bmatrix} \tag{5.1}$$

**STEP 3 – Compute Distance Matrix:**
To quantify the dissimilarities between the models trained on different domains, we calculate the distance matrix $D$. This matrix is computed using the pairwise Euclidean distances between the models' performance vectors, derived from performance matrix $M$ and is defined as:

$$D_{ij} = \sqrt{\sum_{k=1}^{N} (M_{ik} - M_{jk})^2} \quad \text{for } i, j = 1, \ldots, N \tag{5.2}$$

**STEP 4 – Apply Single Linkage Clustering:**
Using [119], we construct a dendrogram, *i.e.*, a tree diagram depicting how

[1] When $N$ is too large, we can randomly select a small subset of the streams to conduct this step. This bounds the number of stream-specific models to train, while providing sufficient diversity. The resulting cluster models are then deployed based on their performance on the node's validation set.

clusters merge based on minimum distance. Formally, for two clusters $A$ and $B$, the single linkage distance $L(A, B)$ is given by Eq. 5.3

$$L(A, B) = \min\{D_{ij} : i \in A, j \in B\} \tag{5.3}$$

where $D_{ij}$ is defined as in Eq. 5.2. This linkage criterion tends to merge clusters with their nearest elements.

**STEP 5 – Define Clusters:**
We define the clusters by selecting a cut-off distance $t$ on the dendrogram (see Fig. 5.2b), guided by expertise, statistical methods, or visual analysis.

## 5.3 Materials and Methods

The dataset, the evaluation protocol, models and training approach are the same as defined in Section 4.3. However, we introduce a new metric for comparison which is the *Adjusted Training Cost* described below.

### 5.3.1 Adjusted Training Cost

We define the training cost $T$ as the total number of training iterations, which equates to the number of weight updates within a model. To facilitate fair comparisons among models trained on clusters of varying sizes, we introduce an adjusted training cost $T^N$ for a specified number $N$ of streams. Considering the number of samples per stream $B$, we adjust the epoch count to ensure that the iteration count remains consistent regardless of the cluster count $K$. This calibration for models trained on any given cluster $k$, with $k \in \{1, \ldots, K\}$, is encapsulated in Eq. 5.4:

$$T_k^N = \frac{B}{\text{batch size}} \times \frac{\text{epochs}_{|K=1} \times N}{n_k} \tag{5.4}$$

Here, $n_k$ represents the stream count in cluster $k$, $B$ is the number of images collected per stream, batch size is the number of samples processed in one forward and backward pass, and $\text{epochs}_{|K=1}$ denotes the number of epochs for the universal model, *i.e.*, when $K = 1$.

## 5.4 Clustering Results

### 5.4.1 Cluster Definition

The cross-domain performance matrix $M$ in Figure 5.2a shows the mAP50-95 score of a model YOLOv8n$^{COCO}$ fine-tuned on source $cam_i$ and tested on target $cam_j$, where $i$ and $j$ are the nine cameras from the WALT dataset. The analysis utilized the first week's data from each camera, sampling $B = 256$ images using `Top-Confidence`.

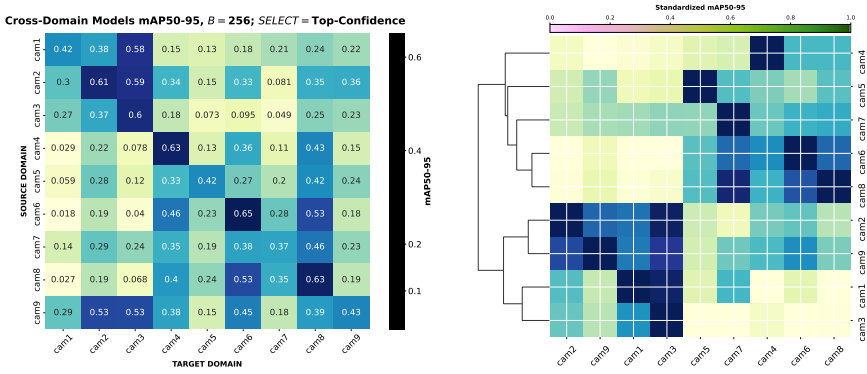
Key observations from Fig. 5.2a are:

1. Models perform best within their own domain.
2. The transfer of models results in performance degradation with variable severity.

The variance in the model's transferability across the stream can be used to group the streams. Having verified our premises, we apply Hierarchical Clustering with *minimum linkage* to detect subtle similarities. Note that in this particular scenario the application of other linkage methods such as *max, average* or *ward* leads to the same clustering. Cluster definition involved experimenting with various threshold values and observing emerging groups in the dendrogram (Fig. 5.2b). The clustering obtained for the nine WALT cameras is presented in Table 5.1.

**Table 5.1** Cameras ID clusters based on cutting the dendogram for different thresholds.

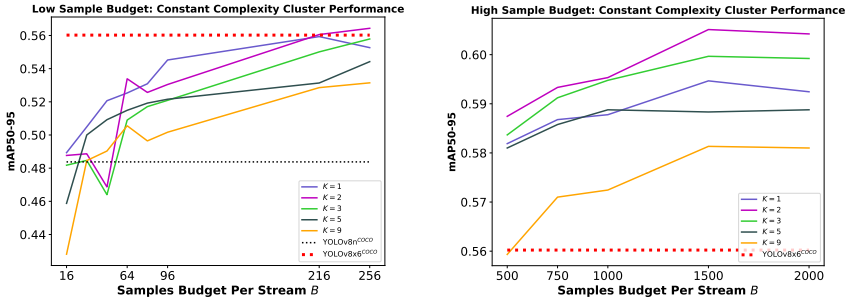
| | **Number of Clusters** $K$ | | | |
|---|---|---|---|---|
| **Cluster No.** | **2** | **3** | **4** | **5** |
| 1 | 1, 2, 3, 9 | 1, 3 | 1, 3 | 1, 3 |
| 2 | 4, 5, 6, 7, 8 | 2, 9 | 2, 9 | 2, 9 |
| 3 | - | 4, 5, 6, 7, 8 | 5, 6, 7, 8 | 6, 7, 8 |
| 4 | - | - | 4 | 5 |
| 5 | - | - | - | 4 |




**(a)** Cross-Domain Performance Matrix $M$.

**(b)** Dendrogram.

**Fig. 5.2** Clustering Definition. Fig. 5.2a depicts the cross-performance matrix $M$, where each element $M_{ij}$, $i, j = 1, \cdots, 9$, represents the mAP50-95 score of a model $\theta_i$ retrained on source domain $cam_i$ and evaluated on target domain $cam_j$. The setting is based on $B = 256$ images sampled using `Top-Confidence`. Fig 5.2b is the associated dendrogram.




**(a)** $B = \{16, 32, 48, 64, 96, 216, 256\}$

**(b)** $B = \{500, 750, 1000, 1500, 2000\}$

**Fig. 5.3** Mean mAP50-95 scores per sample budget per stream ($B$) for varying numbers of clusters were observed. Models underwent training for 100 epochs with a batch size of 16. Key findings indicate that, at a constant complexity, a universal model ($K = 1$) is preferable for lower $B$ values. However, for larger $B$ values, segmenting the system into two or three clusters yields superior outcomes.

### 5.4.2 Clustering vs Samples Budget

Figure 5.3 presents mAP50-95 scores as a function of the amount of images per stream $B$ for different numbers of clusters $K$. The clusters' definition follows Fig. 5.2b. The observations are:

1. On smaller budgets, training one model for all streams is superior. In fact, the smaller $K$ is, the better are the models.
2. On larger budgets, $K = 2$ and $K = 3$ are dominating while the scenario $K = N = 9$ is underperforming.
3. For all $K \in (1, \cdots, 9)$, the models reach their peak accuracy around $B = 1500$ samples per stream, irrespective of $K$.

The amount of images per stream $B$ dictates the best $K$. Training a single model for all streams proves most effective for a low sample budget $B$, reflecting the data-hungry nature of DNN. However, for a larger $B$, clustering increases performance. In fact, there is a a "sweet spot" between stream-specific and universal models. This suggests that clustering offers robustness by leveraging more samples and a more specific distribution facilitating the learning, particularly in less complex architectures. Yet, there's a potential downside to excessive clustering, as it might overlook the need for a model of diversity.

### 5.4.3 Clustering vs Constant Iteration per Model

Since a larger K induces more models trained with smaller training sets, we also investigated the relationship between $K$ and the number of training epochs.

Fig. 5.4 illustrates the mAP50-95 scores as a function of the number of iterations $T_1^{K=1}$ in a log-scale for the universal model. Models for clusters with $K > 1$ adjust their epoch counts, in accordance with Eq. 5.4, to maintain a consistent number of training iterations (*i.e.*, model updates) across all models, regardless of $K$. The figure uses distinct markers to represent sample sizes of $B = 16,\ 96$ and 256 per stream, and vertical lines to mark the epoch counts when $K = 1$.

The key observations are:

1. Increasing $B$ from 16 to 96, and further to 256, results in substantial improvements outpacing gains from extended epoch counts.

2. In comparison with Figure 5.3, smaller clusters ($K = 5, 9$) bridged the performance gap with larger-cluster scenarios, particularly as iteration counts increased.
3. Performance tends to plateau and even decline after reaching a base epoch (epochs$_{K=1}$) of 80 epochs, suggesting the onset of overfitting beyond a base 100 epochs.

We conclude that careful escalation of the epochs enabled smaller $K$ values to optimize model performance while minimizing the risk of overfitting.

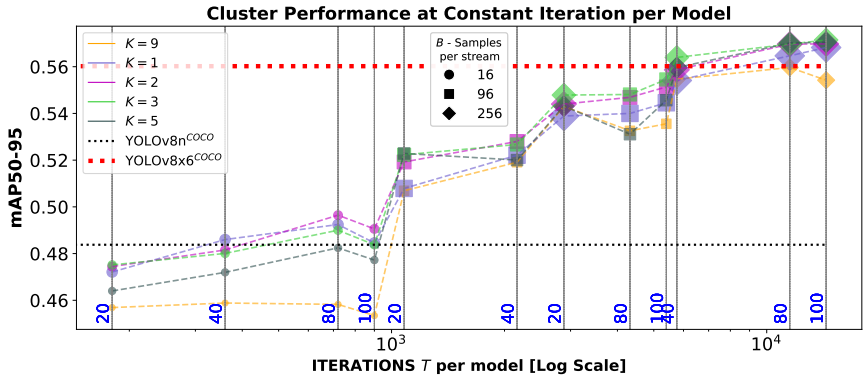



**Fig. 5.4** Mean mAP50-95 scores are presented over log-scaled iterations ($T$) for each model. Markers denote sample sizes per stream ($B = 16, 96, 256$). Vertical lines indicate the epochs for $K = 1$. For $K > 1$, the epoch count is adjusted to maintain constant iteration counts across models, following Equation 5.4. Our analysis reveals that an increase in epoch counts benefits smaller cluster configurations ($K = 3, 5, 9$), enabling them to achieve comparable or superior performance to more universal configurations ($K = 1, 2$).

