UCLouvain
FUCaM MONS

LOUVAIN
School of Management

Louvain School of Management

Application of Model Chaining in Computer Vision for industrial environments with limited data diversity.

Author: Adrien Chen

Promoters: Mr. Vande Kerckhove – Mrs. Satinet

Academic Year: 2023-2024

Master in Management Engineering with a Specialization in Business Analytics

Declaration Regarding AI Tool Usage in Master's Thesis

We recognize that AI tools might be valuable aids during the master's thesis work, but they are not infallible. Remember that transparency fosters trust, and acknowledging AI's role enhances the credibility of your work.

Therefore, when deciding to use such a tool, you need to adhere to the following principles of responsible use of AI.

1. **Critical Evaluation :**
    - We critically assessed the AI-generated output, ensuring its alignment with our research objectives.
    - Any modifications or corrections were made based on our expertise and domain knowledge.
2. **Transparency :**
    - We acknowledge the use of [NAME TOOL / SERVICE] transparently, emphasizing that it contributed to our work but did not replace human judgment.
    - Our commitment to transparency ensures the integrity of this thesis.
3. **Ethical Considerations :**
    - We actively monitored for biases or unintended consequences introduced by the AI tool.
    - Our ethical responsibility guided our decisions throughout the research process.

Declaration **(This declaration is mandatory and must appear on the first page (after the title page) of the document.**

During the preparation of this master's thesis, the author(s) utilized [Chat GPT and Scribbr] for the following purpose:

1.Coding: Chat GPT represented a powerful help for the development of the computer vision sytems (for the coding part)

2.Writing: For the writing part of the thesis, Chat GPT was used as a translator and a grammatical corrector. Scribbr was used to generate the references in the APA format.

After using [Chat GPT and Scribbr], the author(s) diligently reviewed and edited the content produced by the tool. We take full responsibility for the final content presented in this thesis.

By signing this declaration, we affirm that the content of this master's thesis reflects our original work, augmented by the responsible use of AI.

I, Adrien Chen, hereby declare that I have used AI as a tool to enhance my productivity and the research I led. Declared on the 29th may 2024.

Abstract

In recent years, the integration of computer vision in industrial applications has significantly advanced, driven by the need for enhanced automation and efficiency. This thesis, completed in partnership with a Belgian company, aims to develop a computer vision solution for automating the detection and classification of truck loads in an industrial context characterized by limited data diversity.

An initial approach was established to directly detect the transported materials. However, this approach suffered from the lack of diversity in the available data, as the training images were predominantly from a single 360-degree camera. To address this issue, the thesis brings a model chaining approach, combining a truck bed detection model with a material classification model. This method leverages the strengths of separate models for detection and classification, each trained on different data to enhance system flexibility and performance.

Acknowledgements.

I would like to extend my deepest gratitude to my promoter, Mr. Vande Kerckhove, and my co-promoter, Mrs. Satinet, for their valuable guidance, patience, and support throughout this project. Their expert advice and encouragement were instrumental in the successful completion of this work.

I also wish to thank the data science team at Technord for entrusting me with this project, providing essential supervision and assistance, and equipping me with the necessary tools and resources. Their expertise and support have been vital to the progress and success of this academic paper.

Lastly, I am grateful to the researchers from the Department of Real Estate and Construction at the University of Hong Kong for providing the images that constituted the evaluation set. Their contribution was crucial for the advancement of my research.

Table of contents

List of Figures

List of Tables

List of Appendix

1. Introduction

In recent years, the integration of computer vision in industrial applications has gained significant momentum, driven by the need to enhance automation and efficiency in various sectors. It is in this rapidly developing environment favourable for research and driven by a desire to undertake a concrete project, that this thesis was completed in partnership with a company.

This thesis was developed tied up to a company's initiative to explore the development of a computer vision solution that can automate a process to detect and classify the type of load carried by a lorry in an industrial setting where the diversity of available data is limited. To realise this project, a first approach was realised by developing an object detection model to directly detect the transported materials. This first approach reached a precision of 74.26%. However, when it comes to testing it on an evaluation set composed of more diverse images of transported materials, a precision of 59.19% was achieved. This drop in performance is explained by a lack of diversity in the training data, as the images used to train the model were taken by the same 360-degree camera.

This dissertation proposes a different approach by using a model chaining computer vision system to compensate for this lack of variety in the available data. The question that remains is: Compared to our first approach (a single model of material detection), can it improve the performance of our system? **In an industrial context characterized by a lack of diversity in the available data, how can model chaining enhance the flexibility of a computer vision system for the detection and classification of transported materials?** To address this question, a model chaining approach was built by combining an object detection model to detect the truck bed that carries the materials, followed by a classification model to classify the materials inside the truck bed. This new approach outperforms the initial one by achieving a precision of 77.27% on an established evaluation set.

In the following chapters, we will review the literature to establish the foundations of object detection and the existing models before exploring the existing literature on the use of model chaining to enhance a computer system's performance. Following the literature review, we will dive into the methodology followed to build the initial approach and the solution proposed. Finally, we will test the final models corresponding to our two approaches.

.

2. Theoretical background and literature review.

In an industrial context characterized by a lack of variety in the available data, how can model chaining enhance the flexibility of a computer vison system for the detection and classification of transported materials? In this chapter, we will delve into this challenge by reviewing the foundations of machine learning before focusing on the convolutional neural networks that enable object detection and image recognition in the field of computer vision. We will then explore the existing research papers on automatic object detection models and research on model chaining systems to address problems such as the lack of diversity in the available data.

First, since we are looking to automate a process that involves predictions to detect and classify materials, we are transitioning from traditional artificial intelligence that uses any techniques to reproduce human intelligence, to using machine learning (ML) models, implicating a subset of ML called Deep learning until reaching the Convolutional Neural Networks (CNNs) that enable computer vision applications such as object detection. (See figure 1 as an illustration)

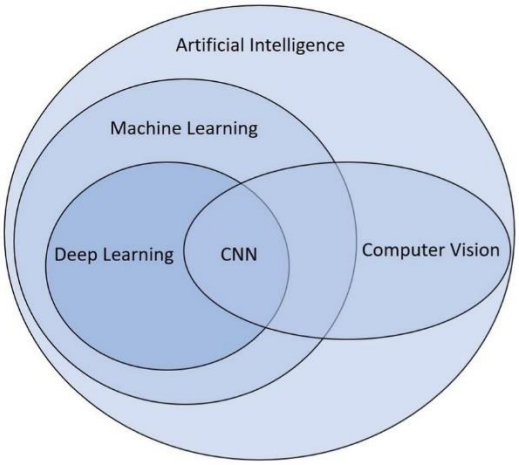


Figure 1: Diagram of artificial intelligence and neural networks in computer vision. (ReachIt, s. d.).

2.1 Machine learning and Computer vision

2.1.1 Machine learning

To understand computer vision techniques using CNNs and their philosophy, it is crucial to explore their origins. Machine learning is a computer science field focused on developing algorithms that enable computers to make predictions or decisions based on historical data. The

core idea involves training a model, that recognizes patterns and relationships within data. This training empowers the model to learn and make predictions. The methodology generally begins with collecting training data that includes examples of inputs and their corresponding outputs. Through its learning process, the model establishes connections between inputs and outputs, continually improving its accuracy with each iteration. (Baştanlar & Özuysal, 2013, p 105)

Several ways to tackle approaches of machine learning can be distinguished:

Supervised learning involves training models using labelled data, where each input is associated with a class label or output value. The training phase teaches the model to map inputs to desired outputs based on this labelled dataset. Generally, the process involves extracting features from labelled data, training a model using this data, and testing it on a new dataset to predict outcomes. This approach is particularly popular for classification tasks that require labelled images for training (Mahadevkar et al., 2022, p. 107299-107300).

Unsupervised learning, on the other hand, uses unlabelled data and requires the model to identify characteristics, patterns, and structures on its own (Mahadevkar et al., 2022, p. 107307). Without predefined labels, the model learns to group data based on inherent similarities, making this approach well-suited for clustering tasks or discovering hidden structures within datasets.

2.1.2 Deep learning and Convolutional Neural Network

Now that we have introduced the field of machine learning, we can go deeper into one of its subsets: deep learning. Deep learning models pursue the same objective of learning from data in order to output predictions or decisions with the specific characteristic that deep learning relies more on artificial neural networks. These neural networks are inspired by the way the human brain works and are made up of layers that each handle different aspects of the data processing (Figure 2). (Janiesch et al., 2021, p 685-688) (Sharifani et al., 2023, p 3898)

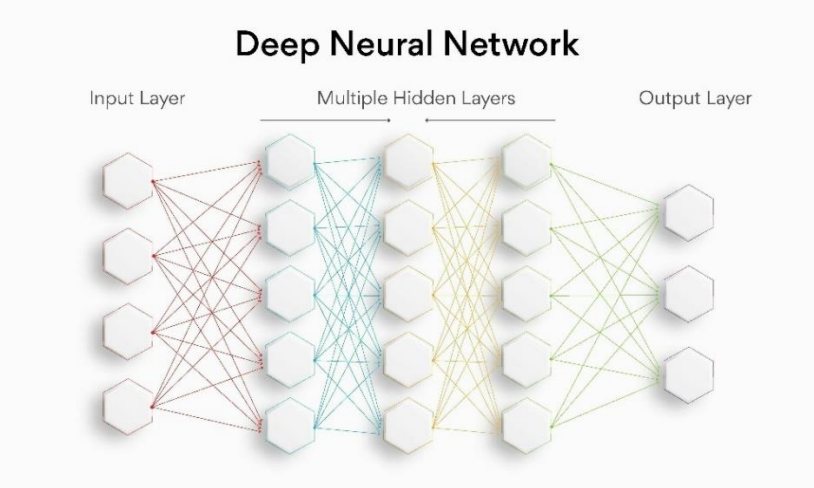


Figure 2: Illustration of an artificial neural network. (Role Of Artificial Neural Network In Artificial Intelligence, s. d.).

A layer refers to a distinct level or stage in the network through which data passes. As we can see in Figure 2, the data is then processed through a sequence of layers that work as a whole to transform the input data step by step, reducing the complexity of the data by considering more generalized features rather than specific details. Transforming raw data into a form where only the essential characteristics that are necessary for making decisions or predictions are retained. By extracting these features, it enables the model to identify patterns that are important for the task considered while reducing the noise of irrelevant details. These layers are sequentially linked until the final layer is ready. (Janiesch et al., 2021, p 685-688) (Sharifani et al., 2023, p 3898) In other words, the data is fed into the input layer. The input is then passed through neurons in the hidden layers. Each neuron takes the input, processes it using a mathematical function, and passes the result to the next layer until the information is completely processed to reach the output layer. The output layer that corresponds to the final prediction. For example, a neural network that processes the data for a classification purpose, will output the probability of the image belonging to a class or another.

Among the various types of neural networks, convolutional neural networks (CNNs) stand as the most popular one in the field of classification and object detection. A CNN is composed of a sequence of layers allowing the creation of the CNN. Among these layers, different types can be identified, each type having its own function. (Wu, 2017)

In the following paragraphs, we will explore these layers in detail, covering the convolutional layer, the pooling layer, the activation functions, and the fully connected layers. The figure 3 illustrates this sequence of layers from the input data to the output layer furnishing the computed

probabilities. We will discuss the role and function of each layer, how they contribute to feature extraction and data processing, and their importance in the architecture of a convolutional neural network (CNN).

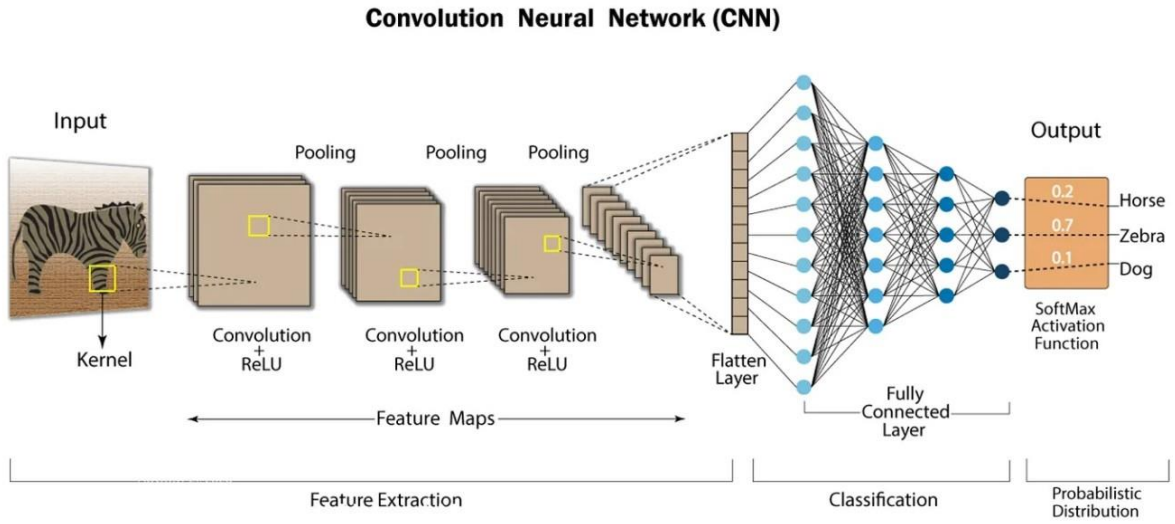


Figure 3: Convolutional Neural Network — CNN architecture (Haque, 2023).

The first and main one is the convolutional layer. This kind of layer uses filters (or kernels) to process input data (such as images) through a mathematical operation called convolution. In the case of image processing, the input data corresponds to a matrix where each node represents a pixel value. These kernels are small matrices that slide through the whole input matrix (where one node of the matrix represents a pixel value). The kernel slides depending on an attributed "stride", corresponding to the number of pixels the filter will move after each operation. A stride of 1 means the filter moves one pixel to the right each time and this process is repeated until the entire input image has been covered. At each position, the filter multiplies its matrix values with the corresponding portion of the input matrix to produce a feature map. A feature map is the output highlighting specific features such as edges or textures within the image. The whole convolution operation involves multiple convolutional layers that extract features by applying various filters that detect edges, textures, or other patterns in the input image. Each filter is designed to focus on a specific type of feature in the input, creating a feature map that represents the presence of these features. Through this whole sequence of convolutional layers, the model learns to visualize the general tendencies of the input image. In this process, lower layers identify simple edges, while deeper layers detect more complex features like shapes or objects (Albawi et al., 2018, p 4-5; CNRS - Formation FIDLE, 2022).

The filter size and the presence of a padding must also be determined depending on the objective of the model. The padding represents the process of adding rows of zeros (or another constant value) around the border of the input matrix before the convolution operation (Figure 4). By adding this border of zeros, it allows to control of the size of the output feature map. Without this process, the spatial dimensions of the output feature map will be reduced during the convolutions compared to the input dimensions. By adding padding, it is possible to keep the dimensions stable but also to ensure that information located on the edges is not lost. Thanks to these rows of zeros, the kernel will be able to retain the useful information anywhere in the image (Albawi et al., 2018, p 4-5; CNRS - Formation FIDLE, 2022).