### 5.4.4 Model Transfer Upon Increment

We investigated the transferability of fine-tuned models across new, albeit similar, domains. Table 5.2 details the performance outcomes for models trained on individual camera domains, an all-but-one combination of domains, and across all domains within a cluster as defined in Table 5.1. This

**Table 5.2** mAP50-95 scores for models trained under three scenarios: 1) camera-specific (SELF), 2) an all-but-one camera combination, and 3) across all cameras. Each model was trained with a sample budget per stream of $B = 256$ and for 100 epochs.

| **Cluster** | **cam$_1$** | **cam$_2$** | **cam$_3$** | **cam$_9$** |
|---|---|---|---|---|
| SELF | 0.47 | 0.65 | 0.64 | **0.48** |
| {cam$_1$, cam$_2$, cam$_3$} | **0.50** | **0.66** | **0.67** | 0.44 |
| {cam$_1$, cam$_3$, cam$_9$} | 0.49 | 0.57 | 0.66 | 0.48 |
| {cam$_2$, cam$_3$, cam$_9$} | 0.38 | 0.65 | 0.64 | 0.47 |
| {cam$_1$, cam$_2$, cam$_3$, cam$_9$} | 0.47 | 0.65 | 0.66 | 0.47 |
| YOLOv8n$^{COCO}$ | 0.39 | 0.53 | 0.54 | 0.40 |

analysis indicates that models transferred from an all-but-one domain scenario to a new, similar camera domain perform less well than a general-purpose model. For instance, a model trained on domains $cam_2$, $cam_3$, and $cam_9$ underperformed on domain $cam_1$ relative to YOLOv8n$^{COCO}$.

Consequently, **extreme caution is recommended when transferring fine-tuned models to new devices**.

## 5.5 Insights

The CSBAD framework performance is determined by the number of images per stream ($B$), active learning strategies (`SELECT`), number of clusters ($K$), number of epochs, and dimensions of the *Students* along with the complexity of the *Teacher*.

As for the optimal number of clusters ($K$), there is no one-size-fits-all answer. The choice depends on the system size ($N$), available hardware, and budget $B$ constraints. A smaller $B$ or $N$ requires a smaller $K$ to create larger clusters, providing sufficient samples and training iterations for model training. Conversely, a larger $N$, $B$, or number of iterations per model $T$ shifts the constraint to hardware capabilities, encouraging model tailoring.

Essentially, $K$ and $B$ **act as an "adjusting variable" within the system** to achieve both a **specificity-diversity trade-off** and sufficient **training iterations per model**.

## 5.6 Limitations and Perspectives

### 5.6.1 Scalability

The applicability of current $K$ results is limited to comparable scales, and determining system behaviors for higher order systems (*e.g.*, $N > 100$) requires experimentation at much higher scales.

### 5.6.2 Non-optimal Resource Management

Our approach necessitates initiating model training from a general purpose model with each iteration of the framework, leading to suboptimal resource utilization. This process neglects the potential advantages of leveraging previously trained models. For instance, in the `CLUSTER` approach, although $N$ stream-specific models are initially developed, they are not utilized in subsequent training for cluster-specific models when $K < N$, resulting in their underutilization.

Future research should investigate **incremental deployment**, with the effective utilization of pretrained models, potentially through methods like weight aggregation [120] and fine-tuning for fewer epochs, to boost learning efficiency and reduce training costs.

### 5.6.3 Compartmentalized Clustering

Our experimental protocol mandated one model per cluster, with each stream assigned exclusively to one cluster. This approach aimed to discern a trade-off between **system**'s **resources and performance**. However, such a compartmentalized strategy may not maximize local camera performance and overlooks potential inter-camera connections across clusters. In contrast, a stream-centric approach empowers each *Student* to select streams for grouping. As Table 5.2 in Appendix 5.4.4 illustrates, integrating data from Cameras 1, 2 and 3 is more beneficial for Camera 1's *Student* than other groupings. Yet, from a systemic perspective, pairing Cameras 1, 2, 3 and 9 yields a superior average.

Therefore, in scenarios where complexity is not a constraint, a shift towards a framework that trains one model per stream becomes viable. This approach empowers each *Student* to select from the $B \times N$ image pool independently, offering more flexibility in their training process.

## 5.7 Conclusion

The introduction of $K$, a variable representing the number of clusters, is crucial for adapting CSBAD to diverse system configurations, emphasizing the importance of balancing specificity, diversity, sample volume, and training epochs to boost the framework's flexibility. In this sense, CSBAD advocates for the clustering of camera domains to move beyond the constraints of specialized or universal networks, thereby enhancing adaptability and performance.

# 6

# Holonic Architecture

*Remark* 6.1. Before proceeding, readers are encouraged to familiarize themselves with design concepts such as "*Complex Systems*" in Section 2.2 and holonic related concepts as "*Agent*", "*Organization*", "*Role*", "*Holon*", and "*Holarchy*" which are defined in Section 2.3.

*Remark* 6.2. At this stage of the research, this chapter should be seen as a preliminary result towards a more conclusive answer to research question RQ5. Note that this chapter's results in Section 6.3 are a reformulation of the author's work [52].

MULTI-CAMERA and multi-method surveillance systems feature a large amount of interconnected components with feedback loops. This complicates system control and evaluation. The heterogeneous nature of their sensors and processing methods requires high levels of abstraction for modeling. They also undergo frequent compositional changes, such as adding or removing units, introducing significant challenges in integration, compliance and service continuity. At last, they can play a role in another bigger surveillance system.

Traditional software engineering tools like Unified Modeling Language (UML) often fall short in representing this complexity given their scriptural nature, limiting scalability and adaptability [21, 23].

This chapter's contributions are first the conception of a multi-camera, multi-method system through the holonic formalism and preliminary experiments to exploit the design. To break down the complexity, the HMAS

is augmented with architecture and organizational concepts.

Specifically:

- High-level formulation of the foundational organizations for multi-scale processing and learning in Section 6.1 and the relationships between super and sub holons instantiating organizations in Section 6.2.
- Section 6.4.1 proposes a system for DNNs modeled as a distributed, multi-tiered, and self-nested structure using holonic formalism. We provide self-organization mechanisms to create a holarchy, relying on the *specificity-diversity* trade-off not only within a group of models but also among successive levels of hierarchy where lower levels are specialist models and the top levels are more generalist models, increasing the range of possible applications.
- Toy experiments as preliminary studies encouraging holarchic development. Section 6.3 shows preliminary evidence that simple, intuitive image processing behaviors can enhance tracking system efficiency by 7% without decreasing accuracy or adding interaction overhead. Section 6.4 considers the integration/removal of sensors on a learning system. Specifically, we compare our mechanisms for integrating new sensors and deletion with a complete re-organization in terms of system accuracy.

## 6.1 Organizational Models

We establish two organizations: the ***Cyber-Physical Platform*** (CPP) organization, which elevates acquired data to a higher semantic level, and the ***Teacher-Student*** (TS) organization, which actively learns by collecting data from its deployment context and involves a higher-order entity for training supervision. Within each organization, we define ***Roles*** and interactions abstract enough to be replicated across any system layer, regardless of the unit's nature (sensor or method) or its operating mode. Section 6.1.1 presents the ***CPP*** and its roles: ***Resource Provider***, ***Observer***, and ***Sensor*** and Section 6.1.2 presents the ***TS*** organization with its roles: ***Specialized Student*** and ***Teacher***.

Note that a unit can belong to both organizations and may fulfill all roles, but at various levels of the system.

### 6.1.1 Organizational Model of the Cyber-Physical Platform (CPP)

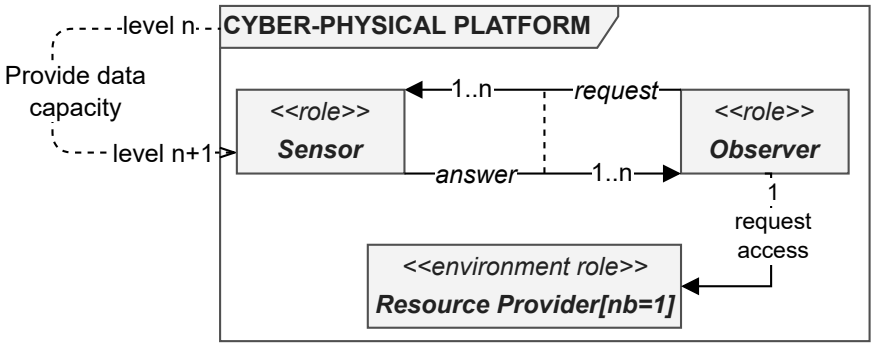



**Fig. 6.1** Organizational model of the *cyber-physical platform*, using the ASPECS notation [6]. The ***Resource Provider*** role ensures a fair distribution of the resources among all parties. The ***Observer*** role has the capacity to deliver perceptions thanks to the data acquired by the ***Sensor*** role. The acquisition of data could rely on another CPP.

***CPP*** (Fig. 6.1) is designed to respond to external requests with perceptions and to manage its finite resources to ensure fair access across multiple surveillance operations. To fulfill its purpose, we introduce the ***Resource Provider*** role, which handles authorizations of legitimate interactions and management of resources. For example, it can prevent missions that exceed available resources or violate standards, such as privacy. The ***Observer*** role has the capacity to provide perceptions after processing data given by sensors. Within the organization, multiple ***Observers*** can operate in parallel, either collaboratively or competitively. The ***Sensor***'s role is to acquire data. The acquisition of data could rely on another ***CPP***.

### 6.1.2 Organizational Model of *Teacher-Student* (TS)

Building upon the Active Distillation framework discussed in Chapter 4 and the specificity-diversity trade-off from Chapter 5, we developed an organizational model that incorporates the roles of ***Specialized Student*** and ***Teacher***, as illustrated in Fig. 6.2.

The ***Specialized Student*** role is designed to continuously collect data on sub-parts of the system's deployment environment. Under the oversight of

a higher-order ***Teacher*** entity, these *Students* learn from this data, adapting their models' weights accordingly.

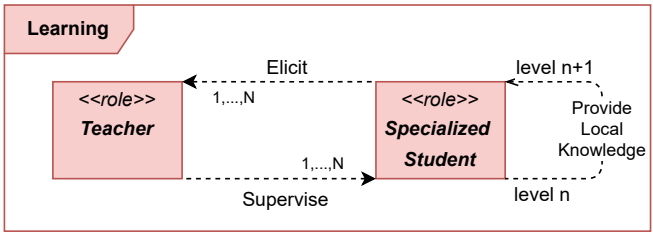



**Fig. 6.2** Organizational model of the ***Teacher-Student***, using the ASPECS notation [6]. The ***Specialized Student*** role involves a component tasked with building expertise over a delineated sub-domain in the system, *i.e.*, a regional distribution. The ***Teacher*** role supervises the learning processes of the *Students*.

## 6.2 Holarchy

This section outlines the relationships between super-holons (higher-level entities) and sub-holons (lower-level entities). The section begins by introducing new notations in Section 6.2.1. It then presents an example of a three-tiered holarchy structure in Section 6.2.2. Each level of this holarchy is a possible instance of an organization defined in Section 6.1.

### 6.2.1 Notations

A holarchy $\mathcal{H}_O^L$ includes up to $L$ vertical layers instantiating an organization $\boldsymbol{O}$, either ***CPP*** or ***TS*** in this work. A holonic unit $i$ at layer $l$, where $l$ ranges from 0 to $L$, is denoted by $\hbar_i^l$ and comprises:

- $X_i^l$: Set of operating data streams of a holon $\hbar_i^l$.
- $\mathcal{T}_i^l$: Training set of a holon $\hbar_i^l$.
- $\mathcal{V}_i^l$: Validation set of a holon $\hbar_i^l$.
- $\theta_i^l$: Processing model of a holon $\hbar_i^l$.
- $\text{SUB}_i^l$: Inner members corresponding to layer $l-1$ of a holon $\hbar_i^l$.

### 6.2.2 Multi-Scale Hierarchical Architecture

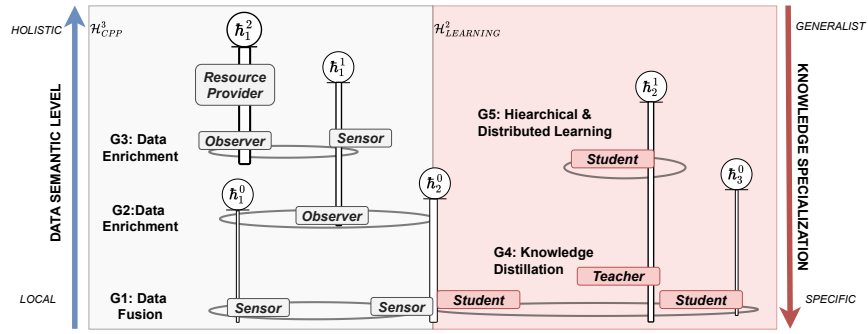



**Fig. 6.3** Holonic architecture inspired by the "cheese board" notation [6, 7]. Each level represents a different hierarchical position, defining both the semantic level of data and the degree of knowledge specialization. On the left, the $\mathcal{H}^3_{\boldsymbol{CPP}}$ instantiates the ***CPP*** organisation, and on the right, the $\mathcal{H}^2_{\boldsymbol{TS}}$ is responsible for active learning. Note that an agent can play several roles and belong to different holarchies.

The system architecture, shown in Fig. 6.3, includes two holarchies: $\mathcal{H}^3_{\boldsymbol{CPP}}$ which processes data across three levels and $\mathcal{H}^2_{\boldsymbol{TS}}$ managing knowledge across two levels.

- At Level 0, agents are the primary functional layer. They employ models designed for specific data streams. Proximity to other agents, either geographic or task-related, allows them to merge outputs and reduce errors. For example, $\hbar^0_1$ and $\hbar^0_2$ form Group G1 to fuse their outputs to feed the data request of a higher-order holon $\hbar^1_1$.

  Nevertheless, agents monitoring the same area may employ different models if their functions necessitate learning different features. Accordingly, $\hbar^0_2$ and $\hbar^0_3$, as ***Specialized Students***, form Group G4 to learn a shared model under a ***Teacher*** holon's supervision via Group G5.

- At Level 1 and 2 — Higher tiers above the agents integrate and synthesize data from specific system areas (*e.g.*, data streams sharing attributes). Holons in the ***Observer*** role, such as $\hbar^1_1$ and $\hbar^2_1$, elevate collected data to a new semantic level.

  Holon $\hbar^1_2$, a higher-order ***Specialized Student***, aggregates validation

sets from $\hbar_2^0$ and $\hbar_3^0$ (*i.e.*, $\mathcal{T}_2^1 = \mathcal{V}_2^0 \cup \mathcal{V}_3^0$), to create a broader, generalized model.

Generally, each semantic level consolidates knowledge across **broader system areas, fostering a holistic view, such as city-scale tracking.** Meanwhile, intermediate layers consisting of ***Specialized Students*** synthesize knowledge from lower tiers, to **deepen collective task understanding** and **increase models' universality**.

## 6.3 Autonomous Multi-Method for Adaptive Systems

Surveillance companies traditionally employ a universal tracker and detection model across all platforms due to their superior predictive capabilities. However, this one-size-fits-all strategy fails to account for the varying complexities across different operational environments. For example, off-peak traffic conditions with fewer vehicles might be adequately managed with simpler, less costly methods such as inter-frame differencing. In contrast, high-density traffic demands more sophisticated and expensive tracking techniques. This mono-method approach often forces a compromise between performance and cost-efficiency, whereas adopting a portfolio of diverse methods and self-organization rules could get the best of both worlds.

The upcoming experiment aims to discuss the potential of designing autonomous units that implement straightforward yet effective rules. We consider a city with various traffic densities and a system tasked with vehicle tracking. Section 6.3.1 describes a rule-based approach utilizing two distinct, autonomous detection methods—one simple and the other more complex. Section 6.3.2 presents the particular datasets and methods used in this experiment. Section 6.3.3 discusses how this approach reduces resource usage even with agent interactions, while preserving accuracy.

### 6.3.1 Toy Design of Two Detector Methods

Consider a holarchy $\mathcal{H}^3_{\boldsymbol{CPP}}$, as depicted in Fig. 6.3. We examine a client, $\hbar_1^2$, assigned to vehicle tracking. This client utilizes a tracking agent, $\hbar_1^1$, which employs two detectors with varying processing costs—inter-frame differencing and a DNN ($\hbar_1^0$ and $\hbar_2^0$, respectively). The holon detectors assess

which option is more cost-effective and performance-efficient. Initially, the less costly but less precise detector is active. Upon detecting vehicles, the system activates the more expensive, accurate detector for enhanced tracking.

An abstract behavior is provided in Algorithm 4.

**Algorithm 4** Rule-Based Interactions for an Autonomous Object Detectors

```
   Input: costX, perfX, densityInfo
   Output: Adaptive behavior based on density, cost, and performance
 1: Agent Algorithm - Method X: Behavior:
 2: if low density and costX is min then
 3:     Act as primary detector
 4: else
 5:     if average density and costX is min then
 6:         Act as activity detector
 7:         if activity detected then
 8:             Notify other method
 9:         end if
10:     else
11:         if high density and perfX is max then
12:             Act as primary detector
13:         end if
14:     end if
15: end if
```

### 6.3.2 Materials and Methods

*AI-CITY-Dataset*

The AI-City data set [1] comprises 65 videos captured by static cameras in six districts of Iowa for vehicles tracking. The seven longest videos are selected and annotated. These videos vary mainly in terms of traffic density; camera angles such as vertical and high angle; and sensor types like PTZ, Bullet, and Dome. Each video is recorded at a frame rate of 10 FPS and lasts no more than approximately five minutes.

*Higher Order Tracking Accuracy*

HOTA [121] is a community standard for evaluating the *effectiveness* of multi-object tracking (MOT). In summary, HOTA $\in$ [0,1] measures how

well the trajectories of matching detections align and averages this over all matching detections, while also penalizing detections that do not match [121].

*Processing Time*

Processing time is evaluated by the elapsed time between each processing phase. In the adaptive approach, we added the cost of agentification (*i.e.*, the cost of agent interactions).

### 6.3.3 Results

We compared the *multi-method* architecture's cost to a *mono-method* approach using 40 cameras from the AI-City Tracking Challenge dataset [1]. For detection, the mono-method approach used *YOLO v4* [122] trained on the MIO-TCD dataset [123], while the multi-method setup included *Background Subtraction* via OpenCV. Both cases employ the SORT tracker [124].

Table 6.1 displays the HOTA scores, processing times, and agentification costs, alongside their respective differences. These tests utilized 40 videos from the cameras in the AI-City Challenge [1]. Processing times are averaged from 10 simulations on an Intel(R) Core(TM) i7-8700K CPU processor running at 3.70GHz and a GPU with NVIDIA GeForce GTX 1080 Ti.

**Table 6.1** Comparison of mono-method architecture vs our proposed approach.

| | | Time (s) | | |
|---|---|---|---|---|
| **Method** | **HOTA** | **Method** | **Agent** | **Total** |
| DNN only | 0.36 | $2327 \pm 21$ | $10 \pm 1$ | 2337 |
| Multi-method | 0.36 | $2169 \pm 31$ | $12 \pm 1$ | 2182 |
| Difference | 0 | -158 | 2 | -155 |

Analysis of Table 6.1 indicates that the system maintains consistent tracking performance at scale. It also reveals a 25% increase in agent interactions; however, overall processing time was reduced by 7%.

## 6.4 Holonic Learning

Section 5.4.4 underscores that existing models cannot be transferred to a new node, even when those models are trained on similar sensors. As an implication, Chapter 5 indicates that both integrating and removing nodes require reinitialization to secure the performance. However, this leads to significant reorganization costs with sensor turnover.

To reduce frequent updates, we assess the impact of integrating $N_+$ devices on the system's models accuracy over time. In fact, while this approach avoids extensive reorganization, it risks forming increasingly random sensor groups, which may compromise the specificity-diversity trade-off as more nodes are integrated and degrade models.

Beforehand, we introduce a holonic structure in Section 6.4.1 which aim at creating a multi-level learning framework, generalizing the CSBAD presented in Chapter 5. Next, Section 6.4.2 introduces a mechanism for incorporating new nodes by coordinating with the top layers and assigning each new node to the group whose DNN model is most accurate on its data stream. Section 6.4.3 presents the materials used in this experiment. Section 6.4.5 presents the impact results through a comparison with a complete system reorganization.

In scenarios of sensor removal, the decision to retain or discard a node's data is critical. Indeed, lightweight DNNs have limited capacity, and retaining outdated data can impede the optimization of performance on remaining sensors. However, integrating old and new data could help mitigate *catastrophic forgetting* [32, 125]. Section 6.4.6 examines the impact of data retention or removal on model accuracy of both remaining and departed sensors.

### 6.4.1 Holonification

In this section, we propose a multi-tiered learning structure (see Figure 6.4), comprising a portfolio of models that range from sensor-specific to models that can be deployed to the whole network (universal) meant to handle the openness limitation of CSBAD. The holonification mechanism, based on the specificity-diversity trade-off, allows for the establishment of various granularity levels. Specifically, upper-layer models are trained on large datasets for greater universality, while lower-layer models utilize smaller

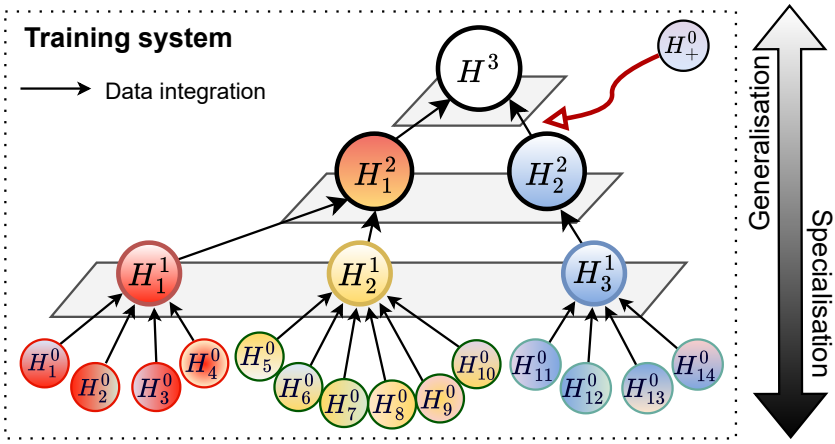



**Fig. 6.4** A three-tiered system built from the specificity-diversity trade-off. The nodes consist of (edge) DNNs trained on specific datasets. Lower nodes are typically tailored to their task (*i.e.*, analytics on a sensor) and operationally more efficient. Higher nodes, trained over vaster and more diverse data, provide generalization ability. This system can scale up or down, causing challenges in integration and adaptability.

datasets to improve specificity. It broadens the stand-point of Chapter 5, where we not only focus on operational performance but also have intermediate and global models, providing a portfolio of models for the system to choose from. For instance, a user needs a model that has a broad understanding of city-wide vehicle detection or specific models like detecting cars in parking lots.