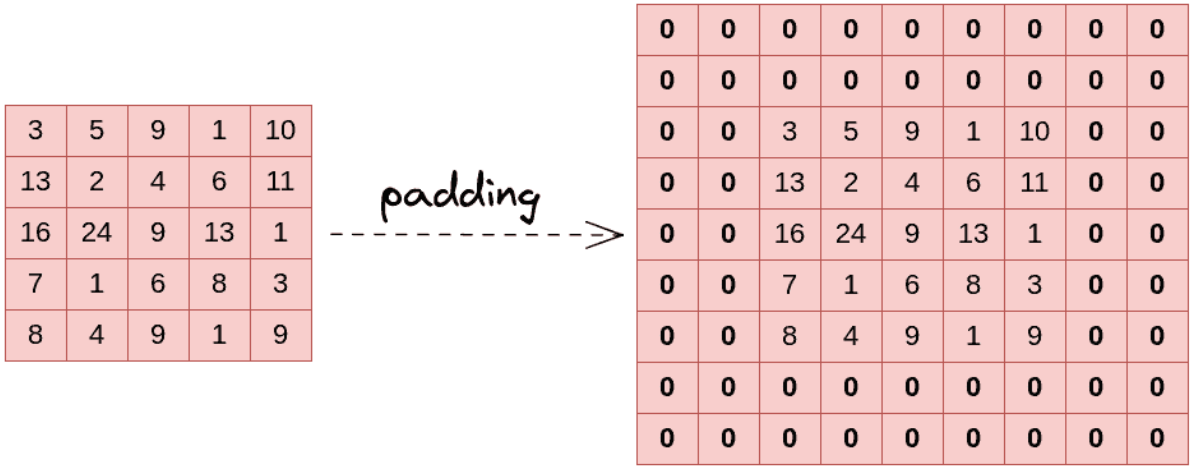


Figure 4: Padding of a Matrix in CNN (G. Nanos, 2024)

An activation function often follows the convolutional layer. The purpose of these functions is to enable the model to learn complex patterns by adding non-linear dynamics to the network. These functions are applied at each neuron in the network to determine whether it should be activated or step aside. Among these activation functions, ReLU is the most used in CNNs because of its computational efficiency and the solution it brings for the vanishing gradients issue (issue causing slow convergence or frozen layers). (Albawi et al., 2018, p 5) (Gurucharan, 2024) (Krishnamurthy, 2024) (CNRS - Formation FIDLE, 2022)

A pooling layer is then applied to reduce the feature maps' dimensions, making the model lighter and faster. Max pooling is commonly used in CNNs, as it retains only the highest value from each pooling window. Through this operation, a simplified feature map that retains the most prominent elements is obtained (corresponding to the highest value of the location). (Albawi et al., 2018, p 5) (Gurucharan, 2024) (CNRS - Formation FIDLE, 2022)

After the convolution, pooling and activation layers have processed the data, the network transitions to fully connected layers, also using activation functions. ReLU is often used in the initial layers while the Softmax activation function is commonly used in the final layer, especially for classification tasks. This Softmax activation function converts the outputs of the fully connected layers into probability scores corresponding to the likelihood that the detected object or input image belongs to each class. (Albawi et al., 2018, p 5) (CNRS - Formation FIDLE, 2022)

2.1.3 Computer Vision

As we have seen in the figure 1, computer vision is a field of artificial intelligence. It often involves the use of machine learning techniques combined with a willingness to reproduce human capabilities using data through machines. Computer vision primarily deals with enabling computers to interpret and understand the visual world, mostly through images and videos. This concept is currently in continual development and has become a fundamental solution for many sectors of activity. The main techniques include classification, object detection, and instance segmentation. All these processes are enabled through algorithms of machine learning / deep learning and image processing. These technologies have found applications in a wide range of fields, from industrial manufacturing to medical research, as diverse sectors increasingly adopt them (A. I. Khan & Al-Habsi, 2020, p 1445).

From a computer perspective, all data is stored and used as binary sequences. Since computer vision primarily handles visual inputs such as images and videos, this data is treated as numerical values of a two-dimensional array, where each element corresponds to an individual pixel. (Khan et al., 2018, p 3)

Integrating machine learning into computer vision enables programmers to empower their machines to interpret images, extract meaningful features and learn autonomously through explorations conducted on their training dataset. (*Key Differences Between Computer Vision and Machine Learning*, n.d.) Several ways to integrate machine learning techniques into computer vision models can be distinguished.

Now that we have defined our objective and the context where we are progressing, we can go forward. As our goal is to determine whether using model chaining performs better than a single object detection model that directly detects the desired targets. Our model chaining approach combines an object detection model to identify regions of interest in the image with a second classification model. This study is set in an industrial context characterized by a lack of variety

in the available data. The following sections will review the existing literature about object detection, image recognition (or classification) and model chaining.

2.1.4 Object Detection

With the objective and context established, we can now define what object detection means. In this section, we will also explore the existing models and review what has already been accomplished in this domain.

Object detection is a fundamental task in the field of computer vision; a branch of computer vision focused on enabling machines to interpret and understand visual information through images and videos. The specific task of object detection involves identifying and locating objects within the input images. Unlike image recognition which classifies an entire image into a class, object detection identifies multiple objects within the image and classifies them into classes. In this field, deep learning models are trained on large datasets of annotated images containing the studied objects to enable them to detect the determined classes on a new dataset. (Wu et al., 2020) (Zhao et al., 2019)

2.1.4.1 Algorithms and frameworks presentation

When it comes to object detection, several powerful libraries are already established and widely used. In these libraries, Faster R-CNN, EfficientDet, YOLO and Detectron2 are among the most popular.

Faster R-CNN

The R-CNN series, starting with the invention of R-CNN, was a significant development in the field of object detection. Introduced by Ross Girshick in 2014, R-CNN employed convolutional neural networks (CNNs) to process region proposals from an image. In this approach, region proposals were generated using an external algorithm like Selective Search. Each region proposal was then passed through a CNN independently to extract features, which were subsequently classified using a Support Vector Machine (SVM). This method laid the groundwork for subsequent advancements in object detection by demonstrating the effectiveness of CNNs in extracting features for accurate object classification. (Girshick, 2014)

The algorithm then evolved into Fast R-CNN before reaching its final version called Faster R-CNN, which enhanced the detection process by introducing a Region Proposal Network (RPN). The model starts by processing the input image through convolutional neural network layers to generate a feature map. The RPN then uses this feature map to identify regions of interest,

which are subsequently refined and classified, optimizing the detection process for better performance and accuracy (Figure 5). (Girshick, 2015)

Two-stage object detection models are usually known to be more flexible and accurate than the one-stage models, with Faster R-CNN being a prime example as it first proposes candidate object regions and then refines these proposals in a second stage for more precise detection. While one-stage models have the advantage of being simpler and more efficient by leveraging predefined bounding boxes (representing the location of the objects to detect). (Tan et al., 2019, p 10782)

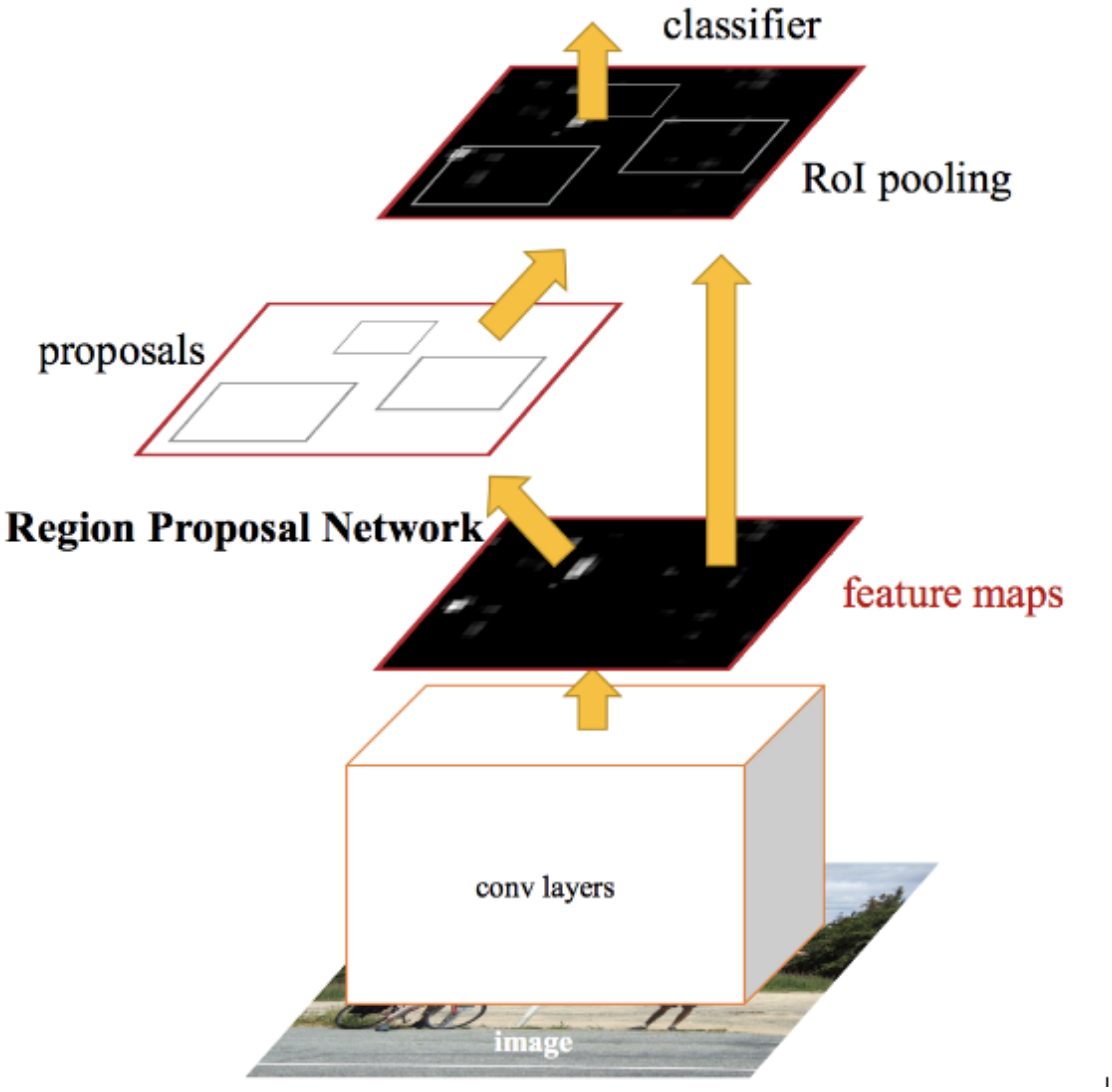


Figure 5: Illustration of Region proposal generation similar to Faster R-CNN. (Pusarla, 2021)

The input image is first processed through a series of convolutional layers. Each of these convolutional layers is equipped with filters or kernels that aim to extract specific features such as edges, textures, or patterns. The inclusion of max pooling layers further enhances the model's capability by capturing the most salient information from each local region. Once the entire

image is processed by the CNN, these feature maps are then passed to the Region Proposal Network (RPN), which generates region proposals indicating potential object locations. These regions represent areas where objects might be present and allow narrowing down the search space for potential objects instead of considering every possible location in the image. (Girshick, 2015, p 1441) (Winter et al., 2021, p 3) This is why this method using Regions with CNN features is called "R-CNN". (Yagüe et al., 2022, p 6)

For each identified region of interest, a fixed-length feature vector is extracted. This process is crucial for facilitating object detection in regions of varying sizes and aspect ratios (Width/Height). Each feature vector extracted from the Region of interest pooling layer, is then processed through a sequence of fully connected layers. These fully connected layers allow the model to learn intricate patterns associated with the features of the proposed region. Finally, the output layers are designed as two sibling branches serving distinct purposes. The former is the SoftMax Output layer responsible for producing probability estimations. These estimations are calculated over the K studied object classes. (Girshick, 2015, p 1441) The class corresponds to the different types of objects the model is trying to detect. The background itself is also often considered as a distinct class and can be considered to add regions that does not correspond to any of the specified classes. (Bansal et al., 2018, p 6) The latter is the Bounding Box Regression output layer providing four values representing the coordinates of the bounding box associated with an object of a particular class. This regression-based model enables learning and fine-tune the exact spatial location and dimensions of the detected objects. (Girshick, 2015, p 1441) (Li, 2021, p 4)

The current version of Faster R-CNN significantly improves upon its predecessor in terms of speed thanks to the integration of a Region Proposal Network (RPN). Instead of relying on the conventional method of sliding a window through the image, it employs RPN to enhance the speed of detection frame generation. This innovative approach eliminates the need for a separate region proposal generation step, streamlining the entire process and making it more efficient. (A detailed explanation of the technical aspects of the functionality is provided in the appendix.) (Li, 2021, p 3)

YOLO

In contrast to Faster R-CNN, YOLO is a one-stage algorithm for object detection. This means that YOLOv8 processes input images, predicts bounding boxes, and performs classification in a single pass through the network. To process the input images, the algorithm employs mosaic

data augmentation, combining multiple images into a single one, providing a diverse set of training sets and improving the model's robustness. A pivotal feature of YOLOv8 is the adaptive prediction of bounding box calculation, dynamically adjusting the predefined boxes that are the initial estimations of the bounding box dimensions computed based on the dataset characteristics. This approach consists of estimating the position and dimensions of an initial bounding box before dynamically adjusting it during the training of the model.

The backbone network processes input images using Conv and C2f modules to extract important features at multiple scales, allowing the algorithm to detect objects of different sizes. The Conv module is similar to the convolutional layers in Faster R-CNN, while the C2f module helps the model capture complex patterns, improving accuracy. (Wang et al., 2023, p 2-3) (Tamang et al., 2023, p 894)

A neck layer then enhances the model's ability to merge features by using the Feature Pyramid Network (FPN) and the Path Aggregation Network (PAN). FPN and PAN combine high-level and low-level features through techniques like up sampling (increasing resolution) and down sampling (decreasing resolution). This helps transfer both context and localization information effectively. (Wang et al., 2023, p 2-3) (Tamang et al., 2023, p 894)

Using these techniques, the model captures detailed information with high-resolution features and abstract information with low-resolution features. Together, FPN and PAN improve the efficiency of detecting objects of various sizes. While FPN focuses on capturing multi-scale features, PAN aggregates context information, and their combination significantly boosts the model's performance. (Youssef, 2002, p 1) (Wang et al., 2023, p 2-7)

In practice, YOLO divides the image into a grid before launching the process of object detection to find the coordinates of the bounding boxes in each cell of the grid. (Xiao et al., 2023, p 4)

The latest version of YOLOv9 was released in February 2024. YOLOv9 introduces two major advancements over YOLOv8: Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN). PGI is a novel system that ensures complete input information is maintained and provides reliable gradient updates, leading to more accurate training of the model. GELAN is a new lightweight network architecture designed to use conventional convolution operations more efficiently, resulting in better parameter utilization and improved performance. These enhancements allow YOLOv9 to achieve superior accuracy and efficiency, even when trained from scratch,

compared to YOLOv8, which primarily utilizes depth-wise convolution and relies on pre-trained models. (Wang et al., 2024)

Detectron2

Detectron2 is an open-source computer vision library developed by Facebook (Merz et al., 2023, p 2). This powerful framework is designed to facilitate the development and deployment of computer vision models, particularly for tasks such as object detection. With an extensive collection of pre-trained models, it guarantees flexibility, extensibility, and user-friendly implementation. This library is constructed on top of PyTorch (a deep learning framework) to ensure efficient GPU utilization. One of its key tools is the Model Zoo, which provides various models specifically for object detection. Detectron2's architecture shares similarities with Faster R-CNN, as both follow a two stages processing approach. Like Faster R-CNN, Detectron2 initially identifies regions of interest before looking for setting the bounding boxes within these regions. (Merz et al., 2023, p 1-4) Although, one-stage model in Detectron2 also exists and are used in some practical cases. (Jabir et al., 2021, p 181) (Evangelista et al., 2022, p 930)

EfficientDet

EfficientDet is a family of Object detection models aiming to find the right balance between network depth, width, and resolution to enhance better performance. (Sharma, 2022) Designed to identify and locate multiple objects within an image, it has been developed by the Google Research Brain Team. EfficientDet employs a single-stage model that integrates a scalable architecture, where network depth, width and resolution are systematically optimized through a compound coefficient (formula used to systematically scale the network). (Tan et al., 2019,p 10782,10788)

More information on how EfficientDet works can be found in the Appendix 2.