Formally, each holon at a layer $l > 0$ is allocated a budget $B^l = B_0 \cdot 10^l$, where $B_0$ represents the number of images. This budget allows each layer's training to incorporate no more than $10^l$ datasets from preceding levels, ensuring that the dataset size for any holon $\hbar^l_i$ does not exceed $B^l$, *i.e.*, $|\mathcal{T}^l_i| \leq B^l$.

To merge holons, we mirror principles from `CLUSTER` methodology in Section 5.2, using the premise that models with similar performance have learned from comparable data, implying data sharing brings diversity of learning examples without excessive or irrelevant heterogeneity.

The remainder of this section describes the holonification process:

**STEP 1 – Cross-Performance Vector.** Assuming holons can transfer their model weights to each other within the same layer. Each holon $\hbar_i^l$ computes a performance vector $P_i$ by evaluating the effectiveness of models from other holons and itself in the same layer on its own validation data $\mathcal{V}_i^l$, according to Equation 6.1.

$$P_i^l := \left[ \; f\left(\theta_1^l, \mathcal{V}_i^l\right); \cdots ; f\left(\theta_{N^l}^l, \mathcal{V}_i^l\right) \; \right]^T \tag{6.1}$$

Where $\theta_j$ denotes the model parameters of the $j$-th holon $\hbar_j^l$, $j \in 1, \cdots, N^l$ and $f(\theta, \mathcal{V})$ the score of a model $\theta$ performance against a validation set $\mathcal{V}$.

**STEP 2 – Pair-Wise Distance Computation.** To quantify the differences between models trained in different domains, holons broadcast their cross-performance vectors $P_i^l$ defined in Equation 6.1 and compute a pairwise distance between their performance in the datasets and the performances of the other holons. Generally, for a holon $\hbar_i^l$, the distance to a holon $\hbar_j^l$ is given by Equation 6.2.

$$D_i(\hbar_j^l) = \sqrt{\sum_{k=1}^{N^l} (P_{ik}^l - P_{jk}^l)^2} \tag{6.2}$$

**STEP 3 – Agglomerative Merging using Single Linkage.** The merging of the holons is an iterative process. The set of holons $\{\hbar_1^l, \cdots, \hbar_{N^l}^l\}$ creates a higher-order holarchy $\mathcal{H}^{l+1}$ to which they belong.

At each iteration, the set of holons $\hbar^l$ transmits their smallest linkage distance. This is defined as the minimal distance between the inner members of the holons. Formally, for two holons $\hbar_A^l$ and $\hbar_B^l$, the single link distance $L(\hbar_A^l, \hbar_B^l)$ is given by Equation 6.3.

$$L(\hbar_A^l, \hbar_B^l) = \min\{D_{ij} : \hbar_i^{l-1} \in \mathsf{SUB}(\hbar_A^l),\ \hbar_j^{l-1} \in \mathsf{SUB}(\hbar_B^l)\} \tag{6.3}$$

After all linkages are evaluated, the pair with the smallest distance merges, involving a combination of their datasets. After merging, the set of holons has decremented, $\{\hbar_1^l, \cdots, \hbar_{N^l-1}^l\}$, and the linkage distances are updated for all agents.

The process ends if there remains only one holon or if the previous merge leads to a holon with a dataset size that exceeds $B^{l+1}$. In the second scenario, the process goes back to STEP 1 for the set $\{\hbar_1^{l+1}, \cdots, \hbar_{N^{l+1}}^l\}$.

**STEP 4 – Model Training.** The final steps consist in training the cluster models on the aggregated data sets.

### 6.4.2 Sensor Integration Process

A new holonic unit $\hbar_+$ joins an holarchy $\mathcal{H}^L$ of $L$ levels. Its integration starts at the highest hierarchical level, $L$, and advances downward to level 1. At each level, $\hbar_+$ is associated with the holon $\hbar_*^1$ demonstrating the highest performance on the new unit's validation set, $\mathcal{V}_+$, subject to meeting budget constraints, *i.e.*, $|\mathcal{T}_*^l \cup \mathcal{T}_+| \leq B^l$. Once integrated, $\hbar_+$'s dataset merges with that of the selected holon, $\hbar_*^l$, necessitating a retraining on the aggregated dataset. If no appropriate holons are available at a required level, the system may initiate reholonification, integrating $\hbar_+$ with the set $\{\hbar_1^l, \cdots, \hbar_{N^l}^l\}$. A pseudo-code is provided in Algorithm 5.

**Algorithm 5** Integration of a New Holonic Unit into a Holarchy

**Require:** $\mathcal{H}^L$: $L$-level learning holarchy
**Require:** $\hbar_+$: A holonic unit
**for** $l = L$ downto 1 **do**
    ▷ **Identify sub-holons whose training sets do not exceed budget constraints**
    FreeHolons $\leftarrow \hbar^l : |\mathcal{T}^l \cup \mathcal{T}_+| \leq B^l$
    **if** FreeHolons $= \emptyset$ **then**
        Reholonification with the set $\{\hbar_1^l, \cdots, \hbar_{N^l}^l\} \cup \hbar_+$.
        **break**
    **end if**
    ▷ **Select the optimal sub-holon for integration**
    FreeModels $\leftarrow$ FreeHolons's models
    $\theta_*^l \leftarrow \arg\max_{\theta^l \in \text{FreeModels}} f\left(\theta^l, \mathcal{V}_+\right)$
    Update model parameters $\theta_*^l$ using $\mathcal{T}_*^l \cup \mathcal{T}_+$.
    ▷ **Integrate $\hbar_+$ into holon of $\theta_*^l$**
    $\text{SUB}(\hbar_*^l) \leftarrow \text{SUB}(\hbar_*^l) \cup \hbar_+$
**end for**

*Remark* 6.3. The cost of holonification is compared to integration on-the-fly based on the amount of communication between the holons. It is build on a single linkage-Hierarchical Clustering, with a complexity of $\mathcal{O}\left(N^2\right)$ [119]. On the other hand, the on-the-fly mechanism has a $\mathcal{O}(N + L)$. This corresponds to the worst-case scenario in which the new agent is compared to

all holons from the upper layer $L$ to layer 0. This mechanism thus offers a cost-effective integration in comparison with a reholonification.

### 6.4.3 Materials and Methods

Two public datasets that included footage from cameras in cities are used. Although they were initially intended for other tasks, they were selected for their realism and compatibility with our use case. As the number of cameras for the two combined datasets is 16, this value is used as the number of holons in the first level of our holarchies, that is, $N = 16$ for the remainder of this article. Namely, we augmented the WALT dataset introduced in Section 4.3.1 with 7 additional cameras from AI-CITY [1]. Details about the AI-CITY dataset can be found in annex Section 6.8. For the base layers, the SBAD model training defined in Section 4.3.3 was used with $B_0 = B = 256$ and `Top-Confidence` for each device. For any model, at any layer, the training is conducted at constant complexity of 10,000 iterations according to Eq. 5.4 with a learning rate of 0.01. The outcomes are presented in Table 6.2.

### 6.4.4 Model Training And Holonification Baseline

We conducted a holonification, as proposed in Section 6.4.1, on a dataset comprising sixteen cameras. We set a multi-layer budget framework $B^l = 256 \cdot 10^l$, implying that layer 0 units do not exceed 256 training samples, layer 1 is limited to 10 units, and layer 2 accommodates up to 100 units.

**Table 6.2** Holonification results, including the average mAP50-95 across the initial population of sixteen cameras.

| Layer | $B^l$ | mAP50-95 | Holonic Structure |
|---|---|---|---|
| 2 | 25600 | 0.65 | $\mathcal{H}^2 : \{\hbar_1^1, \hbar_2^1, \hbar_3^1\}$ |
| 1 | 2560 | 0.66 | $\mathcal{H}_1^1 : \{\hbar_1^0, \hbar_2^0, \hbar_3^0, \hbar_9^0\}$<br>$\mathcal{H}_2^1 : \{\hbar_4^0, \hbar_5^0, \hbar_6^0, \hbar_7^0, \hbar_8^0\}$<br>$\mathcal{H}_3^1 : \{\hbar_{16}^0, \hbar_{17}^0, \hbar_{18}^0, \hbar_{19}^0, \hbar_{20}^0, \hbar_{22}^0, \hbar_{24}^0\}$ |
| $\text{YOLOv8n}^{\text{COCO}}$ | N.A | 0.498 | N.A |

### 6.4.5 Incremental Integration

We assessed the impact of integrating $N_+$ new units into a holonified system, structured with budget limits of $B^l = 256 \cdot 10^l$. Using our integration mechanism described in Algorithm 5, we evaluated two scenarios: integrating one ($N_+ = 1$) and three ($N_+ = 3$) additional agents.

For statistical reliability, these experiments were repeated across sixteen different configurations of $N$ agents. The results indicate that the average mAP50-95 on the cameras test set was $0.66 \pm 0.003$ for $N_+ = 1$ and $0.66 \pm 0.006$ for $N_+ = 3$, demonstrating consistency with the baseline established in Table 6.2. However, the current experiments do not provide conclusive evidence regarding the maximum number of agents that can be integrated without a performance decline, suggesting the need for further investigation.

### 6.4.6 Re-training upon Agent Departure

We conducted a study to evaluate data management strategies in response to the departure of agents from a holonic system, specifically targeting subholons $\hbar_i^0$ from $\hbar^2$ (developed in Section 6.4.4). To understand the impact of agent departures, we simulated the sequential exit of each agent $i = 1, \cdots, 16$ and assessed $\hbar^2$'s model accuracy response under two scenarios:

1. **No retraining**. This implies that the data is kept.
2. **Dataset Exclusion and Retraining**. Upon the departure of an agent, their dataset $\mathcal{T}_i^0$ removed from the collective dataset $\mathcal{T}^2$, and the model is retrained for 10,000 iterations at a learning rate of 0.005. This process is repeated until $\mathcal{T}^2$ becomes empty.

Fig. 6.5 displays the difference in model accuracy for sensors that departed (yellow curve) and those that remained (blue curve) with the no-retraining scenarios.

- There was minimal improvement in the accuracy of the remaining sensors, more pronounced only as the number of exiting holons increased.
- Initially, there was a graceful degradation in performance for sensors that departed, which became more abrupt as more data was excluded.