2.1.4.2 Evaluation and comparison

This section goes through different research papers and case studies where 2 or more algorithms presented in the previous section are used and evaluated.

A first research paper led by Jabir and collaborators (2021) compares all the above algorithms on a single case consisting of detecting weeds in crops. The result of this study shows that YOLOv5 has the best overall performance compared to the other algorithms on the studied dataset. YOLOv5 is one of the previous versions of YOLOv8 which has a similar backbone

(Terven et al., 2023, p 22). Even though Detectron2 has better precision on this dataset (97% precision), it tends to make mistakes by detecting what shouldn't be detected as the researched object (42% of error on the dataset compared to 30% for YOLOv5). About the training time, YOLOv5 and Detectron2 are the two fastest (2h and 5h respectively). While Faster R-CNN and EfficientDet seem to offer poor performance and a running time of 10 to 12h on the studied dataset. (Jabir et al., 2021, p 181)

Another article studying YOLOv5 and Faster R-CNN implemented on Detectron2, shows similar results on a dataset of birds. Detectron2 detects all the birds (with the only error of combining two birds) while YOLOv5 missed the half-captured birds. (Evangelista et al., 2022, p 933)

As a relatively newly released model (2023), YOLOv8 still needs to be tested in lots of fields but some tests comparing it to the other models have already been made. A first evaluation made by M. Reza Azhar Priyadi and Suharjito demonstrates that in terms of precision, YOLOv8s outperforms EfficientDet Lite 4 for a dataset composed of oil palm fruit from different maturity poses. YOLOv8 is then better in terms of object detection however, EfficientDet 4 Lite seems to be better in terms of object classification and bounding boxes setting. (Priyadi & Suharjito, 2023, p 1,6)

An evaluation of YOLOv8 and Detectron2 has been done by 3 students of Kongu Engineering College in India on a dataset for surface defect detection on steel strips. After training and testing their models, they found that even though Detectron2 has a longer computational time, it generates more accuracy than YOLOv8. (with over 95% accuracy) (Akshayanivashini et al., 2023, p 5-6)

In conclusion, YOLO appears to have the best overall performance, particularly for real time applications, due to its rapid computation speed. Besides that, Detectron2 models seems to provide the best results in term of detecting the object we want it to detect but has the drawback of detecting more unintended objects compared to YOLO.

2.1.5 Image recognition

The other common computer vision task is the classification or image recognition. For this kind of tasks, computers categorize images based on their overall content. As for Object detection, classification models often employ techniques using CNNs. The procedure is then similar with

the main difference that even though a label is needed for the training image dataset, no bounding box must be defined as the whole image needs to be classified. (Zeng, 2023, p 430)

In the image classification field, many different algorithms already exist and are optimized to perform with high accuracy on various datasets.

A first research paper comparing the performances of VGG19, ResNet50 and Inception-V3 on classifying food images demonstrates that the three models are viable with a slightly better accuracy for Resnet50 and Inception-V3 with 97.29 and 97.57 respectively. While VGG19 is at 96.86% precision. (Andrew & Santoso, 2022, p 1)

A second research led by S.Poudel and P.Poudyal in a context of waste materials classification, points to the same conclusion with Inception V3 showing the best performance. (Poudel & Poudyal, 2022, p 1) While Resnet-50 leads in terms of precision in medical dataset for diseases classification or for a face recognition task. (Pan et al., 2023, p 1) (Nyarko et al., 2022)

On the other hand, some computer scientists decided to introduce YOLOv8 for the classification task. As mentioned earlier, YOLO is primarily known for object detection. But among its versions, YOLOv8-cls stands out. This version is specifically tailored for classification tasks and has demonstrated performance comparable to other leading models. Research conducted by the university of Tikrit on Ophthalmic disease classification, demonstrates a superior performance of YOLOv8 (94% precision) compared to Resnet-50 (92%) and VGG-16 (88%). (Khalaf & Abdulateef, 2024) Another study supports this conclusion and introduces another version of YOLOv8 applied to a dataset of lettuce nutrient deficiency classification. (Sikati & Christian, 2023)

Most of the popular existing models are usable in every kind of datasets. The main determinant of their performance is the fine tuning of the training set. In this logic, Resnet-50 and mainly YOLOv8n-cls (both using CNNs) will be the models used and studied in this research paper.

2.1.6 Evaluation metrics

Several evaluation metrics will be introduced in this research paper. The precision measuring the accuracy of the model's positive predictions. It is defined as the ratio of true positive detections to the total number of positive predictions. In object detection, precision indicates how many of the detected objects are correctly identified.

The mean average precision at IoU (Intersection over Union) = 0.5 (mAP 50) measures the overlap between the predicted bounding box and the actual ground truth bounding box. An IoU threshold of 0.5 means that the predicted bounding box must cover with at least 50% of the ground truth box to be considered a correct detection.

The mean average precision across IoU thresholds (mAP 0.5:0.95) provides a comprehensive evaluation by averaging the mAP over multiple IoU thresholds. This average ranges from 0.5 to 0.95 in increments of 0.05.

2.2 Transfer learning and Model chaining.

In this section, we will address two crucial concepts: transfer learning and model chaining. Transfer learning is an essential step in building any computer vision system using machine learning, as it allows the use of pre-trained models to enhance performance. Model chaining, the method we will implement and test in this dissertation to compensate the lack of variety in the available data.

2.2.1 Transfer learning

Transfer learning is another concept widely used in the machine learning field and notably in computer vision. This usage consists of reusing a trained model for a specific task, as a starting point for the training of a model on a second task. This strategy has proven its efficiency, especially in a context of lack of amount and quality of data. The employment of a pre-trained model leverages existing neural networks that have been trained on previous datasets (from regular dataset of a case study to a large one like those used in ImageNet, COCO or GoogleNet challenges). Through this process, data itself is not shared; images used to train the pre-trained model are not reused but the learned features of the pre-trained model are. This approach allows the new model to bypass the initial stages of learning basic patterns and directly focus on the more complex features. This not only saves computational time and resources, but also improves the robustness of the model in scenarios where data acquisition represents a challenge. The new model can focus on new features or on refining the existing ones to specific tasks and datasets. (Zhou et al., 2020, p 1) Model Zoo is one of these collections of pre-trained models wildly used for object detection. Notably by Tensorflow and Pytorch (popular frameworks and library for computer vision). (Chen & Shah, 2021, p 2) (Hsiang et al., 2020, p 1)

2.2.2 Model chaining

Model chaining in machine learning involves organizing the workflow into independent, reusable components. In our case, we have two main components corresponding to a model of object detection and a model of classification. There are various ways to combine these components to build different ML systems, enabling each part of the workflow to be used in multiple contexts. Sequentially linked, these ML parts form a linear chain where the output of one model becomes the input of the next one (Figure 6). This method helps accelerate development and deployment of machine learning models but also allows building complex models through different modules that can be adjusted without affecting the entire system. (if the outputs stay in the right format) (Bald, 2023)

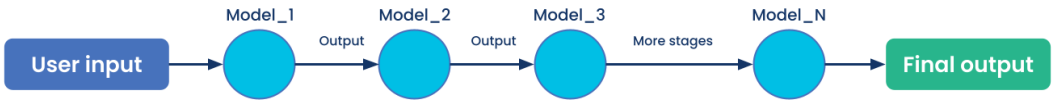


Figure 6: Illustration of Model Chaining. (Multi-model Composition With Ray Serve Deployment Graphs | Anyscale, n.d.)

A first research paper led by Weizmann, Institute of Science, introduces the model chaining to address the challenge of detecting object parts that are not easily recognizable by their overall appearance (such as hands or parts of an animal from low-resolution images). Instead of attempting to detect the target object directly within a single model, a model chain is built to detect an entity (a human or animal), before using the detected part as input for another object detection model to spot the specific target (hand or part of an animal). (Karlinsky et al., 2010)

A last paper outlines a use of model chaining into a modelling pipeline (fully automated system from data collection to final predictions) for pipeline defects detection. To enhance efficiency and accuracy, the research proposes a two-part model chaining strategy. The first part aims at the detection of regions of interest thanks to an object detection model. This model is designed to identify regions that are most likely to contain anomalies. Following this stage, a second model uses the regions of interest as input to apply another object detection task to analyse the presence of defects. This computer vision system helps reduce the dataset preparation time while increasing the reliability and accuracy of the system. (Moradi et al., 2018)

3. Project overview and methodology

3.1 Introduction

In this chapter, we will explore the origins of this thesis linked to a Belgian company, and then delve into the development of an initial approach. This first approach, which focuses on material detection, will serve as the reference model for benchmarking in this dissertation. We will then analyse the results of the initial model and identify its limitations, primarily due to a lack of diversity in the training data. Subsequently, we will propose a solution based on the concept of Model chaining. As discussed in the literature review, the application of model chaining systems can enhance a system's accuracy and compensate a lack of data available (instead of using a single model of object detection). In our case, while thousands of images are available, they suffer from a lack of diversity, making the model highly specialized in a certain context but less effective in varying contexts.

This thesis aims to determine whether model chaining can address this issue in a scenario where the standard method of a single object detection model struggles. In an industrial context characterized by a lack of diversity in the available data, how can model chaining enhance the flexibility of a computer vision system for the detection and classification of transported materials?

3.2 Problem definition

This thesis and its primary research question originate from a project executed by a Belgian company called Technord. This project was initiated by the department of Data Sciences to discover, explore, and establish a new branch oriented towards computer vision. Future computer vision activities will soon be launched, and the company's objective is to build a solid foundation with a comprehensive overview of the best existing models to develop some functional and general-purpose models.

The company Technord offered to its clients the possibility to add computer vision to their panel of industrial solutions. After consulting potential clients, several of them expressed interest in computer vision solutions related to classifying transported materials. While these materials and the environments where images are captured can vary from client to client, some common characteristics exist. The materials are consistently transported by trucks, and the images are

taken by a camera mounted above the truck on a gantry. The materials can generally be categorized into 4 general classes: wood, soil, stone, and sand.

In summary, the objective is to create a computer vision system able to detect and classify the materials transported in open-top truck beds as they pass under a gantry equipped with cameras.

3.3 Material detection model

This section aims to introduce the initial approach briefly before discussing the outputs of this first model, guiding the research presented in this article. Throughout this research paper, we will refer to this model as either “the initial approach” or “the material detection model”. The functioning and the justification for the selected model will be explained in the following chapter discussing the object detection model.

The first approach was to apply an object detection model, trained on a dataset of labelled images (with the bounding box of the object we want to detect) of trucks transporting materials. Since manually labelling hundreds or thousands of images represents a heavy and time-consuming process, the first step in data collection was to look for existing annotated dataset. This search led to a dataset of around 1300 images divided into a training set of 1100 images and a validation set of 200 images, found on Roboflow, a platform offering a large panel of free labelled datasets provided by other users. This dataset consists of a series of similar images taken by a 360-degree camera positioned directly above truck beds (Figure 7). Five classes are represented: brick, concrete, empty body, soil, and wood.

Figure 7: Example of training image.

After training the YOLOv9 model for 100 epochs (iterations of training) on the dataset found on Roboflow, a trained model was obtained. An overall precision of 74.26% was reached with

a mAP 0.5 of 75.88% and an mAP 0.5:0.95 of 50.67% on the validation set. mAP_50 corresponds to the mean Average precision of the predicted bounding box covering at least 50% of the bounding box corresponding to the reality. While the metric mAP 0.5:0.95 indicates a more comprehensive assessment, averaging the mAP across a range of intersection from 0.5 to 0.95. As our objective is to get the type of material transported without any need for a bounding box precisely delimiting the truck bed, the overall precision will be our main evaluation metric.

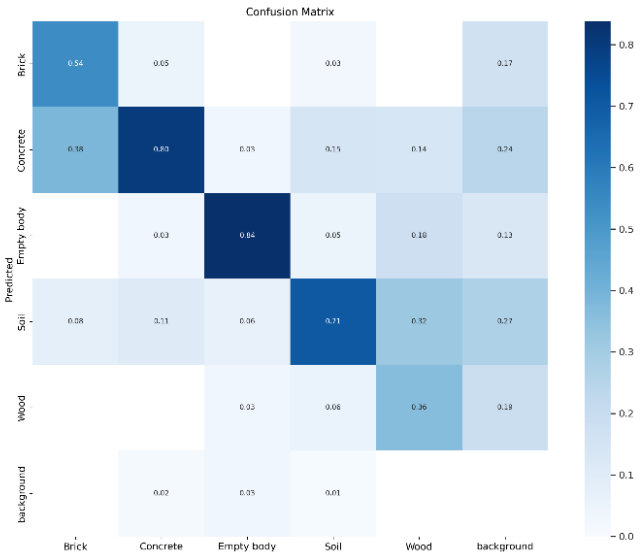


Table 1: Confusion matrix of materials detection

Table 1 shows the confusion matrix of the object detection model, providing insights into how well the model distinguishes between the different types of materials. We can see that the model successfully detects bricks about 54% of the time, 80% for the concrete, 84% for the empty body, 71% for the soil and 36% for the wood.

These results demonstrate that while the model can detect transported materials, improvements can be made as it sometimes misses materials and confuses classes. For example, while wood is correctly detected 36% of the time, it is misclassified into the soil class at a frequency of 32%. How can this be improved? The most effective way to enhance an object detection model is by fine-tuning its training dataset. This involves adding diverse images of the concerned classes into the training set. In our case, images of our different materials transported in a truck bed need to be found. To boost performance, this new dataset should include images taken from above the truck and in various angles, quality levels and lighting conditions, using different camera types. The dataset used for this initial approach did not fulfil these conditions due to the lack of diversity in the training images. All images were taken by the same type of camera, with

consistent resolution and quality, and positioned directly above the truck bed. Despite varying lighting conditions, the fixed camera angle resulted in a limited variety of perspectives in the training set.

This phase of dataset collection represented a challenge due to the limited number of annotated images that met our criteria, as much as the images without annotation. Only 3 datasets of annotated images of materials transported by trucks were found.

The first one corresponds to the one used in this first model of object detection. 1300 images from 360-degree camera with good quality and various luminosity (picture taken in the daylight and during the night) but which needs to be fine-tuned with other images taken from other types of cameras and angles.

A second dataset of 2000 annotated images was also found on Roboflow but 3 main elements were arguing its incorporation into our model. First, the images were taken from the same type of camera as the used dataset (360-degree camera), many images were duplicates of those in the previous dataset and some of the images were miss-annotated. Mistakes in the annotation process can lead to a diminution of the model's performance, as it would learn from wrong objects. The frequent mistakes in annotation operation are labelling an object to a wrong class or miss-defining the bounding box of the detected object.

A third dataset of 11500 annotated images was found but it was again pictures taken by the same type of camera, same angle, and the quality of most of the pictures were too poor to be considered due to the camera's dirtiness.

In the absence of suitable annotated datasets, an alternative could be to find or create a non-annotated dataset. However, no free and accessible images meeting the required criteria were available.

We are then in an industrial context where a single model of object detection can not perform well enough due to a lack of diversity in the training images available. The solution that this academic paper is going to investigate, and build is the conception of a computer vision system based on the principle of model chaining. It has been identified in the literature review that model chaining can enhance the performance of a model by improving its precision or by compensating for the lack of quality.

The question that remains is: Can it improve the performance of our system? **In an industrial context characterized by a lack of diversity in the available data, how can model chaining**

enhance flexibility of a computer vison system for the detection and classification of transported materials?

In the following section, an alternative using the model chaining concept through a model of object detection and a model of classification will be proposed. The solution will be divided into two main models structured into 6 main phases following the machine learning value chain, corresponding to the methodology applied in the project (*The Value Chain of Machine Learning*, n.d.)

3.3 Model Chaining system

In this section, we will introduce and develop the methodology used to create a model chaining system. To enhance the system's performance and compensate the lack of diversity in our dataset, the model chaining implemented here will split our initial object detection model that detected the materials, into two sequential models. The first one is an object detection model (1) that focuses on identifying the truck bed, regardless of its contents. The coordinates of the detected bounding boxes then serve as the input for a second model of classification (2) that will categorize the detected truck bed into one of the specified material types. As the problem definition has already been outlined in a previous section, we will directly start with the second phase of the machine learning value chain: the data collection.

.

3.3.1 Data collection.

1 The second step of the machine learning value chain is the Data collection. As it is the only viable and well-annotated dataset of images of transported materials found, the dataset used for the truck bed detection model, is the same as in the initial approach (Figure 7). This dataset was sourced from Roboflow, a platform providing a wide range of annotated datasets for download. The downloaded dataset is divided into 2 sets (train and validation) with annotations available in various formats depending on the model used.

For the second classification model in our system, a separate data collection effort is necessary. This is because the second model has a distinct objective, requiring its own specific training dataset tailored to its classification goals.

[2] To build an image recognition model, hundreds or thousands of representative images for each class are necessary. As defined in the introduction, we are looking to classify 4 types of materials: soil, stone, wood, and sand. To build the first version of the dataset, files containing between 700 and 900 images have been downloaded from different datasets available on Roboflow. This new dataset then contains images of stone (mainly gravel), soil (close-up images of soil from different types and colours), sand (close-up images of sand from different colours) and wood (mainly images of planted trees and their branches). On all these sets, data augmentation techniques were applied by the users who submitted them to Roboflow. Techniques such as rotations, translations, and the addition of white noise (white dots dispersed on the image).

3.3.2 Data storage.

Once the data is collected, it needs to be stored in a secure and accessible location. Training deep learning models requires significant computational resources in terms of time and processing power, so a GPU (Graphics Processing Unit) is essential. A GPU is a graphic processing technology designed to accelerate and handle complex mathematical calculations. Google Colab is one of the cloud services offering GPU access and cloud storage so, the datasets are stored on Google drive for convenient access as the model is trained on Google Colab.

3.3.3 Focus on the truck bed detection model.

3.3.3.1 Data preparation of the truck bed detection model

The next step of the ML value chain is data preparation, which ensures that the data is properly formatted as input for the model of object detection. Since Roboflow already provides data in a structure suitable for our model (with the annotations), only minimal adjustments were needed for the truck bed detection. An example of Data preparation for a model built from scratch can be found in the Appendix 3. The model in the Appendix corresponds to a test of Detecton2 on a Dataset manually built and annotated identifying the presence of bottles in images.

Now that we have collected our data, it is time to build the model. However, before starting the training process, we need to select the appropriate model and prepare the data accordingly to ensure it fits well. Nowadays, it is more popular to use an optimized existing model like YOLO, Detectron2 or Faster R-CNN than building a model from scratch using CNNs. From the literature review, we have seen different comparisons allowing us to say that YOLO and

Detectron2 represent two viable and promising models. Detectron2 has the advantage of being more customizable, making it popular for research in the domain of CNNs while YOLOv9 is more recent (February 2024), and popular among the computer vision programmers' community. After discussions within the data science department and their personal network, YOLOv9 was selected as the reference model for object detection.

The current chosen model for truck bed detection is then YOLOv9. The structure of the data required by YOLOv9 consists of having our images divided into two folders (the former containing the training set allowing the model to learn and the latter containing the validation set to provide an unbiased evaluation of a model after each iteration). Each of these folders contains 2 other folders. One for the images and another for the annotations. These annotations represent a series of txt files containing the associated class number to detect, and the coordinates of the four corners of the bounding box that frames the object. The images and their labels are linked by their names as both are the same (name.jpg and name.txt). Finally, a yaml file is used to give the different data paths to the model, the number of classes to detect and their names.

Since the downloaded dataset originally aimed to detect materials inside the truck bed and our focus is only to detect the truck bed itself to output its coordinates in the image, some modifications were made in the yaml file and all the txt files to group all the existing classes into a single one: "truck bed" or "benne" in French.

3.3.3.2 Algorithm programming of the truck bed detection model.

Now that the model is chosen and the data is prepared to fit with YOLOv9, the training phase can begin. As seen in the literature review, YOLOv9 uses the CNNs to efficiently process and analyse the images, extracting features across the convolutional layers and detecting objects. The source code of YOLOv9 can be found and cloned on GitHub (https://github.com/WongKinYiu/yolov9). A few parameters still need to be set.

The first one is the pre-trained model used to leverage learned features from vast datasets (cfr Transfer learning section of the literature review) . The extended pre-trained model of YOLOv9 is picked for its polyvalence of context trained. A final data preparation operation is imposed and included in the YOLOv9 training command. The resolution of the images for the training was set to 640x640 pixels. It represents the resolution of treatment by default of YOLOv9. The images are then reshaped to the chosen resolution to normalize them. It allows the model to compare and make the kernel slide uniformly through the images. A higher resolution can

enable the model to extract more precise features but increase considerably the computational costs. While a lower resolution has the advantage of reducing the computational requirements but can negatively impact the model's performance, as the extraction of detailed features might be enabled. Finally, the number of epochs (iteration of training on the train and validation sets) is set to 100 epochs.

Fine-tuning of the model

After 100 epochs of training, the model achieved a precision of 99.5% and a mean average precision (mAP) of 99.4% on the validation set provided by Roboflow containing image similar to training set (Figure 7). Appendix 3 includes a sample of predictions with confidence scores, showing how certain the model is in its predictions. Examining the metric evolution across epochs reveals that precision stabilizes after 82 epochs at 99.2%.

The question that we can ask ourselves is: Should we keep the weights of the training after 82 or 100 epochs? More epochs provide more training, improving the model's accuracy for images like the training set. However, too many epochs can lead to overfitting, meaning that our model will become overly specialized in data similar to the training set's context but might become less flexible by seeing its performance decrease on images from another type of camera/ angle.

To assess this, a test set was manually built to evaluate our truck bed detection model in various contexts. This test set contains 22 images. Twelve of them or taken by 360-degree camera (from another dataset than the one used to train our first model), while the last ten images are from a dataset provided by researchers from the University of Hong Kong who conducted a study on estimating construction waste truck payload volume using monocular vision. (Chen et al., 2022) Figure 8 represents a typical example of each part of the test set.

Figure 8: Sample of the Test set for evaluation.

The test set was used on both object detection models (having the number of epochs as only difference). Both successfully detected most truck beds, with errors committed by both models.

These errors will be addressed in the final chapter when evaluating the entire computer vision system, as this test aims to select the most efficient model.

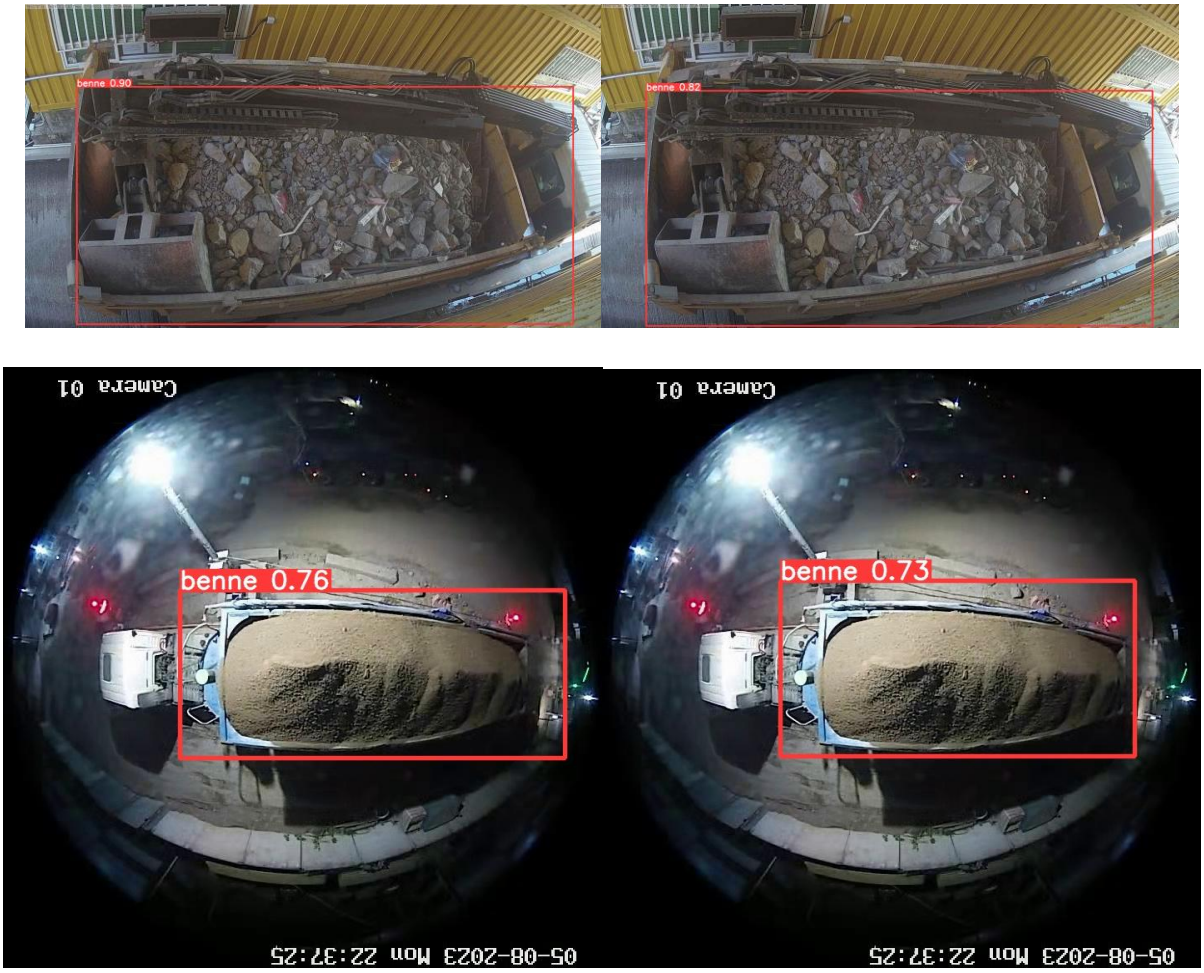


Figure 9: Comparison of object detection model: 82 epochs vs 100 epochs

As we can see in Figure 9, both object detection models were applied to the images and the truck beds were identified in both cases. The confidence rates reveal that the model trained for 82 epochs is more confident in its predictions than the one trained for 100 epochs. This finding suggests that the latter is likely overfitting, making it more accurate on images it learned on but less efficient on unseen images. The model trained for 82 epochs is then the one that will be implemented into our model chaining system.

This same test dataset will be used at the end of this research paper to compare the initial approach of materials detection to the new model chaining system built.

3.3.3.3 Application development of the truck bed detection model

The final phase of the machine learning value chain is about what comes next. What can we do with this created model? The current model outputs images with truck beds framed (Appendix 2). The objective of this first model in the model-chaining approach is to isolate the truck bed portion of the original image to classify this region of interest in the following model. To accomplish this, the coordinates of the detected truck bed need to be saved for future use. As the detection command relies on the YOLO library's source code, modifications were made to the detection source code to store the computed coordinates of detected truck beds in a file. This file will serve as input for the next model in the chain, which will classify the truck bed's contents.

3.3.4 Focus on the classification model

Now that we have the coordinates of the truck bed thanks to our first model, the next step is to extract the region of interest to classify it using another deep learning model. This chapter will follow the same structure as ML value chain to go through all the steps of the creation of a classification model with the difference that unlike the previous model, the dataset will have to be fine-tuned to reach better performance. This process of dataset fine-tuning will be developed in the "Algorithm programming" section. This model will also be further developed, especially regarding how the training process goes as this model is further fine-tuned than the previous one.

3.3.4.1 Data preparation of the classification model

On the image recognition side, three main elements need to be prepared to allow the training of the model. The first one is the structure of our dataset, so it fits into our model. The data need to be structured into 2 folders (a train set, and a validation set), with each of them containing 4 folders named by their corresponding classes. Finally, each class folder contains its respective images of the train or validation set.

The second element is to create a yaml file giving the path of the different folders where the images are, the number of existing classes and their names.

The third one corresponds to the conversion of the outputs of the last model into the inputs of the current model. Previously, we have computed and stored the coordinates of each truck bed detected in txt files named with the same name as the original image. These txt files stored in a

designated folder are opened to take back the coordinates before converting them into the original resolution of the image. As a reminder, to be detected, the images are reshaped into the same resolution as the one used for the training (640x640 pixels in our case). The coordinates are then computed with the reshaped resolution. As we want to take the region of interests in its original resolution, the coordinates need to be converted back into the original resolution to extract the part of the image that interest us without any modifications. Finally, the truck bed is extracted from the image and stored in a variable that will serve as input for the classification model.

A second part of data preparation has then been done by fine-tuning the dataset and some training parameters through a trail-and-error process of training our classification model in YOLOv8n-cls. The adjustments of the dataset and the parameters were made based on confusion matrices produced after each fine-tuning operation and the constitution of independent test sets for each class to spot the kind of images that could be integrated into the training set to enhance its performance.

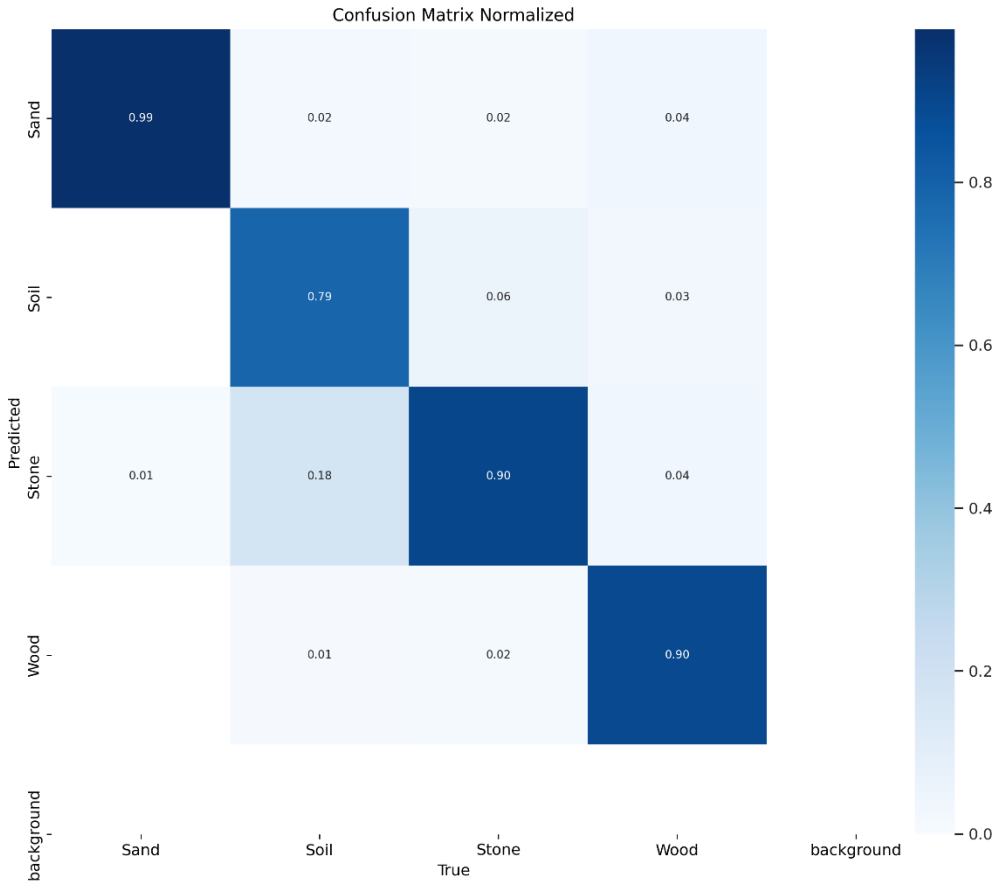


Table 2: Normalized Confusion Matrix of YOLOv8n-cls with 64 pixels as training resolution.