These results confirm that while lightweight models benefit from specialization, this comes at the cost of generalization ability. At this point in our research, we cannot definitively recommend whether to retain or delete data. This decision should be considered as a strategic tool to manage a "controlled shift" towards the new network composition, necessitating further assumptions about the turnover rate of agents.

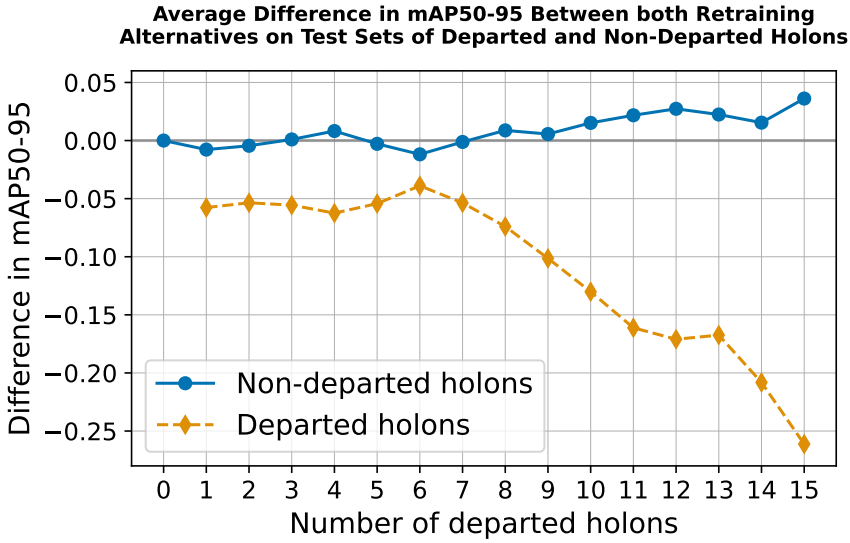



**Fig. 6.5** Performance difference of $\hbar^2$'s model between the baseline of keeping the data and the one that discards the data. The blue curve illustrates difference performance for remaining sensors between discarding and the baseline, while the yellow curve is the performance difference associated with departed sensors. Results show that gains are more substantial when many sensors exited, with a lower performance on the departed agents' dataset.

## 6.5 Insights and Limitations

This chapter aimed to provide an holonic modeling of a multi-method, multi-camera system. We leveraged organizations, agentification, hierarchy, multi-scale modeling presented in the introduction, to experiment

how they could relevantly be applied in our system.

The experiment in Section 6.3 utilized the multi-scale hierarchical modeling and agent proposed in Section 6.2.2, along with a system incorporating a portfolio of methods. By dynamically selecting the optimal processing technique based on the context's density, the system reduces resource consumption without compromising vehicle tracking quality. Notably, this is accomplished without the overhead associated with holon interactions.

From a learning perspective, we modeled a multi-tiered structure with self-organization mechanisms in Section 6.4. We then explored the impact of regularly adding or removing sensors in that system. Although the proposed behaviors demonstrate improvement compared to the baseline, they are not intended to be definitive in controlling open multi-sensor, multi-method systems. For example, while Section 6.4.5 suggests the method performs well, we have not sufficiently stressed the system to determine its limitations. Conversely, Section 6.4.6 indicates that the departure of an agent should be considered in terms of the agent's turnover rate, particularly in cases where the camera or environment is expected to return. We need to further stress test the system; that is, starting with a system of size $N$, stress tests can evaluate how many new components ($N_+$) can be integrated without compromising service quality.

This chapter's limitations are symptomatic of the lack of a comprehensive comparative protocol **to study the impact and necessity of such architecture on system scalability**. A well-defined industrial use case, with a set of metrics to evaluate the impact of agent designs on computer vision practices, would better illustrate how holonic paradigms can improve system adaptability and scalability—especially in scenarios where camera systems require autoscaling and autotuning [115]. However, at this stage, this chapter should be considered an introductory discussion on Holonic Multi-Agent Vision Systems (HMASVS).

## 6.6 Perspectives

### 6.6.1 Customization Based on End-Users

Our multi-sensor, multi-method architecture can be **augmented with a multi-user dimension**, aiming to meet the unique requirements of each end-user. This could allow personalized processing and alerts tailored to

specific applications, such as industrial, security, or traffic management. More precisely, a **holonic agent could assess its user's profile**, dynamically steering the system's behavior based on it. For instance, the system could adjust detection methods and data selection frequency based on risk profile: a high-risk profile might activate multiple detection methods and increase sampling, while a low-risk profile could opt for a streamlined, resource-efficient approach.

### 6.6.2 Inter-Holonic Knowledge Transfer

The holarchy in Section 6.2.2 proposes a vertical cascade of knowledge transfer where intermediate layers could act as both *Teachers* to lower layers and *Students* from higher layers. This can be seen as a generalization of the **source-free** distillation scheme discussed in Section 3.1. Yet, a **source-based** distillation, where a holon transfers knowledge at the same level can be a powerful **horizontal expansion**.

To support such a claim, we setup a series of sixteen trials, each excluding a different camera from the dataset to simulate a new node. Subsequently, some *Students* were fine-tuned on the new node dataset, to which they had not been previously exposed. The *Students* evaluated included:

- $\theta^2$: Trained on all fifteen cameras.
- $\theta^1_*$: Model from the cluster, which would integrate the camera post-clustering.
- $YOLOv8n^{\text{COCO}}$: A general-purpose object detection model.

Figure 6.6 illustrates fine-tuning performance across epochs, measured by the average mAP50-95. Training spanned 5 epochs with a learning rate of 0.005.

Results show that $\theta^2$ or $\theta^1_*$ models enable faster adaptation and even increase the peak accuracy of the new *Student* compared to the general-purpose YOLOv8n$^{\text{COCO}}$. This demonstrates the potential of source-based TTA, and more generally inter-holonic knowledge transfer, to accelerate learning and increase peak accuracy.

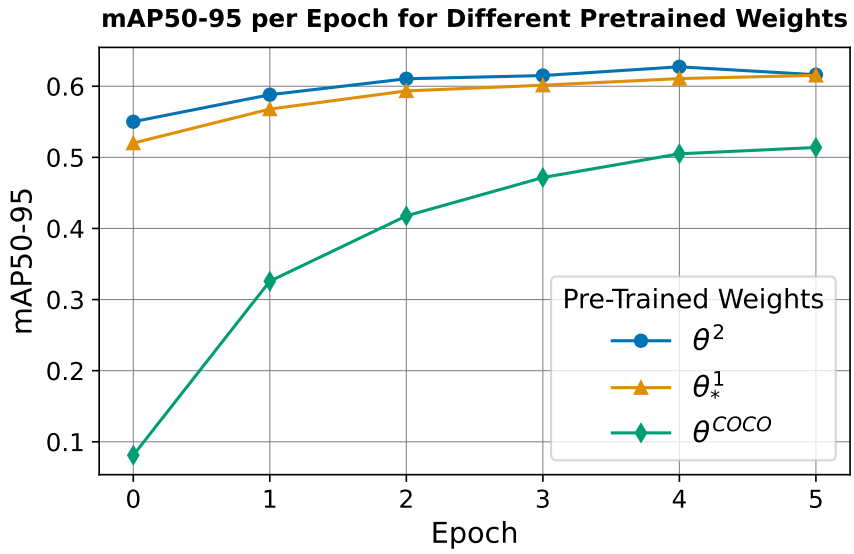



**Fig. 6.6** mAP50-95 per epoch for a new model starting from universal model $\theta^2$, group-specific $\theta^1_*$, and general-purpose $\theta^{\text{COCO}}$. The superiority of $\theta^2$ highlights the efficiency of selecting a pretrained model closer to the source.

## 6.7 Conclusions

We proposed a system in which each component (sensors, methods) is endowed with autonomy, allowing them to make decisions without referencing a central system, and interaction skills, enabling them to request and provide observations. Given the intrinsic complexity of such a surveillance system, we selected the organizational and holonic multi-agent paradigm for modeling. The different organizations and roles of the system are highlighted (Section 6.1) and described, as well as the interactions and role behaviors (Section 6.2). To validate the proposed model, a city-scale vehicle tracking system consisting of multiple cameras at single or multiple intersections across a city was considered (Section 6.3). In Section 6.4.1 we argued for a distributed holonic learning design based on a specificity-diversity trade-off to address the challenges of integrating and managing the deployment of deep neural networks within scalable sensor networks.

The experimental results in Sections 6.3 and 6.4 demonstrate the practicality and efficiency of our proposed mechanisms. However, we are at an early stage and must work towards more scalable open sensor systems to provide reliable design, tuning, and debugging to ensure control in uncertain environments and to mitigate costs associated with failure. This also includes further circumscription of the scope and high-level comparative tools to assess such designs that are convincing for the agent as well as the computer vision community.

## 6.8 Annex - Additional AI-CITY Dataset

This dataset complemented WALT dataset for the experiment in Section 6.4. It was annotated by us and they can be found at `https://uclouvain-my.sharepoint.com/:f:/g/personal/dani_manjah_uclouvain_be/Eo8rhx1M829BhauqoX7mHxgBBuOA_nKRgDfHUqVqLfazYA?e=Izr6gJ`. Samples from the set can be found in Fig. 6.7.

**(a)** s05c016

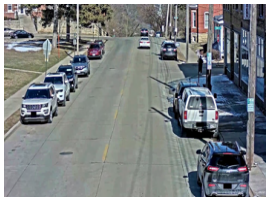
**(b)** s05c017

**(c)** s05c018

**(d)** s05c019

**(e)** s05c020

**(f)** s05c021



**(g)** s05c022

**(h)** s05c024

**Fig. 6.7** All Cameras in AI-city

# 7

# Discussion, Insights, Limitations and Conclusion

## 7.1 Are We Scalable Now?

Scalability is a multifaceted challenge, addressed at various levels—from individual camera operations to system-wide and broader systems of systems. This section summarizes the thesis's contributions toward developing more scalable camera systems.

### 7.1.1 Camera-level

The SBAD methodology drastically reduces annotation and training costs, substantially lowering the high expenses associated with human annotation. To illustrate this, let us estimate the annotation cost in both dollars and time for the test set in Section 4.3.1, which consists of 1,850 images. Assuming a conservative cost of $0.25 per image [12], and an average human annotation time of 6.84 seconds per image, the total cost would be approximately 4 hours of work and $462.50. In comparison, using a YOLOv8x6 model on A100 TensorRT, with an inference time of 3.53 ms per image [10], the entire test set could be pseudo-labeled in just 6 seconds at a cost of $0.01, given a GPU rental rate of $6.50 per hour [126]. Thus,

pseudo-labeling is around 50,000 times more cost-effective and 2,000 times faster than human annotation. A summary of this example is provided in Table 7.1.

Additionally, SBAD crafts powerful lightweight DNNs that outperform the *Teacher* model while remaining compact enough for deployment on low-cost hardware [10]. Furthermore, SBAD does not require local image storage. Comparisons with human annotators also revealed no performance drop due to pseudo-annotation.