Table 2 corresponds to the initial confusion matrix with the images reshaped to a resolution of 64x64 pixels. A comparison of the performance reflected by the confusion matrix was then conducted for a resolution for the training of 128, 256 and 512 pixels. From the results of this comparison, it has been demonstrated that the more we increase the resolution, the better the model performs as it can extract more detailed features, allowing it to reduce the confusion between the classes. However, a resolution of 512 pixels seems to represent a breaking point as the model's performance drastically dropped. This might be since the resolution exceeds the original resolution of some inputs. The resolution is then fixed at 256 pixels for better predictions.

After deeper analysis of the training sets, an issue of variety for some classes was identified. The stone dataset was mainly represented by gravel, and the wood dataset only contained images of planted trees, which does not fit real applications as the transported wood would more likely be constituted of tree branches or trunks. Around 400 images were manually removed from the stone dataset and 600 images from the wood dataset. The images were manually removed in order to delete most of the images created through data augmentation techniques. This way, a larger variety of initial images remained instead of keeping a shorter variety of images with their modified replications. The datasets were then completed by extracting around 250 images of rocks from diverse available datasets on Roboflow and about 150 images of rocks and concrete were manually selected from Google Images. For the completion of the wood dataset, 500 images were extracted from datasets of branches or wood log detection on Roboflow and another hundred images were also picked from Google Images.

To pinpoint other issues, a small test set was assembled with images from Google Images. This test set consists of images of all the classes under various lighting conditions and reliefs. After testing them on the current model, it was noticed that there were a common point between most of the misclassified images. Most of them were images with reliefs such as piles of wood, soil, stone or sand.

Since no dataset of piles of materials was found, a hundred images of each class were downloaded from Google Images and added to our training set. This final operation of dataset fine-tuning allowed us to get an optimized model.

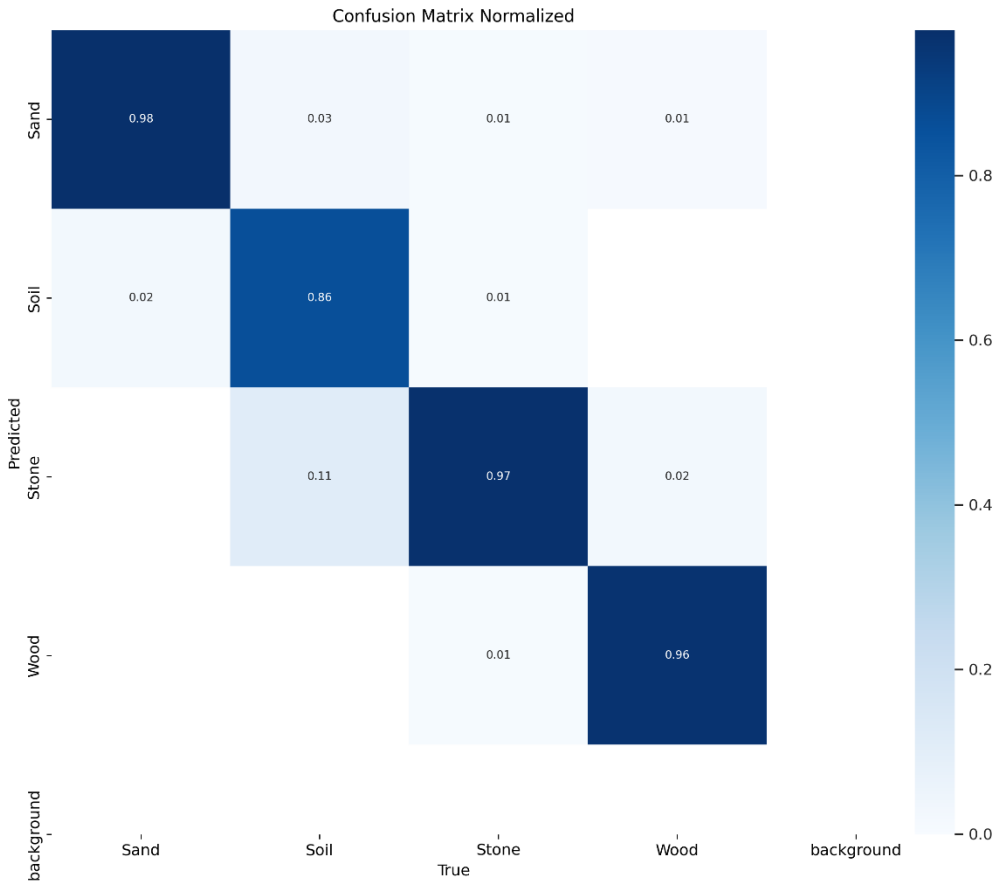


Table 3: Normalized Confusion Matrix of the final model of classification.

As we can see in Table 3 representing the normalized confusion matrix of our final classification model, sand is classified with 98% of precison,97% for the stone, 96% for the wood and 86% for the soil. The precision of the soil classification is lower than the others, but it finds its explanation in the fact that images of soil composed of a lot of stones or gravel, tend to be classified as stone. This can be solved by removing all images of soil with stone from the soil dataset if we want to consider a soil only when it is clean. However, in our industrial context, it does not represent any issue. It could even give an opportunity to consider an intermediate class when the confidence rates of the two classes are close for a single image to classify.

Finally, by testing the model on detected truck beds, frequent misclassifications were detected on images from the dataset provided by the researchers from Hong Kong.

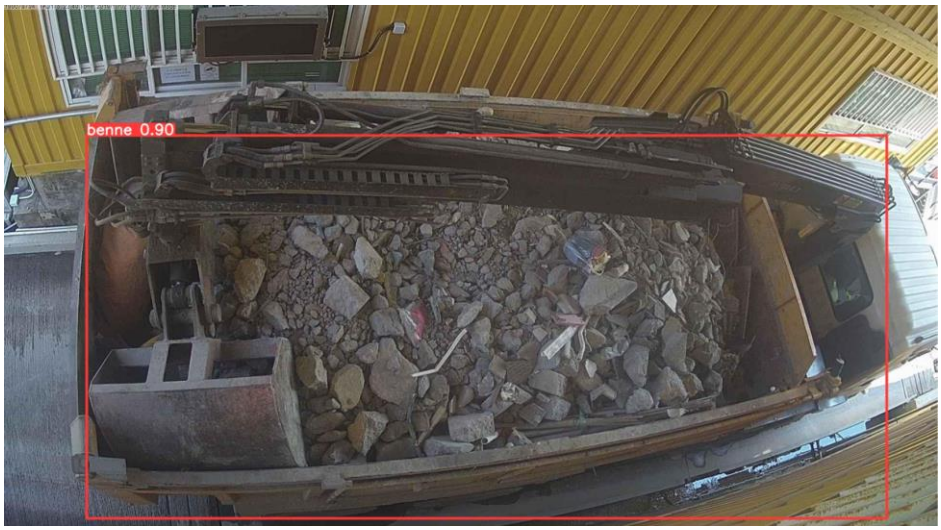


Figure 10: Truck bed detection on image provided by researchers from Hong Kong University.

Figure 11: Input image for the classification model before reduction.

As shown in Figure 10, a loader arm occupies a significant portion of the truck bed. The model needs to handle such noise present at the boarders or even within the truck bed. To mitigate this noise, a reduction of 20% of the region of interest was applied, focusing more on the center of the input image (which represents the material to classify). By doing this, we go from the image illustrated by figure 11 that has been classified as “Wood” due to the presence of the loader arm. To the image presented in Figure 12 classified as stone with 87% of confidence. A comparison of the performance was then made by adjusting the reduction rate. A rate of 50% was finally chosen to fully focus the classification model on the center of the truck bed.

Figure 12: Input image for the classification model after 20% reduction.

3.3.4.2 Algorithm programming of the classification model

Based on the literature review, we have seen several existing classification models. All of them are viable and their performance is mainly dependent on the fine-tuning of the image dataset. We have also seen that Resnet-50 is among the most popular and high performing, and YOLOv8n-cls is potentially the most performant in certain datasets. The first classification model was then built on Resnet-50 and reached an accuracy on the validation set of 83.23% after 50 epochs. We can compare the two models by creating another classification model with the same training dataset on YOLOv8n-cls. A precision of 89% is reached by YOLO after 50 epochs.

As noted in the previous section, the decision to fine-tune the dataset on YOLO was taken as YOLO seems to offer better initial performance and mainly a better computational time. As GPU access is not freely unlimited, computational units had to be bought for this research. When using a GPU, Google Colab debits a certain number of computational units depending on the time the GPU is activated. As we had not unlimited computational resources, the computational time was the major decision factor that drove our decision to YOLO.

Ultimately, a classification model was built using YOLOv8n-cls with around a thousand images of each class to train the model. This model achieved an overall precision of 94.26% on a validation set composed of images of soil, stone, wood, and sand.

3.3.4.3 Application development of the classification model

Now that we have our second model, which takes the output of the previous model of truck bed detection as input, what's next? With two functional models, we need to ensure that the entire

system operates effectively as a whole. The next and final chapter will be focused on the evaluation of the whole computer vision system on a test set designed to represent various real-world scenarios.

4. Evaluation and comparison of the approaches

Until now, each model (both from the initial approach and those used in our model chain system), have been evaluated independently with their own validation set. In this chapter, we will take back our test set that we built to evaluate and identify the best-trained weights of our model of truck bed detection. We will use this test set to evaluate both approaches and determine whether or not, dividing a task into several models by applying model chaining enhances performance in our industrial context with lack of diversity in the available data.

This test set is composed of 22 images of materials transported by trucks. 12 images were taken by a 360-degree camera; like the training set of the object detection models (from another dataset other than the one used to train our first model). This first part of the evaluation represents data on which the models are most likely to perform well, as they were trained in that context. Ten other images are from the dataset provided by the researchers from the University of Hong-Kong. These images represent a similar situation, as the pictures were taken by a camera on a gate. However, the context differs because the images are not taken by a 360-degree camera and are not taken exactly from above the truck bed; they are taken from above but slightly to the side.

The three main metrics that will be used during the whole evaluation will be 1) the precision in the material detection. Corresponding to the ability of the models to detect the transported materials for the initial approach and to detect the truck bed for our developed solution. 2) The precision of classification measuring the ability to detect the right class for the initial approach and to accurately classify the region of interest detected by the second model. The undetected materials/ truck bed will not be considered for the computation of the classification precision. Finally, 3) an overall precision will be calculated as the rate of precision for the whole process of rightly detecting and classifying the materials. Knowing that the two approaches do not have the exact same classes, the bricks and concrete of the initial approach are considered under the stone class of the classification model and the sand class of the model chaining system is considered as soil for the material detection model.

The hypothesis is that as the initial approach has been fully trained on a dataset provided by a 360-degree camera, it will perform better than the model chaining on the first part of the test. However, we expect that the material detection approach will suffer more from its lack of variety in its training data as we evaluate the models on images from other contexts through the

second part of the test. On the other hand, we hypothesize that the model chaining approach will better adapt to different unseen scenarios.

4.1 Test on similar images.

The first part of our test involves the 12 unseen images with a similar context to the ones used to train the models (from 360-degree cameras).

On the material detection side, the material detection approach performs well on this image set, reaching a precision of 100%. On its ability to predict the right class, a misclassification occurred once, dropping the classification accuracy to 91.67%. The figure 13 illustrates the occurring error; the soil represented in the truck bed is detected as concrete. (The full test of the initial approach is provided in the Appendix 5 and 6)

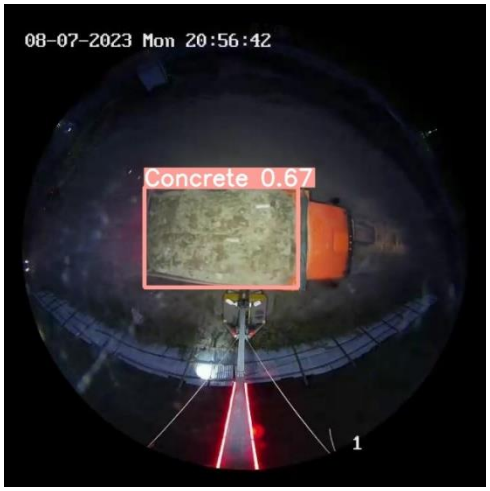


Figure 13: Misclassification of the materials detection model on a 360-degree camera.

On the other hand, our developed computer vision system detects the truck bed of this evaluation set without any mistakes. However, the precision of the classification aspect falls to 83.33% as 2 wrong predictions are made by our classification model. The two images corresponding to trucks carrying wood are classified as stone or sand.

These misclassifications can be explained by the fact that the training set of our classification model does not contain any images taken by a 360-degree camera. In these kinds of images, the objects represented suffer from deformations that can affect the model's performance. Especially for the classification of objects that can be distinguished by their shape such as the wood, unlike soil or sand that can be distinguished by their texture.

On this dataset, the approach of material detection on a single stage outperforms the model chain with an accuracy of 91.67% against 83.33%. These results were expected as the initial approach is more specialized in this type of data.

4.2 Test on other contexts.

This second phase of the evaluation mainly involves images taken by cameras placed on the side of a gate where the trucks transporting materials need to pass.

The former approach succeeds in detecting the materials 60% of the time in this dataset. Among the six images where the materials are detected, four of them are correctly classified.

For the latter approach, truck beds are detected at a rate of 80%, with one wrong classification among the detected regions of interest being declared.

From this set, we have then an overall precision of 40% for the single model system against 70% on our model chaining system. (The images tested on the material detection model can be found in the Appendix 7 and 8)

By considering the two parts of the evaluation set, we obtain an overall accuracy of 59.09% (13 images correctly predicted among the 22 tested) against 77.27% for our solution using model chaining (17/22). (Table 4)

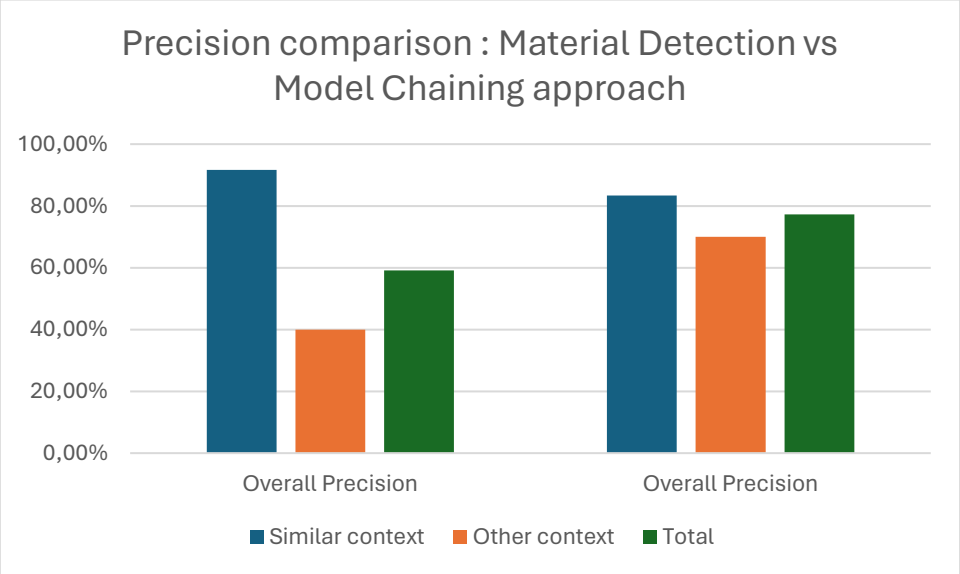


Table 4: Graph comparing the Material detection approach and the Model chaining approach.

Table 4 illustrates these percentages showing that in our case, the use of model chaining in our computer vision system to detect and classify transported materials in an industrial context with a lack of diversity in the data available, does indeed improve the performance of our system.

By dividing the initial task of material detection into two sequential models of truck bed detection and classification, a more flexible computer vision system was developed. This model chaining approach is more capable of handling and performing well on unseen data from various contexts.

5. Conclusion

In this thesis, we explored the application of model chaining in computer vision within an industrial context characterized by limited data diversity. This research was linked to Technord, a Belgian company aiming to develop a flexible solution for detecting materials. The literature review provided a comprehensive overview of various algorithms and techniques in object detection and classification, highlighting the performance of models such as YOLO, Faster R-CNN, Detectron2 and EfficientDet in different contexts. Additionally, the review discussed existing cases of model chaining, where sequential models were used to improve precision and/or compensate for a lack of data available.

In this dissertation, two approaches were tested. The first consisted of implementing a model of object detection to directly detect the material within a truck bed. This model was trained on a dataset extracted from Roboflow and contained around 1300 images captured by a 360-degree camera placed on a gate above the materials transported by trucks. We have seen from the results that it represents the most suitable model for the aim of detecting and classifying materials as long as we are in a context close to the one it knows.

As soon as we diverge from this well-known context by using images taken on another site using another camera and angle to capture the images, the performance of the model of material detection are directly impacted. It first struggles to detect the materials and tends to more foten make wrong predictions about the class decision.

On the other hand, the second approach consisted of another computer vision system that divided the task of materials detection into two models. The first model aims to detect the truck bed that transports the materials. The coordinates corresponding to the region of interest that we want to classify are then converted before serving as input for the second model that will extract the region of interest to classify the type of materials represented in the truck bed.

By going through this model chaining process, we used two different models. Each of them having their own task and their own training, making the whole system more flexible and capable of handling unknown data.

Limits and future work

The adoption of the concept of model chaining did compensate for the lack of variety in the data available but it does not represent a miracle solution either. Some solutions can be explored by the company or researchers to further deal with this lack of variety in data.

The first one which is more of advice for the company, is to find more data. Adding more images of the client's trucks with their materials to compensate for the issue of variety will help the computer system to perform better and improve its prediction. By adding that kind of images to the model of truck bed detection, the upgraded model will become more specialized in a defined context. Therefore, the model will be more suited for the client's specific context. These same images can also be used for the classification part. All the training images used for the classification model of our system were taken "in the nature". Adding images of stone, soil, sand, and wood taken in a specified context, will also enhance the classification model's efficiency.

The second one is an opportunity to explore a limit of this thesis in future research. Except for a few data provided by Roboflow for the classification model, no data augmentation techniques have been used in our research. It has been proven that using data augmentation techniques can improve a model's performance (Zoph et al., 2020). It is notably used for applications where a lack of sufficient amount of data available is noticed. However, this industrial case of materials detection lacks data but especially in diversity of data. A future work could consist of applying data augmentation techniques either on the first approach of material detection or the first model of our model chaining (truck bed detection) to see if it enhances the system's performance. However, it is important to note that data augmentation can be a time-consuming operation, especially in the field of object detection, as the annotations needs to be modified depending on the transformations executed.

The study focuses on specific models (YOLOv9 and YOLOv8n-cls). While these models are powerful and enabled us to conduct this research, their architectures were not explored in depth. The results might differ with the use of alternative models or by optimizing other parameters influencing the convolutional neural networks (CNNs) used.

Another limit of this work is linked to the evaluation of the approaches. The evaluation set used to test the approaches consists of a relatively small number of images (22 images). The small size of this dataset is explained by the lack of available data of transported materials linked to an industrial context. However, relying on such a small dataset for evaluation can be risky, as it may not capture the full diversity and complexity of real-world scenarios. This could lead to overestimating the model's performance and robustness. For more reliable assessment, a larger and more varied datasets are necessary to ensure the model can generalize well to different unseen contexts.

Finally, a class that initially had to be implemented in the computer vision system is the “empty body” one (empty truck bed). Unfortunately, it had to be discarded. It was determined that it was unfeasible with the available data. Indeed, in addition to the lack of data available, it is hard for the model to determine what an empty body is for a general application. Depending on the cleanliness of the truck bed, the brightness, the presence of shadows or even the type of the truck bed, the model can easily confuse an empty body with another class. To implement this class, the company would need a series of images corresponding to a typical empty body of the client’s trucks.

6. Bibliography.

- Wu, X., Sahoo, D., & Hoi, S. C. H. (2020). Recent advances in deep learning for object detection. *Neurocomputing*, *396*, 39–64. https://doi.org/10.1016/j.neucom.2020.01.085
- Zhao, Z., Zheng, P., Xu, S., & Wu, X. (2019). Object Detection with Deep Learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232. https://doi.org/10.1109/tnnls.2018.2876865
- Jabir, B., Falih, N., & Rahmani, K. I. (2021). Accuracy and efficiency comparison of object detection Open-Source models. *International Journal of Online and Biomedical Engineering*, *17*(05), 165. https://doi.org/10.3991/ijoe.v17i05.21833
- Bansal, A., 1, Sikka, K., Sharma, G., University of Maryland, SRI International, & NEC Labs America. (2018). Zero-Shot Object Detection. In *University of Maryland, College Park, MD* [Journal-article]. https://openaccess.thecvf.com/content_ECCV_2018/papers/Ankan_Bansal_Zero-Shot_Object_Detection_ECCV_2018_paper.pdf
- Li, W. (2021). Analysis of object detection performance based on faster R-CNN. *Journal of Physics: Conference Series*, *1827*(1), 012085. https://doi.org/10.1088/1742-6596/1827/1/012085
- Wang, X., Gao, H., Jia, Z., & Li, Z. (2023). BL-YOLOV8: an improved road defect detection model based on YOLOV8. *Sensors*, *23*(20), 8361. https://doi.org/10.3390/s23208361
- Tamang, S., Sen, B., Pradhan, A., Sharma, K., & Singh, V. K. (2023, February 17). *Enhancing COVID-19 safety: Exploring YOLOV8 object detection for accurate face mask classification*. https://ijisae.org/index.php/IJISAE/article/view/2966
- Youssef, A. (2002). Analysis and comparison of various image downsampling and upsampling methods. *The George Washington University*. https://doi.org/10.1109/dcc.1998.672325
- Xiao, B., Nguyen, M., & Yan, W. Q. (2023). Fruit ripeness identification using YOLOv8 model. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-023-16570-9

- Xiao, B., Nguyen, M., & Yan, W. Q. (2023). Fruit ripeness identification using YOLOv8 model. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-023-16570-9

- Merz, G., Liu, Y., Burke, C. J., Aleo, P. D., Liu, X., Kind, M. C., Kindratenko, V., & Liu, Y. (2023). Detection, instance segmentation, and classification for astronomical surveys with deep learning (deepdisc) : detectron2 implementation and demonstration with Hyper Suprime-Cam data. *Monthly Notices Of The Royal Astronomical Society*, *526*(1), 1122-1137. https://doi.org/10.1093/mnras/stad2785
- Evangelista, I. R. S., Catajay, L. T., Palconit, M. G. B., Bautista, M. G. A., Concepcion, R., Sybingco, E., Bandala, A. A., & Dadios, E. P. (2022). Detection of Japanese Quails (Coturnix japonica) in Poultry Farms Using YOLOv5 and Detectron2 Faster R-CNN. *Journal Of Advanced Computational Intelligence And Intelligent Informatics*, *26*(6), 930-936. https://doi.org/10.20965/jaciii.2022.p0930
- Jabir, B., Falih, N., & Rahmani, K. I. (2021). Accuracy and Efficiency Comparison of Object Detection Open-Source Models. *International Journal Of Online And Biomedical Engineering*, *17*(05), 165. https://doi.org/10.3991/ijoe.v17i05.21833
- Zhou, D., Zhou, X., Zhang, H., Yi, S., & Ouyang, W. (2020). Cheaper Pre-training Lunch : An Efficient Paradigm for Object Detection. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2004.12178
- Chen, W., & Shah, T. (2021b). Exploring low-light object detection techniques. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2107.14382
- Hsiang, H., Chen, K., Li, P., & Chen, Y. (2020). Analysis of the Effect of Automotive Ethernet Camera Image Quality on Object Detection Models. *IEEE*. https://doi.org/10.1109/icaiic48513.2020.9065232
- Terven, J. R., Córdova-Esparza, D., & Romero-González, J. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision : From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning And Knowledge Extraction*, *5*(4), 1680-1716. https://doi.org/10.3390/make5040083
- Evangelista, I. R. S., Catajay, L. T., Palconit, M. G. B., Bautista, M. G. A., Concepcion, R., Sybingco, E., Bandala, A. A., & Dadios, E. P. (2022b). Detection of Japanese Quails (Coturnix japonica) in Poultry Farms Using YOLOv5 and Detectron2 Faster R-CNN. *Journal Of Advanced Computational Intelligence And Intelligent Informatics*, *26*(6), 930-936. https://doi.org/10.20965/jaciii.2022.p0930
- Priyadi, M. R. A., & Suharjito, S. (2023). Comparison of YOLOv8 and EfficientDet4 Algorithms in Detecting the Ripeness of Oil Palm Fruit Bunch. *IEEE*. https://doi.org/10.1109/iciss59129.2023.10291928

- Akshayanivashini, C. V., Anand, R., & Karthikraj, S. C. (2023). Steel Strip Surface Defect Detection and Segmentation Using Detectron2. *Social Science Research Network*. https://doi.org/10.2139/ssrn.4453087
- Başţanlar, Y., & Özuysal, M. (2013). Introduction to Machine Learning. Dans *Methods in molecular biology* (p. 105-128). https://doi.org/10.1007/978-1-62703-748-8_7
- Khan, A. I., & Al-Habsi, S. (2020). Machine Learning in Computer Vision. *Procedia Computer Science*, *167*, 1444-1451. https://doi.org/10.1016/j.procs.2020.03.355
- Khan, A. A., Laghari, A. A., & Awan, S. A. (2018). Machine Learning in Computer Vision: A review. *ICST Transactions on Scalable Information Systems*, 169418. https://doi.org/10.4108/eai.21-4-2021.169418
- *Key differences between computer vision and machine learning*. (n.d.). Kili-website. https://kili-technology.com/data-labeling/computer-vision/computer-vision-and-machine-learning-differences
- Mahadevkar, S. V., Khemani, B., Patil, S., Kotecha, K., Vora, D., Abraham, A., & Gabralla, L. A. (2022). A review on Machine Learning Styles in Computer Vision—Techniques and Future Directions. *IEEE Access*, *10*, 107293–107329. https://doi.org/10.1109/access.2022.3209825
- Teki, S. (2024, March 3). *Data Preparation for Machine Learning: Ultimate Guide*. DagsHub Blog. https://dagshub.com/blog/the-ultimate-guide-to-data-preparation-for-machine-learning/#:~:text=Data%20preparation%20involves%20multiple%20processes,feature%20processing%20and%20data%20governance.
- Singh, B., Najibi, M., Sharma, A., & Davis, L. S. (2021). Scale Normalized Image Pyramids with AutoFocus for Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1. https://doi.org/10.1109/tpami.2021.3058945
- Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T., Shlens, J., & Le, Q. V. (2020). Learning data augmentation Strategies for object detection. In *Lecture notes in computer science* (pp. 566–583). https://doi.org/10.1007/978-3-030-58583-9_34
- Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *EM*, *31*(3), 685–695. https://doi.org/10.1007/s12525-021-00475-2
- Sharifani, K., Amini, M., MahamGostar Research Group, University of North Carolina at Charlotte, & University Technology Malaysia (UTM). (2023). Machine Learning and Deep Learning: A review of Methods and applications. *World Information Technology and Engineering Journal*, *10*(07). https://www.researchgate.net/profile/Mahyar-

Amini/publication/371011515_Machine_Learning_and_Deep_Learning_A_Review_of_Methods_and_Applications/links/646ef1106a0082273fa8d97b/Machine-Learning-and-Deep-Learning-A-Review-of-Methods-and-Applications.pdf

- Albawi, S., Bayat, O., Al-Azawi, S., & Uçan, O. N. (2018). Social touch gesture recognition using convolutional neural network. *Computational Intelligence and Neuroscience*, *2018*, 1–10. https://doi.org/10.1155/2018/6973103
- Introduction to Convolutional Neural Networks. (n.d.). *National Key Lab for Novel Software Technology*. https://cs.nju.edu.cn/wujx/paper/CNN.pdf
- Gurucharan, M. (2024, April 29). Basic CNN Architecture: Explaining 5 layers of convolutional neural Network. *upGrad blog*. https://www.upgrad.com/blog/basic-cnn-architecture/
- Krishnamurthy, B. (2024, February 26). *An introduction to the RELU activation function*. Built In. https://builtin.com/machine-learning/relu-activation-function
- CNRS - Formation FIDLE. (2022, November 27). *Seq. 02 / Les CNN (Live)* [Video]. YouTube. https://www.youtube.com/watch?v=S8gCPOIFYfM
- Zeng, L. (2023). The development of image classification models based on computer vision. *Highlights in Science, Engineering and Technology*, *34*, 430–434. https://doi.org/10.54097/hset.v34i.5505
- Poudel, S., & Poudyal, P. (2022). Classification of Waste Materials using CNN Based on Transfer Learning. *Kathmandu University*.https://doi.org/10.1145/3574318.3574345
- Andrew, A., & Santoso, H. (2022). Compare VGG19, ReSNet50, Inception-V3 for review food rating. *Sinkron*, *7*(2), 845–494. https://doi.org/10.33395/sinkron.v7i2.11383
- Pan, Y., Liu, J., Cai, Y., Yang, X., Zhang, Z., Li, H., Zhao, K., Yu, L., Zeng, C., Duan, J., Xiao, P., Li, J., Cai, F., Yang, X., & Tan, Z. (2023). Fundus image classification using Inception V3 and ResNet-50 for the early diagnostics of fundus diseases. *Frontiers in Physiology*, *14*. https://doi.org/10.3389/fphys.2023.1126780
- Nyarko, B. N. E., Wu, B., Zhou, J., Agordzo, G. K., Odoom, J., & Koukoyi, E. (2022). Comparative analysis of AlexNet, Resnet-50, and Inception-V3 models on masked face recognition. *2022 IEEE World AI IoT Congress (AIIoT)*. https://doi.org/10.1109/aiiot54504.2022.9817327
- Sikati, J., & Christian, N. J. (2023). YOLO-NPK: A Lightweight Deep Network for Lettuce Nutrient Deficiency Classification Based on Improved YOLOv8 Nano. *MDPI*. https://doi.org/10.3390/ecsa-10-16256