**Table 7.1** Comparison of annotation costs and times between YOLOv8x6 and Human.

| | **YOLOv8x6** | **Human** (*lower bound*) |
|---|---|---|
| **Annotation Time per Image [ms]** | 3.53 | 6,840[1] |
| **Total Annotation Time [s]** | 6.53 | 12,654 |
| **Cost per Image [$]** | <0.01 | 0.25 |
| **Total Cost [$]** | 0.01 | 462.5 |

### 7.1.2 System-Level

At the system level, our clustering approach enhances scalability by addressing the limitations of both universal and specific models. Training a universal model on an ever-expanding dataset requires sophisticated infrastructure and complex distributed training, introducing new challenges [33]. Conversely, a specific approach multiplies models, complicating their monitoring [14]. Our clustering method strikes a balance between these extremes, offering a scalable solution that enhances predictive power beyond both universal and specific models.

Additionally, this domain-agnostic clustering method avoids assumptions about data stream correlations. This allows for automated clustering without human intervention—an investment that, despite higher computational costs, is worthwhile [14]. Furthermore, the method remains scalable since the cost of adding an increment increases linearly [127]. When integrating a new camera, with the previous matrix already calculated, only $N$ evaluations are required to update the $(N+1) \times (N+1)$ cross-performance matrix. After integration, as the amount of clusters is $K$, at most $K$ models may need retraining.

### 7.1.3 System of Systems

Our research on scalable design is still in its early stages, and a definitive approach has yet to be established. Advancing this work requires developing a test framework relevant to both design and computer vision. From a design perspective, the framework should evaluate whether the design is complex enough to handle real-world challenges while integrating seamlessly into industrial practice. From a computer vision standpoint, it should assess whether the sophistication of our approaches justifies their implementation based on the benefits. The framework should also include stress-testing the system. Starting with a system of size $N$, stress tests can evaluate how many new components ($N_+$) can be integrated without compromising service quality. Both quantitative and qualitative analyses are necessary.

### 7.1.4 Conclusion

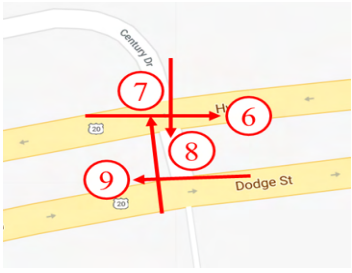
This thesis provides tools and strategies to improve scalability, acknowledging that it remains an ongoing challenge. Achieving scalability demands setting relevant goals, assessing progress, and continually adapting to new challenges.

*Remark* 7.1. I would like to offer a personal opinion. Scalability should be considered in a nuanced way, tailored to the specific scale being targeted. For instance, when labeling a method as scalable, it should be accompanied by a clear indication of the scale, such as SCAL$^3$ to denote a guarantee of linear cost evolution up to $10^3$ elements (*i.e.*, 1,000 elements). This scale-specific approach clarifies the methods' scope and applicability and opens new research avenues to extend scalability to larger scales.

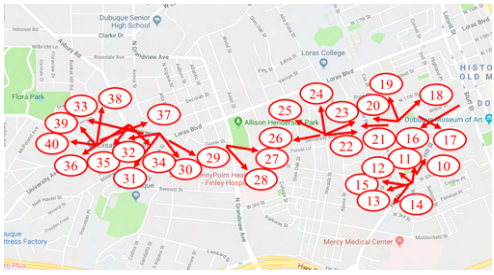
## 7.2 Qualitative Sustainability Considerations

This thesis posits that expanding camera networks, by increasing coverage and installing redundant sensors, enhances accuracy and resilience against device faults and occlusions (illustrated in Fig. 7.1a and Fig. 7.1b). We motivate such an expansive sensor network with the use case of traffic analysis. Specifically, such a system could more accurately assess traffic and optimize flow, potentially lowering carbon emissions by reducing congestion.

However, sensor systems present a double-edged sword for the environment: while they offer significant benefits, they can also have detrimental environmental effects [128]. The environmental impacts generated throughout a system's life cycle must be considered, as these could offset the net benefits. In fact, this could call for reevaluating optimal camera deployment, **balancing between deploying more sensors and maintaining positive environmental indicators.**




**(a)** Cameras observing the same crossroad, increasing system resilience in case of device faults or occlusions [1, 34].




**(b)** A camera network tracking vehicles across intersections, aiding traffic engineers in understanding journey times, optimizing signal timing, and mitigating congestion [1].

**Fig. 7.1** Surveillance frameworks enable data-driven urban governance, facilitating the conception of mobility plans or monitoring threats (*e.g.*, loitering, abandoned objects, etc.). However, the environmental impacts generated throughout the system's life cycle—including production, operation, and disposal of electronics—must be considered, as these could offset the benefits.

Life Cycle Assessment (LCA) is an environmental framework focusing on direct environmental effects of a product or service. It encompasses the entire life cycle of electronics, from material extraction and production to operation and waste management [129]. In our case, it involves camera hardware and the training infrastructure for DNNs.

LCA is a **quantitative approach** where the **total impact can be obtained by summing the impacts of each sub-component** [130]. Furthermore, LCA is a standardized ISO 14040 methodology, facilitating consensus about defining and measuring the impact [130]. Nevertheless, it does not capture indirect effects (*e.g.*, optimization benefits of more scalable methods) and higher-order effects (*e.g.*, rebound effects). These effects require investigation using other methodologies, which can be challenging

due to the multidimensional nature of environmental frameworks [131].

Despite these limitations, by focusing on direct effects, **LCA** helps **untangle environmental sustainability considerations, providing an initial but foundational perspective**. Its quantitative nature also presents an opportunity to explore the intersection of environmental sustainability (focused on direct effects) and scalable camera networks. Readers interested in conducting further LCA studies can refer to Thibaut Pirson's thesis on environmental impacts [131].

## 7.3 Insights and Take Away Messages

### 7.3.1 Deploy Lightweight Models with Balanced Data Specificity-Diversity

Our findings support practitioners to employ lightweight specialized *Students* rather than large generalist models (*i.e.*, YOLOv8x trained on COCO). Yet, our key message is that such specialization should balance between features specificity and diversity.

For researchers, this topic triggers an interesting question about predicting and managing this trade-off within clustering strategies. Especially when clusters composition evolves over time based on changes in the operational environment (*i.e.*, addition and deletion of cameras).

### 7.3.2 Prioritize Data Quality Over Annotation Quality

Our experiments indicate that **sampling quality** plays a primary role in *Student* performance, while **annotation quality** is of secondary importance (see Table 4.1). This indicates to practitioners and researchers **to prioritize the constitution of an informative training set, before investing in training larger and more complex *Teacher* models.** As discussed in Section 3.2.1, techniques from the literature on key frame selection can inspire the design of sampling strategies.

### 7.3.3 Agents and Computer Vision: A Marriage Worth Considering?

Our initial exploration of holonic agents within computer vision reveals a dichotomy between the two fields due to the distinct methodologies and

standards. Computer vision communities focus on image processing algorithms surpassing standardized benchmarks on narrowly-defined problems in object recognition, segmentation, or tracking. Separately, agentists' concerns are how agents perceive, interpret, and act upon information within an environment.

In other words, agent-based research views methods as a "black box" serving a larger purpose within a system of interacting agents. Similarly, computer vision scientists may see agent-based organizational questions as higher-level implementations beyond their scope. Consequently, a joint work faces challenges like:

1. **Differing evaluation metrics:** What constitutes a significant contribution in one field might be considered trivial or irrelevant in another.
2. **Language and terminology barriers:** Each field has its own jargon, which can make communication and understanding difficult.
3. **No publication venues:** There's a scarcity of forums that equally value contributions spanning both domains.

The introduction of a hybrid *Multi-Agent Vision Processing* framework can help researchers at the intersection of those fields, provided that such venture adds value. As a future work, we propose the writing of a position paper identifying use cases as well as highlighting the complementarity of both fields to tackle them.

### 7.3.4 A Portfolio of Methods For Greater Flexibility

The choice and development of computer vision algorithms, rely on their prediction capacity outperforming other existing methods. The focus on DNNs is symptomatic of this race for performance [132].

However, there is no free lunch; put another way, DNNs, like any other method, have drawbacks. For instance, the interpretability of the deep features is complex [133] and their decisions cannot be easily understood by an expert [134]. Overlooking other methods deprives us of providing a suitable answer for a particular problem as underscored in Section 6.3. Note that research on handling heterogeneous and complementary methods remains immature [135] and remains a great research direction.

## 7.4 Summary

This thesis presents the CSBAD framework and a holonic architecture designed to address the economic and engineering challenges associated with large-scale and ever-growing video analytics systems.

Chapter 4 explores the SBAD framework, which leverages a *Teacher-Student* approach to minimize dependence on human-annotated data and customizes training for specific environments. The `Top-Confidence` strategy, where the *Student* selects high-confidence images, effectively reduces the annotation and training costs by decreasing the necessary training set size.

Chapter 5 extends SBAD into a multi-camera environment and develops its clustered version CSBAD. CSBAD advocates for the clustering of camera domains to move beyond the constraints of specialized or universal networks, enhancing adaptability and performance. The introduction of $K$, a variable representing the number of clusters, is crucial for adapting CSBAD to diverse system configurations. It also helps balance specificity-diversity taking into account the sample volume to boost peak performance.

Chapter 6 conceptualizes the camera network as a holarchy of autonomous yet interdependent agents. This structure facilitates dynamic adaptation to changes in network size, including integration and removal of sensors without extensive reconfiguration. A limitation of this work, which also constitutes a future research direction, would be to develop a comparative tool to evaluate the design not only from a conceptual point of view, but also whether it can be easily distilled into industrial practices.

In conclusion, for practitioners, CSBAD decreases the computational cost of managing and adding cameras, thereby minimizing infrastructure demand and preventing system overload. Furthermore, with the use of lightweight DNNs, we bolster applications built on edge devices without sacrificing accuracy. In addition, our holonic vision lays foundational work for pushing scalability further by reducing technical debt and improving workload distribution. These contributions serve as a foundation for future innovations in intelligent and scalable surveillance, setting the stage for the next generation of urban and industrial monitoring systems.

**Thank you for reading!**

# List of Publications

1. D. Manjah, D. Cacciarelli, B. Standaert, M. Benkedadra, G. R. de Hertaing, B. Macq, S. Galland, and C. De Vleeschouwer, "Stream-based active distillation for scalable model deployment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4998–5006, June 2023.
2. D. Manjah, D. Cacciarelli, C. De Vleeschouwer, and B. Macq, "Camera clustering for scalable stream-based active distillation," *arXiv:2404.10411*, 2024.
3. D. Manjah, S. Galland, C. De Vleeschouwer, and B. Macq, "Autonomous Methods in Multisensor Architecture for Smart Surveillance," in *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, 2024, pp. 823–831
4. D. Manjah, K. Hagihara, G. Basso, and B. Macq, "Implementation of a Holonic Multi-agent System in Mixed or Augmented Reality for Large Scale Interactions," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, 2020, pp. 447–450, Springer.
5. G. R. de Hertaing, D. Manjah, and B. Macq, "TrajViViT: A Trajectory Video Vision Transformer Network for Trajectory Forecasting," in *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM*, 2024, pp. 753–760,
6. (*SOFTWARE*) D. Manjah, "Aptitude," GitHub repository, 2024. Link: `https://github.com/manjahdani/aptitude`
7. (*SOFTWARE*) D. Manjah, "CSBAD," GitHub repository, 2024. Link: `https://github.com/manjahdani/CSBAD`

# Bibliography


[1] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, X. Yang, Y. Yao, L. Zheng, P. Chakraborty, C. E. Lopez, A. Sharma, Q. Feng, V. Ablavsky, and S. Sclaroff, "The 5th ai city challenge," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

[2] S. Okamoto, P. Scerri, and K. P. Sycara, "The impact of vertical specialization on hierarchical multi-agent systems.," in *AAAI*, pp. 138–143, 2008.