- Bald, M. (2023, September 27). *Working Smarter with Machine Learning Model Chains | Wallaroo.AI*. Wallaroo.AI. https://wallaroo.ai/working-smarter-with-machine-learning-model-chains/#:~:text=Model%20chains%20are%20essentially%20two,input%20of%20the%20second%20model
- Moradi, S., Zayed, T., & Golkhoo, F. (2018). Automated sewer pipeline inspection using computer vision techniques. *Pipelines 2018*. https://doi.org/10.1061/9780784481653.064
- Karlinsky, L., Dinerstein, M., Harari, D., & Ullman, S. (2010). The chains model for detecting parts by their context. *Weizmann Institute of Science*. https://doi.org/10.1109/cvpr.2010.5540232
- *The value chain of machine learning*. (n.d.). https://levity.ai/blog/machine-learning-value-chain

WongKinYiu. (n.d.). *GitHub - WongKinYiu/yolov9: Implementation of paper - YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*. GitHub. https://github.com/WongKinYiu/yolov9

- Chen, J., Lu, W., Li, Y., Wu, Y., & Xue, F. (2022). Estimating construction waste truck payload volume using monocular vision. *Resources, Conservation and Recycling*, *177*, 106013. https://doi.org/10.1016/j.resconrec.2021.106013
- Pusarla, V. (2021, December 30). Object Detection using Regions with CNN features - Analytics Vidhya - Medium. *Medium*. https://medium.com/analytics-vidhya/object-detection-using-regions-with-cnn-features-557392e22f84
- *Multi-model composition with Ray Serve deployment graphs | Anyscale*. (n.d.). Anyscale. https://www.anyscale.com/blog/multi-model-composition-with-ray-serve-deployment-graphs
- Wang, C., Yeh, I., & Liao, H. M. (2024). YOLOV9: Learning what you want to learn using programmable gradient information. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2402.13616
- Sharma, R. (2022, January 7). EFfIcientDET: Scalable and efficient object Detection. *Medium*. https://medium.com/analytics-vidhya/efficientdet-scalable-and-efficient-object-detection-384a5df9011a
- ReachIt. (s. d.). *Convolution Neural Network (with Use Case - Pytorch)*. https://www.reachiteasily.com/2021/02/convolutional-neural-network.html

- *Role of Artificial Neural Network in Artificial Intelligence*. (s. d.). https://www.turing.com/kb/importance-of-artificial-neural-networks-in-artificial-intelligence
- Haque, K. N. (2023, 3 avril). *What is Convolutional Neural Network — CNN (Deep Learning)*. https://www.linkedin.com/pulse/what-convolutional-neural-network-cnn-deep-learning-nafiz-shahriar/
- Girshick, R. (2015). Fast R-CNN. *IEEE*. https://doi.org/10.1109/iccv.2015.169
- Nanos, G. (2024, March 18). *Deep neural networks: Padding*. Baeldung. https://www.baeldung.com/cs/deep-neural-networks-padding

7. Appendix.

Appendix 1: Further explanations of EfficientDet

To achieve its performance level, the algorithm combines three primary components. The first one is the Bidirectional Feature Pyramid Network (BiFPN) which facilitates swift and efficient multi-scale feature fusion. BiFPN enhances the model's feature fusion process by enabling the network to discern the importance of various input image features at various scale to enhance its ability to detect objects across a range of sizes. (Tan et al., 2019,p 10782-10783) The second element is the compound scaling method. Instead of using heuristic or manual methods like traditional models, EfficientDet's compound scaling method uniformly adjusts the network's depth, width and resolution using a compound coefficient, ensuring an optimal balance between accuracy and computational resources. This scaling method is applied to the backbone network (the part of the model in charge of the process and feature extraction through convolutional layers from the input image) and the BiFPN. (Tan et al., 2019,p 10784) The third key element contributing to EfficientDet's effectiveness is its refined and optimized training process. This approach encompasses a series of advanced optimization techniques and training strategies that not only reduce computational time but also boosts the model's capacity to process and recognize patterns in new and unseen images. Through these strategic implementations, EfficientDet ensures its ability to deliver state-of-the-art object detection performance efficiently, making it a valuable asset in the field of object detection and computer vision.

Appendix 2: Testing phase of Detectron2 for a bottle detection

A first model has been developed in order to test how easy Detectron2 is to use (for the download and the use of all its packages) and how efficient it is. As Detectron2 works with pre-trained models, a pre-trained model of faster R-CNN has been selected to create a test model aiming to detect bottles of every type (bottle of water, beer, wine, broken, plastic crushed, etc).

This first training has then been realized by adding one class "bottle" and 98 manually annotated images of bottles (annotations done through the software developed by the Oxford University https://www.robots.ox.ac.uk/~vgg/software/via/). As it was initially a test, some parameters have been dragged down in order to reduce the training time. It has been done by setting the number of ROI (regions of interest) to 128 and the number of iterations to 300. The code used

for this model has been inspired by the one realised by **Aarohi Singla** (https://github.com/AarohiSingla/Detectron2-Tutorial/blob/main/Detectron_maskrcnn_custom_dataset_baloon.ipynb). With some differences to fit it to our own dataset annotated in another format of Json.

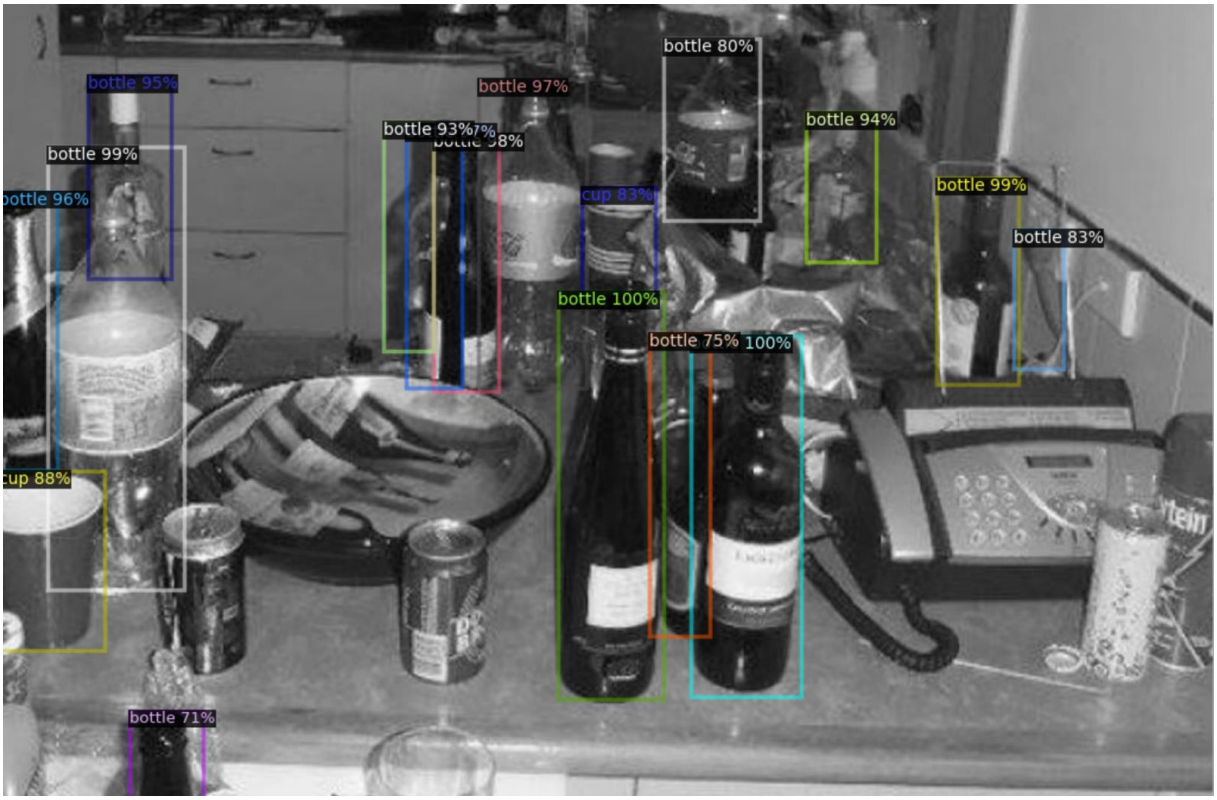


Figure A: Output of the first model of Detectron2

This initial training session concluded in 1 hour and 13 minutes, and the model demonstrated satisfactory performance on a manually annotated validation set.

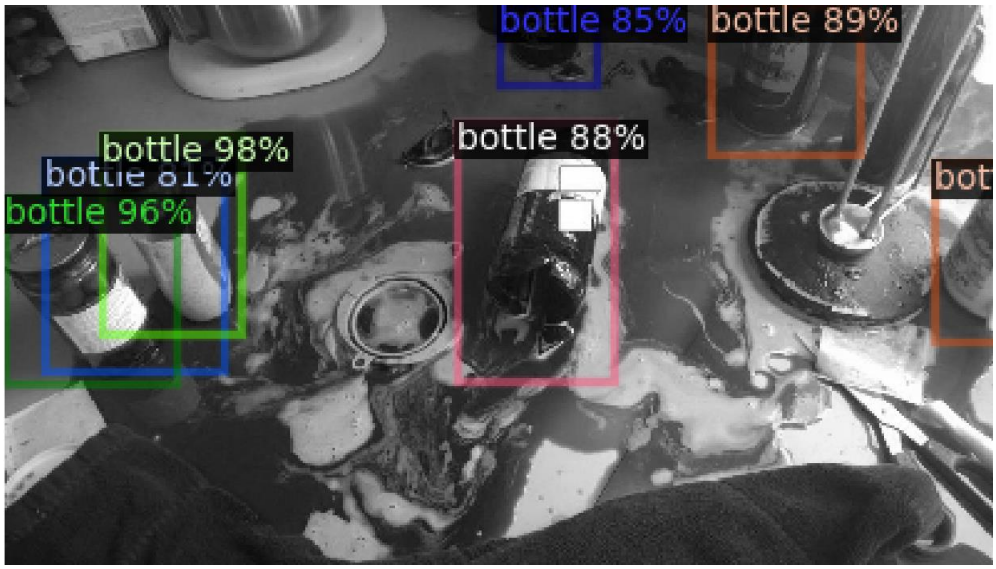


Figure B: Confusion of class of the first model of Detectron2

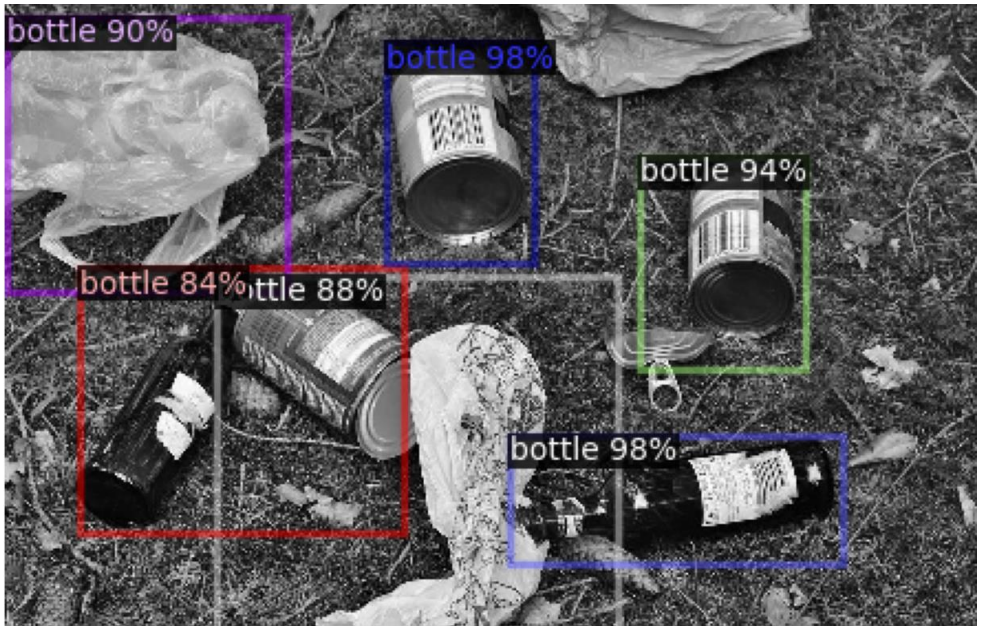


Figure C: Confusion of class of the first model of Detectron2

Figure D: Confusion of class of the first model of Detectron2

However, the model exhibited some inaccuracies such as misclassifying objects. Through the figures B to D, we can see that the model tends to detect and classify some objects like glasses, plastic bags or cans as bottles. This confusion is likely due to the small size of the training dataset and the model's singular focus on the bottle class. Training our model with other datasets corresponding to other classes which for the model tends to confuse, would help him to see the difference between these classes.

Figure E: Weaknesses of the first model of Detectron2

In the figure 5, we can see that in some images with a lot of different objects and contrast, the model struggles to identify the targets. This could be solved by adding more similar data in the training set; with images of garbage in the forest.