[3] J. Á. Román, S. Rodríguez, and J. M. Corchado, "Distributed and specialized agent communities," in *Trends in Practical Applications of Agents and Multiagent Systems: 11th International Conference on Practical Applications of Agents and Multi-Agent Systems*, pp. 33–40, Springer, 2013.

[4] J. A. Román, S. Rodríguez, and J. M. Corchado, "Improving intelligent systems: Specialization," in *Highlights of Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, (Cham), pp. 378–385, Springer International Publishing, 2014.

[5] C. Bucilu, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, (New York, NY, USA), p. 535–541, Association for Computing Machinery, 2006.

[6] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam, "Aspecs: an agent-oriented software process for engineering complex systems," *Autonomous Agents and Multi-Agent Systems*, vol. 20, no. 2, pp. 260–304, 2010.

[7] M. Feraud and S. Galland, "First comparison of sarl to other agent-programming languages and frameworks," *Procedia Computer Science*, vol. 109, pp. 1080 – 1085, 2017.

[8] B. Rao, H. F. Durrant-Whyte, and J. Sheen, "A fully decentralized multi-sensor system for tracking and surveillance," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 20–44, 1993.

[9] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by Internet of Things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[10] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023.

[11] D. Yin, A. Pananjady, M. Lam, D. Papailiopoulos, K. Ramchandran, and P. Bartlett, "Gradient diversity: a key ingredient for scalable distributed learning," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, vol. 84 of *Proceedings of Machine Learning Research*, pp. 1998–2007, PMLR, 2018.

[12] K. Technology, "Data labeling services price q3 2023 benchmark," 2023. Accessed: 12/10/2023.

[13] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum self-paced learning for cross-domain object detection," *Computer Vision and Image Understanding*, vol. 204, p. 103166, 2021.

[14] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.

[15] J. Beal, M. Viroli, D. Pianini, and F. Damiani, "Self-adaptation to device distribution changes," in *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pp. 60–69, 2016.

[16] A. Diaconescu, S. Frey, C. Müller-Schloer, J. Pitt, and S. Tomforde, "Goal-oriented holonics for complex system (self-)integration: Concepts and case studies," in *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pp. 100–109, 2016.

[17] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[18] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier, "Cluster-based scalable network services," in *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, SOSP '97, (New York, NY, USA), p. 78–91, Association for Computing Machinery, 1997.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[20] S. Rodriguez, V. Hilaire, N. Gaud, S. Galland, and A. Koukam, "Holonic Multi-Agent Systems," *Natural Computing Series*, vol. 37, pp. 251–279, 2011.

[21] H. A. Abbas, "Organization of multi-agent systems: An overview," *International Journal of Intelligent Information Systems*, vol. 4, no. 3, p. 46, 2015.

[22] M.-P. Gleizes, "Self-adaptive complex systems," in *Multi-Agent Systems*, (Berlin, Heidelberg), pp. 114–128, Springer, 2012.

[23] Y. Wautelet, C. Schinckus, and M. Kolp, "Agent-based software engineering, paradigm shift, or research program evolution," in *Research Anthology on Recent Trends, Tools, and Implications of Computer Programming*, pp. 1642–1654, IGI Global, 2021.

[24] A. Koestler, *The ghost in the machine*. London, UK: Hutchinson, 1967.

[25] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution," *Computers in Industry*, vol. 66, pp. 99–111, 2015.

[26] A. Esmaeili, Z. Ghorrati, and E. T. Matson, "Holonic learning: A flexible agent-based distributed machine learning framework," in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, (Richland, SC), p. 525–533, International Foundation for Autonomous Agents and Multiagent Systems, 2024.

[27] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[28] K. El Khoury, J. Samelson, and B. Macq, "Deep learning-based object tracking via compressed domain residual frames," *Frontiers in Signal Processing*, vol. 1, 2021.

[29] K. El Khoury, T. Godelaine, S. Delvaux, S. Lugan, and B. Macq, "Streamlined hybrid annotation framework using scalable codestream for bandwidth-restricted uav object detection," in *2024 IEEE International Conference on Image Processing (ICIP)*, pp. 1581–1587, 2024.

[30] D. Manjah, D. Cacciarelli, B. Standaert, M. Benkedadra, G. R. de Hertaing, B. Macq, S. Galland, and C. De Vleeschouwer, "Stream-based active distillation for scalable model deployment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4998–5006, 2023.

[31] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2020.

[32] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[33] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[34] Y.-C. Liu, J. Tian, N. Glaser, and Z. Kira, "When2com: Multi-agent perception via communication graph grouping," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[35] J. J. Valencia-Jiménez and A. Fernández-Caballero, "Holonic multiagent systems to integrate independent multi-sensor platforms in complex surveillance," *Proceedings - IEEE International Conference on Video and Signal Based Surveillance 2006, AVSS 2006*, pp. 49–54, 2006.

[36] H. A. Simon, "How complex are complex systems?," *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, vol. 1976, no. 2, p. 507–522, 1976.

[37] M. Abdelrahim, J. M. Hendrickx, and W. P. Heemels, "MAX-consensus in open multi-agent systems with gossip interactions," *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, vol. 2018-January, pp. 4753–4758, 2018.

[38] J. M. Hendrickx and S. Martin, "Open multi-agent systems: Gossiping with random arrivals and departures," *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, vol. 2018-January, pp. 763–768, 2018.

[39] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, p. 115–152, 1995.

[40] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[41] F. Yan, S. Sundaram, S. V. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2013.

[42] M. Kolp, P. Giorgini, and J. Mylopoulos, "Multi-agent architectures as organizational structures," *Autonomous Agents and Multi-Agent Systems*, vol. 13, pp. 3–25, 2006.

[43] M. Schatten, P. Grd, M. Konecki, and R. Kudelić, "Towards a formal conceptualization of organizational design techniques for large scale multi agent systems," *Procedia Technology*, vol. 15, pp. 576–585, 2014. 2nd International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering.

[44] E. Garcia, E. Argente, A. Giret, and V. Botti, "Issues for organizational multiagent systems development," in *Sixth International Workshop From Agent Theory to Agent Implementation (AT2AI-6)*, pp. 59–65, Citeseer, 2008.

[45] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard, "ADELFE: a methodology for adaptive multi-agent systems engineering," in *Engineering Societies in the Agents World III*, vol. 2577 of *Lecture Notes in Computer Science*, (Madrid, Spain), pp. 156–169, Springer, 2002.

[46] M. Wooldridge, N. R. Jennings, and D. Kinny, "The gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000.

[47] J. Pavón and J. Gómez-Sanz, "Agent oriented software engineering with ingenias," in *Multi-Agent Systems and Applications III*, (Berlin, Heidelberg), pp. 394–403, Springer, 2003.

[48] A. Omicini, "Soda: Societies and infrastructures in the analysis and design of agent-based systems," in *Agent-Oriented Software Engineering*, pp. 185–193, Springer Berlin Heidelberg, 2001.

[49] J. Castro, M. Kolp, and J. Mylopoulos, "Towards requirements-driven information systems engineering: the tropos project," *Information Systems*, vol. 27, no. 6, pp. 365–389, 2002.

[50] M. Ulieru and R. A. Este, "The holonic enterprise and theory emergence: On emergent features of self-organization in distributed virtual agents," *Cybernetics & Human Knowing*, vol. 11, no. 1, pp. 79–98, 2004.

[51] D. Manjah, K. Hagihara, G. Basso, and B. Macq, "Implementation of a holonic multi-agent system in mixed or augmented reality for large scale interactions," in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection*, (Cham), pp. 447–450, Springer International Publishing, 2020.

[52] D. Manjah, S. Galland, C. D. Vleeschouwer, and B. Macq, "Autonomous methods in multisensor architecture for smart surveillance," in *Proceedings of the 16th International Conference on Agents and Artificial Intelligence*, vol. 3, pp. 824–832, 2024.

[53] A. Esmaeili, N. Mozayani, M. R. J. Motlagh, and E. T. Matson, "The impact of diversity on performance of holonic multi-agent systems," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 186–201, 2016.

[54] A. Esmaeili, N. Mozayani, M. R. Jahed-Motlagh, and E. T. Matson, "Towards topological analysis of networked holonic multi-agent systems," in *Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection*, (Cham), pp. 42–54, Springer International Publishing, 2019.

[55] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *NIPS 2014 Deep Learning Workshop*, 2015.

[56] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, 2018.

[57] R. Mishra and H. P. Gupta, "Designing and training of lightweight neural networks on edge devices using early halting in knowledge distillation," *IEEE Transactions on Mobile Computing*, 2023.

[58] E. Tanghatari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Federated learning by employing knowledge distillation on edge devices with limited hardware resources," *Neurocomputing*, vol. 531, pp. 87–99, 2023.

[59] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Semi-supervised learning for domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019.

[60] J. Liang, R. He, and T. Tan, "A comprehensive survey on test-time adaptation under distribution shifts," *International Journal of Computer Vision*, pp. 1–34, 2024.

[61] J. Liang, D. Hu, Y. Wang, R. He, and J. Feng, "Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 8602–8617, 2022.

[62] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision – ECCV 2010*, (Berlin, Heidelberg), pp. 213–226, Springer Berlin Heidelberg, 2010.

[63] A. Cioppa, A. Deliege, M. Istasse, C. De Vleeschouwer, and M. Van Droogenbroeck, "Arthus: Adaptive real-time human segmentation in sports through online distillation," in *Proceedings of the*

*IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[64] R. T. Mullapudi, S. Chen, K. Zhang, D. Ramanan, and K. Fatahalian, "Online model distillation for efficient video inference," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3573–3582, 2019.

[65] D. Rivas, F. Guim, J. Polo, P. M. Silva, J. L. Berral, and D. Carrera, "Towards automatic model specialization for edge video analytics," *Future Generation Computer Systems*, 2022.

[66] A. Habibian, H. Ben Yahia, D. Abati, E. Gavves, and F. Porikli, "Delta distillation for efficient video processing," in *Computer Vision – ECCV 2022*, (Cham), pp. 213–229, Springer Nature Switzerland, 2022.

[67] K. Vilouras, X. Liu, P. Sanchez, A. Q. O'Neil, and S. A. Tsaftaris, "Group distributionally robust knowledge distillation," in *International Workshop on Machine Learning in Medical Imaging*, pp. 234–242, Springer, 2023.

[68] C. Baykal, K. Trinh, F. Iliopoulos, G. Menghani, and E. Vee, "Robust active distillation," *arXiv preprint arXiv:2210.01213*, 2022.

[69] M. Goldblum, L. Fowl, S. Feizi, and T. Goldstein, "Adversarially robust distillation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3996–4003, 2020.

[70] B. Settles, "Active learning literature survey," *Computer Sciences Technical Report, University of Wisconsin–Madison*, 2009.

[71] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *ICLR 2018*, 2017.

[72] D. Yoo and I. S. Kweon, "Learning loss for active learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[73] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep batch active learning by diverse, uncertain gradient lower bounds," in *2020 International Conference on Learning Representations*, 2019.

[74] E. Elhamifar, G. Sapiro, A. Yang, and S. S. Sasrty, "A convex optimization framework for active learning," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 209–216, Institute of Electrical and Electronics Engineers Inc., 2013.

[75] S. Agarwal, H. Arora, S. Anand, and C. Arora, "Contextual diversity for active learning," in *European Conference on Computer Vision (ECCV) 2020*, 2020.

[76] V. Prabhu, A. Chandrasekaran, K. Saenko, and J. Hoffman, "Active domain adaptation via clustering uncertainty-weighted embeddings," in *International Conference on Computer Vision (ICCV) 2021*, 2020.

[77] S. Sinha, S. Ebrahimi, and T. Darrell, "Variational adversarial active learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5972–5981, 2019.

[78] T. Yuan, F. Wan, M. Fu, J. Liu, S. Xu, X. Ji, and Q. Ye, "Multiple instance active learning for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[79] D. Cacciarelli, M. Kulahci, and J. S. Tyssedal, "Stream-based active learning with linear models," *Knowledge-Based Systems*, vol. 254, p. 109664, 2022.

[80] D. Cacciarelli, M. Kulahci, and J. S. Tyssedal, "Robust online active learning," *Quality and Reliability Engineering International*, vol. ENBIS Special Issue, 2023.

[81] D. Cacciarelli, M. Kulahci, and J. S. Tyssedal, "Online active learning for soft sensor development using semi-supervised autoencoders," in *ICML 2022 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2022.

[82] J. M. Rožanec, E. Trajkova, P. Dam, B. Fortuna, and D. Mladenić, "Streaming machine learning and online active learning for automated visual inspection," *14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022*, vol. 55, no. 2, pp. 277–282, 2022.

[83] A. Narr, R. Triebel, and D. Cremers, "Stream-based active learning for efficient and adaptive classification of 3d objects," in *Proceedings*

- *IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 227–233, Institute of Electrical and Electronics Engineers Inc., 2016.

[84] R. Savran Kızıltepe, J. Q. Gan, and J. J. Escobar, "A novel keyframe extraction method for video classification using deep neural networks," *Neural Computing and Applications*, vol. 35, no. 34, pp. 24513–24524, 2023.

[85] A. A. Gharahbagh, V. Hajihashemi, M. C. Ferreira, J. J. M. Machado, and J. M. R. S. Tavares, "Best frame selection to enhance training step efficiency in video-based human action recognition," *Applied Sciences*, vol. 12, no. 4, 2022.

[86] C. Dang and H. Radha, "Rpca-kfe: Key frame extraction for video using robust principal component analysis," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3742–3753, 2015.

[87] X. Yan, S. Z. Gilani, M. Feng, L. Zhang, H. Qin, and A. Mian, "Self-supervised learning to detect key frames in videos," *Sensors*, vol. 20, no. 23, 2020.

[88] A. Aner and J. R. Kender, "Video summaries through mosaic-based shot and scene clustering," in *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part IV 7*, pp. 388–402, Springer, 2002.

[89] S. Sun and X. Gong, "Hierarchical semantic contrast for scene-aware video anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22846–22856, 2023.

[90] Y. Wang, H. Wang, and X. Li, "An intelligent recommendation system model based on style for virtual home furnishing in three-dimensional scene," in *2013 International Symposium on Computational and Business Intelligence*, pp. 213–216, IEEE, 2013.

[91] Z. Li, R. Wang, H. Li, B. Wei, Y. Shi, H. Ling, J. Chen, B. Liu, Z. Li, and H. Zheng, "Hierarchical clustering and refinement for generalized multi-camera person tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5519–5528, 2023.

[92] G. Szűcs, R. Borsodi, and D. Papp, "Multi-camera trajectory matching based on hierarchical clustering and constraints," *Multimedia Tools and Applications*, pp. 1–24, 2023.

[93] X. Wang, K. Tieu, and W. E. L. Grimson, "Correspondence-free multi-camera activity analysis and scene modeling," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[94] A. Specker, D. Stadler, L. Florin, and J. Beyerer, "An occlusion-aware multi-target multi-camera tracking system," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4173–4182, 2021.

[95] T. Patel, A. Y. H. Yao, Y. Qiang, W. T. Ooi, and R. Zimmermann, "Multi-camera video scene graphs for surveillance videos indexing and retrieval," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 2383–2387, IEEE, 2021.

[96] C. Simon, J. Meessen, and C. De Vleeschouwer, "Visual event recognition using decision trees," *Multimedia Tools and Applications*, vol. 50, pp. 95–121, 2010.

[97] S. Peng, J. Yin, Z. Yang, B. Chen, and Z. Lin, "Multiview clustering via hypergraph induced semi-supervised symmetric nonnegative matrix factorization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 10, pp. 5510–5524, 2023.

[98] D. Huang, C.-D. Wang, and J.-H. Lai, "Fast multi-view clustering via ensembles: Towards scalability, superiority, and simplicity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11388–11402, 2023.

[99] H. D. Park, O. G. Min, and Y. J. Lee, "Scalable architecture for an automated surveillance system using edge computing," *Journal of Supercomputing*, vol. 73, no. 3, pp. 926–939, 2017.

[100] H. A. Abbas, S. I. Shaheen, and M. H. Amin, "Organization of multi-agent systems: an overview," *arXiv preprint arXiv:1506.09032*, 2015.

[101] A. R. Hilal and O. A. Basir, "A scalable sensor management architecture using BDI model for pervasive surveillance," *IEEE Systems Journal*, vol. 9, no. 2, pp. 529–541, 2015.

[102] N. Nezamoddini and A. Gholami, "A survey of adaptive multi-agent networks and their applications in smart cities," *Smart Cities*, vol. 5, no. 1, pp. 318–347, 2022.

[103] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[104] B. Porter and R. Rodrigues Filho, "Distributed emergent software: Assembling, perceiving and learning systems at scale," in *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pp. 127–136, 2019.

[105] Y. Yang, J. Luo, Y. Wen, O. Slumbers, D. Graves, H. Bou-Ammar, J. Wang, and M. E. Taylor, "Diverse auto-curriculum is critical for successful real-world multiagent learning systems," *CoRR*, vol. abs/2102.07659, 2021.

[106] S. Gherardi, "Learning: Organizational," in *International Encyclopedia of the Social & Behavioral Sciences (Second Edition)*, pp. 695–698, Oxford: Elsevier, second edition ed., 2015.

[107] M. Terabe, T. Washio, O. Katai, and T. Sawaragi, "A study of organizational learning in multiagents systems," in *Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments*, (Berlin, Heidelberg), pp. 168–179, Springer Berlin Heidelberg, 1997.

[108] J. Dong, R. Liu, Y. Qiu, and M. Crossan, "Should knowledge be distorted? managers' knowledge distortion strategies and organizational learning in different environments," *The Leadership Quarterly*, vol. 32, no. 3, p. 101477, 2021.

[109] A. Esmaeili, J. C. Gallagher, J. A. Springer, and E. T. Matson, "Hamlet: A hierarchical agent-based machine learning platform," *ACM Trans. Auton. Adapt. Syst.*, vol. 16, no. 3–4, 2022.

[110] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[111] B. Xu, W. Xia, W. Wen, P. Liu, H. Zhao, and H. Zhu, "Adaptive hierarchical federated learning over wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2070–2083, 2021.

[112] Z. Ma, Y. Xu, H. Xu, J. Liu, and Y. Xue, "Like attracts like: Personalized federated learning in decentralized edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1080–1096, 2024.

[113] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 19586–19597, Curran Associates, Inc., 2020.

[114] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 41–47, 2020.

[115] D. Weyns, I. Gerostathopoulos, N. Abbas, J. Andersson, S. Biffl, P. Brada, T. Bures, A. Di Salle, M. Galster, P. Lago, G. Lewis, M. Litoiu, A. Musil, J. Musil, P. Patros, and P. Pelliccione, "Self-adaptation in industry: A survey," *ACM Trans. Auton. Adapt. Syst.*, vol. 18, no. 2, 2023.

[116] D. Manjah, D. Cacciarelli, C. D. Vleeschouwer, and B. Macq, "Camera clustering for scalable stream-based active distillation," 2024.

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

[118] N. D. Reddy, R. Tamburo, and S. G. Narasimhan, "Walt: Watch and learn 2d amodal representation from time-lapse imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9356–9366, 2022.

[119] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, no. 1, pp. 30–34, 1973.

[120] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[121] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "Hota: A higher order metric for evaluating multi-object rracking," *International Journal of Computer Vision*, vol. 129, pp. 1–31, 2021.

[122] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020.

[123] Z. Luo, F. Branchaud-Charron, C. Lemaire, J. Konrad, S. Li, A. Mishra, A. Achkar, J. Eichel, and P.-M. Jodoin, "Mio-tcd: A new benchmark dataset for vehicle classification and localization," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5129–5141, 2018.

[124] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.

[125] J. Le, X. Lei, N. Mu, H. Zhang, K. Zeng, and X. Liao, "Federated continuous learning with broad network architecture," *IEEE Transactions on Cybernetics*, vol. 51, no. 8, pp. 3874–3888, 2021.

[126] H. Face, "Pricing," 2023. Accessed: [today's date].

[127] T. Hossfeld, P. E. Heegaard, and W. Kellerer, "Comparing the scalability of communication networks and systems," *IEEE Access*, pp. 1–1, 2023.

[128] L. M. Hilty and B. Aebischer, *ICT innovations for sustainability*, vol. 310. Springer.

[129] International Telecommunication Union (ITU), "Recommendation ITU-T L.1410: Methodology for environmental life cycle assessments of information and communication technology goods, networks and services." `https://www.itu.int/rec/T-REC-L.1410`, 2014. Online; accessed 24-October-2024.

[130] P. Teehan, *Integrative approaches to environmental life cycle assessment of consumer electronics and connected media*. PhD thesis, University of British Columbia, 2014.

[131] T. Pirson, *Environmental perspectives and limits for the Internet of Things in the Anthropocene: going beyond systematic Life-Cycle Assessment*. PhD thesis, UCL-Université Catholique de Louvain, 2023.

[132] H. H. Barrett, "Is there a role for image science in the brave new world of artificial intelligence?," *Journal of Medical Imaging*, vol. 7, no. 01, p. 1, 2019.

[133] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, pp. 818–833, Springer Verlag, 2014.

[134] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2016.

[135] A. Campagner, D. Ciucci, and F. Cabitza, "Aggregation models in ensemble learning: A large-scale comparison," *Information Fusion*, vol. 90, pp. 241–252, 2023.