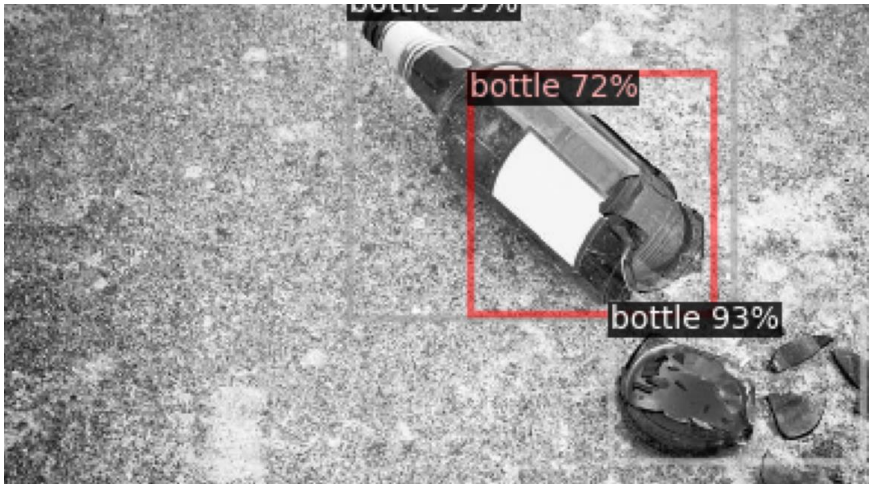
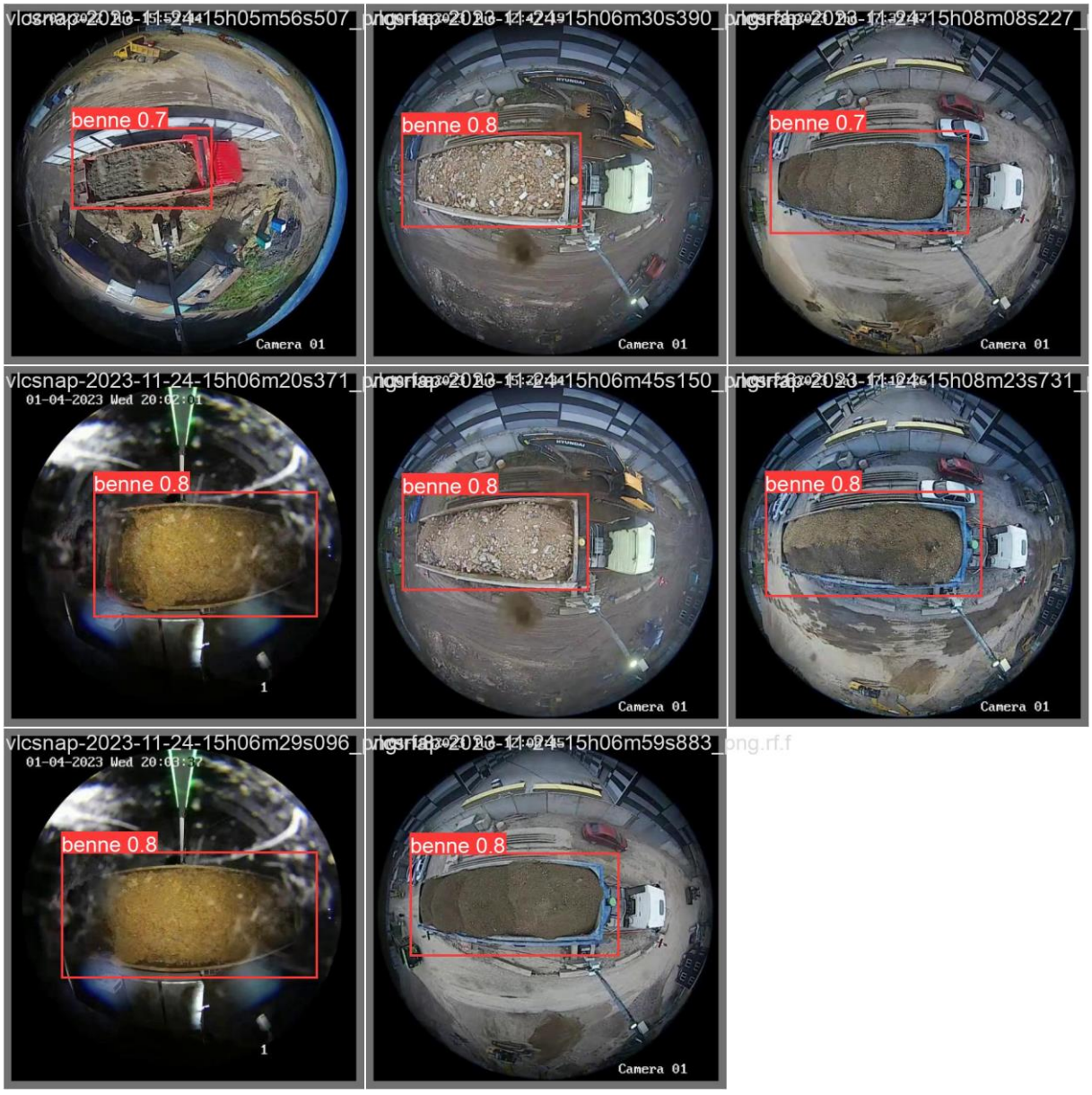


Figure F: Broken bottle detection succeeded.

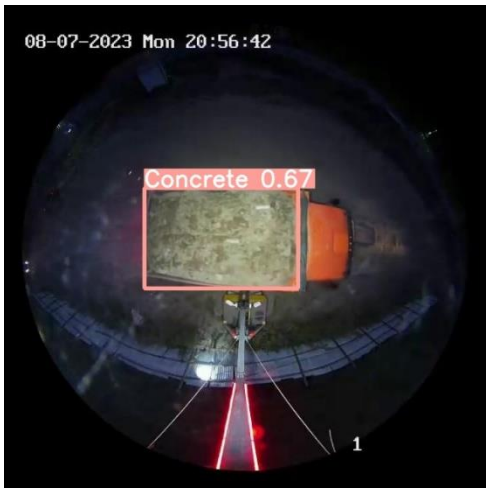
Figure G: Bottle detection failed

In the figure F, we can see that the model succeeded to spot the broken bottle while it couldn't spot it in the figure G. This could be linked either to the lack of training data or the bad quality of the figure G.

Appendix 3: Predictions of the truck bed detection model on a sample of the validation set.

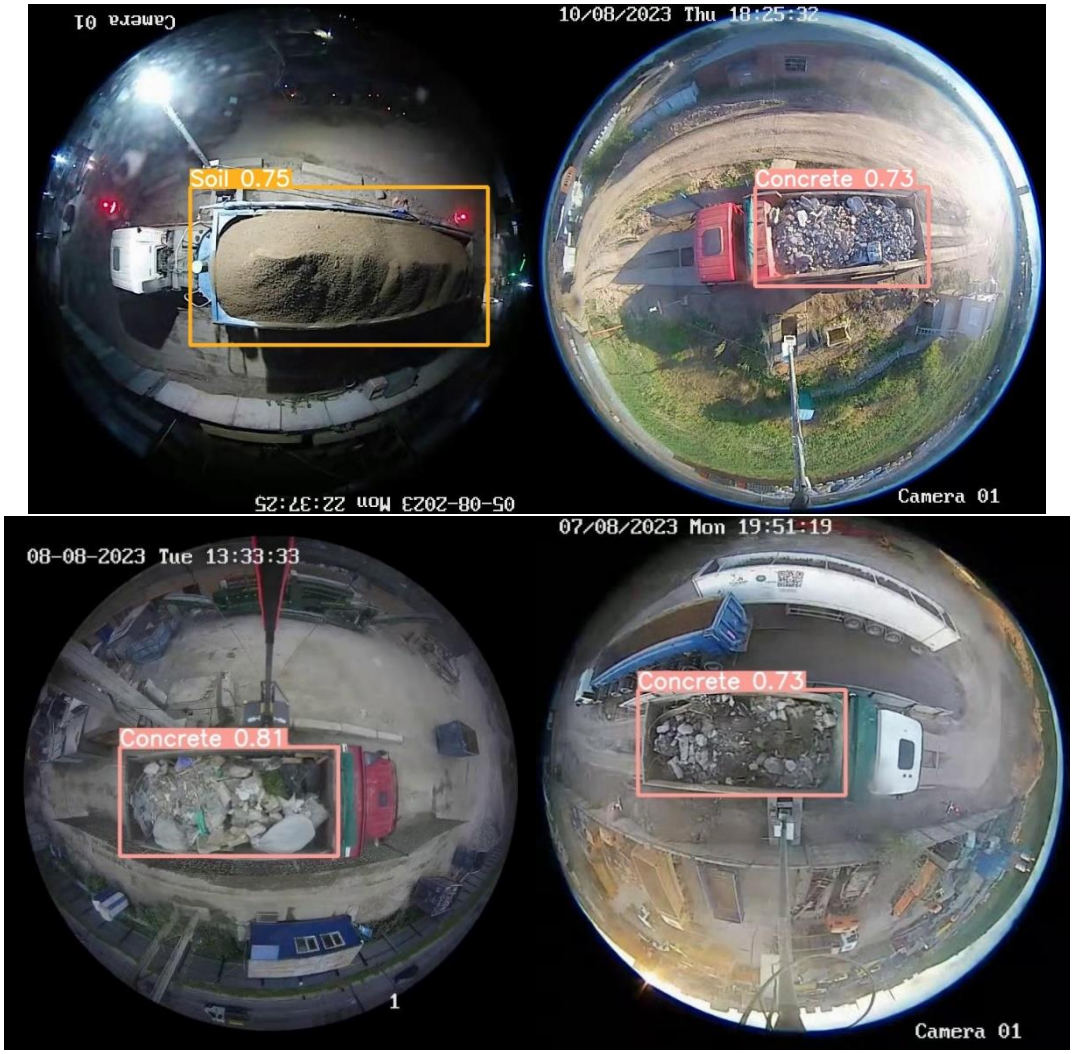


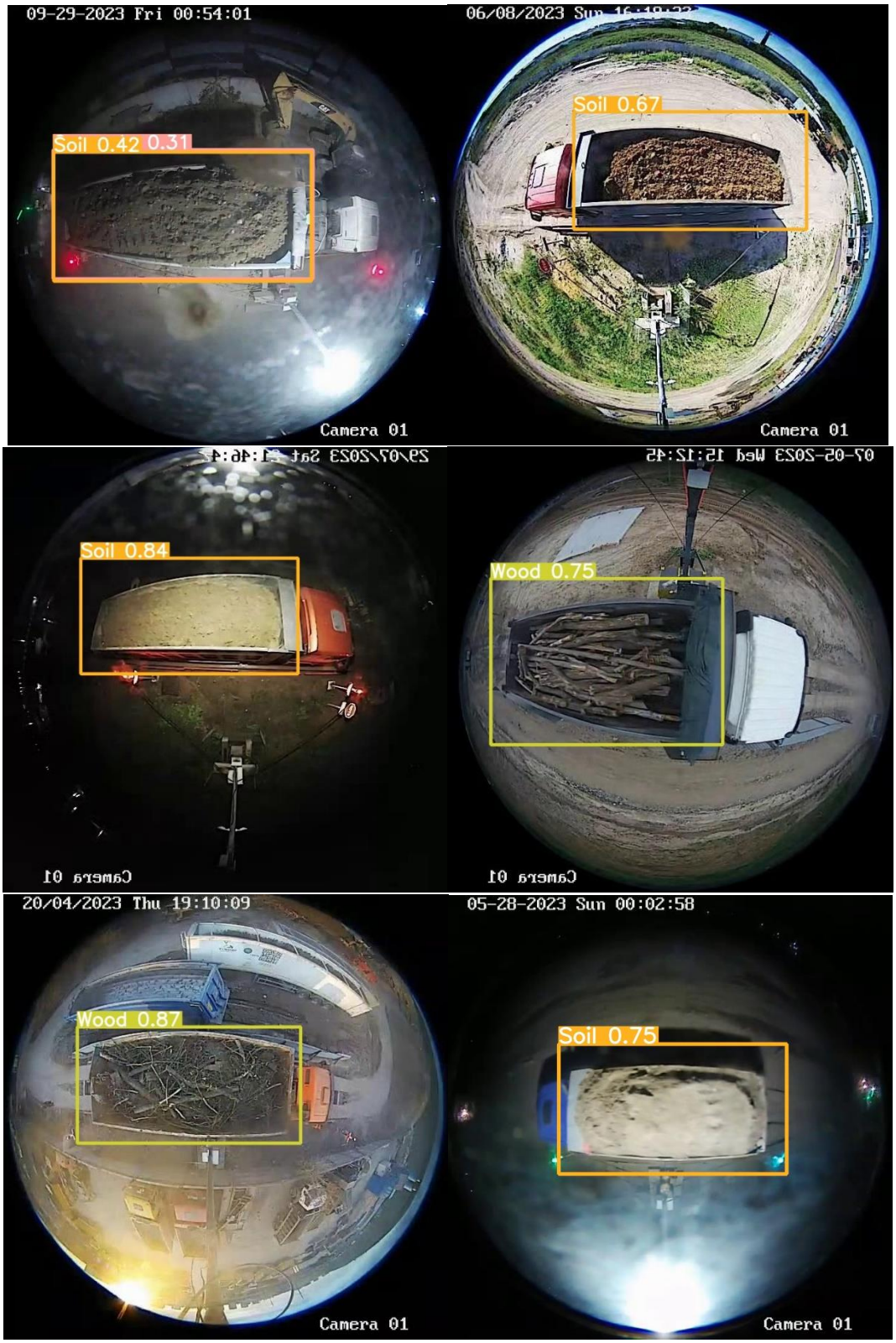
Appendix 4: Fails of the initial approach on images of 360-degree images



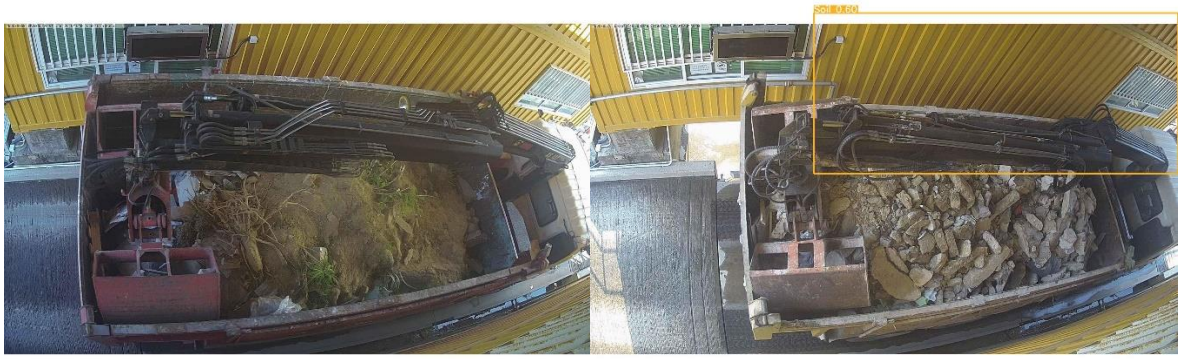
Appendix 6

Appendix 5: Success of the initial approach on images of 360-degree images




09-29-2023 Fri 00:54:01
Soil 0.42
0.31
Camera 01
Soil 0.67
Camera 01
Soil 0.84
Camera 01
Wood 0.75
Camera 01
20/04/2023 Thu 19:10:09
Wood 0.87
Camera 01
05-28-2023 Sun 00:02:58
Soil 0.75
Camera 01

Appendix 6: Fails of the initial approach on images provided by Hong Kong university.



Appendix 7: Success of the initial approach on images provided by Hong Kong university.

Abstract: In recent years, the integration of computer vision in industrial applications has significantly advanced, driven by the need for enhanced automation and efficiency. This thesis, completed in partnership with a Belgian company, aims to develop a computer vision solution for automating the detection and classification of truck loads in an industrial context characterized by limited data diversity.

An initial approach was established to directly detect the transported materials. However, this approach suffered from the lack of diversity in the available data, as the training images were predominantly from a single 360-degree camera. To address this issue, the thesis brings a model chaining approach, combining a truck bed detection model with a material classification model. This method leverages the strengths of separate models for detection and classification, each trained on different data to enhance system flexibility and performance.

Résumé : Ces dernières années, l'intégration du computer dans les applications industrielles a considérablement progressé, motivée par le besoin accru d'automatisation et d'efficacité. Cette thèse, réalisée en partenariat avec une entreprise belge, vise à développer une solution de computer afin d'automatiser la détection et la classification de materiaux transportés par des camions dans un contexte industriel caractérisé par un manque de diversité dans les données disponibles.

Une approche initiale a été établie pour détecter directement les matériaux transportés. Cependant, cette approche a souffert du manque de diversité des données disponibles; les images d'entraînement provenant principalement d'une seule caméra à 360 degrés. Pour remédier à ce problème, la thèse propose une approche de Model chaining combinant un modèle de détection de la benne de camion avec un modèle de classification des matériaux. Cette méthode exploite les avantages de modèles distincts pour la détection et la classification, chacun étant entraîné sur des données différentes pour améliorer la flexibilité et la performance du système.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management
Chaussée de Binche 151, 7000 Mons, Belgique | **www.uclouvain.be/lsm**