	Compulsory Summer Internship Report	Code : DO-PFE-01
		Indice de révision : 00
		Edition : 07/2022

Compulsory Summer Internship Report

Field: Industrial Computing and Automation

Level: 4th Year

Subject :

Leveraging Vision-Language Models (VLMs) for Automated Image Annotation in Shop-Analytics

Realized By : **Siwar Gharbi**

Host Company :

anavid®

Academic year : 2024-2025

	Compulsory Summer Internship Report	Code : DO-PFE-01
		Indice de révision : 00
		Edition : 07/2022

Compulsory Summer Internship Report

Field: Industrial Computing and Automation

Level: 4th Year

Subject :

Leveraging Vision-Language Models (VLMs) for Automated Image Annotation in Shop-Analytics

Realized By : **Siwar Gharbi**

Entreprise d'accueil :

anavid®

Responsable à l'entreprise:	*Avis de la commission des stages*
M. Mohamed Amine Ben Abdeljelil	

Academic Year :2024-2025

Acknowledgments:

I would like to express my sincerest thanks and deep gratitude to my supervisor

Mohamed Amine Ben Abdeljelil for his guidance and valuable advice throughout my project at Anavid. The trust and knowledge they shared with me were invaluable.

This experience allowed me to develop a thorough understanding of Vision Language Models (VLMs) and to acquire valuable skills to approach technical challenges in a methodical and innovative way.

I sincerely hope to stay in contact with the Anavid team and look forward to the opportunity to collaborate again in the future. Thank you once again for this exceptional two-months internship.

Abstract:

This report provides an overview of my two-month internship at Anavid , where I worked on developing a fully automated data annotation pipeline using Vision Language Models (VLMs). The primary objective was to design and implement a Proof of Concept and optimize it using a Vision Language as a start Gemini 2.5 then research into lightweight VLMs to enable us the deployment of annotation models on local servers, reducing dependency on external services. During the internship, I applied computer vision algorithms and researched multiple Vision-Language Models, evaluating and deploying the most suitable ones for the task. I integrated these models into a processing pipeline that combined traditional CV techniques with VLM outputs to optimize accuracy and efficiency.

The internship culminated in the successful development of a fully automated image annotation pipeline. This experience enhanced my technical skills and deepened my understanding of AI applications in vision-oriented tasks.

Table of Contents:

List of Figures and Tables:

List of Figures:

List of Tables:

Glossary of Acronyms:

1)**VLMs**:Vision Language Models:are artificial intelligence (AI) models that blend computer vision and natural language processing (NLP) capabilities.

2)**AI** : Artificial intelligence:is technology that enables computers and machines to simulate human learning, comprehension, problem solving, decision making, creativity and autonomy.

3)**CCTV**:Closed-Circuit Television is a video surveillance technology that transmits video signals to a limited set of monitors.

4)**POS**:point of sale, enables merchants to process payments and log transactions.

5)**IDE**:Integrated Development Environment:s software that combines commonly used developer tools into a compact GUI (graphical user interface) application. It is a combination of tools like a code editor, code compiler, and code debugger with an integrated terminal.

6) **JSON**: JavaScript Object Notation : is a lightweight format for storing and transporting data.

7)**CNNs**:Convolutional Neural Networks are deep learning models designed to process data with a grid-like topology such as images. They are the foundation for most modern computer vision applications to detect features within visual data.

8)**API:** application programming interface, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features and functionality.

9) **C2F** :Convolutional Feature : is a module for object detection. is designed to enhance feature extraction by capturing more complex patterns in the data, which is crucial for improving detection accuracy.

10)**SPPF** :Spatial Pyramid Pooling-Fast: modules help increase the receptive field without a significant increase in computational demands. This allows the network to

integrate contextual information from a larger area of the input image, which is beneficial for detecting objects at various scales.

13**)Idefics3:i**s an open multimodal model that accepts arbitrary sequences of image and text inputs and produces text outputs. The model can answer questions about images, describe visual content, create stories grounded on multiple images, or simply behave as a pure language model without visual inputs.

14)**Llama-3.1-8B** :The Meta Llama 3.1 collection of multilingual large language models (LLMs) is a collection of pretrained and instruction-tuned generative models in 8B, 70B and 405B sizes (text in/text out).

15)**SmolLM2 1.7B:**SmolLM2 is a family of compact language models available in three sizes: 135M, 360M, and 1.7B parameters. They are capable of solving a wide range of tasks while being lightweight enough to run on-device.

16)**SigLIP:**SigLIP is a multimodal image-text model similar to CLIP. It uses separate image and text encoders to generate representations for both modalities.

17)**GPU:** graphical processing unit: is an electronic circuit designed to speed computer graphics and image processing on various devices. These devices include video cards, system boards, mobile phones and personal computers (PCs).

18)**CPU**: Central Processing Unit is the primary component of a computer that performs most of the processing inside the computer. It executes instructions from programs and manages the operations of other hardware components. The CPU is often referred to as the "brain" of the computer, as it carries out the basic arithmetic, logic, control, and input/output operations specified by the instructions.

19)**RAM :**Random Access Memory, also called "volatile memory," is a component that allows your computer to store short-term data for faster access.

1.General Introduction :

Vision-Language Models (VLMs) [1]are advanced deep learning systems that integrate computer vision and natural language processing to understand and describe visual content in human language. By leveraging deep neural networks, VLMs[1] excel at interpreting complex image features alongside textual information, enabling applications such as automatic image annotation, captioning, and multimodal understanding.

During my internship at Anavid, I worked on a project focused on developing an intelligent annotation system based on Vision-Language Models (VLMs)[1] to automate and improve the accuracy of labeling images in datasets for computer vision tasks. The main goal was to generate high-quality data for the SA (Scene Analysis) project, which aims to train AI models to automatically understand and interpret visual environments, including objects, spatial relationships, and activities. By combining deep learning with vision-language integration, the system made data annotation faster, more efficient, and more accurate.

2. Overview of the Host Company:

2.1.Anavid – Company Overview:

Founded in 2018, Anavid is a DeepTech company using AI and computer vision to help retailers reduce losses and increase sales by enhancing the customer experience. Their General Data Protection Regulation (GDPR)-compliant solutions leverage existing video surveillance infrastructure to generate real-time data and provide actionable insights. Anavid's platform includes AI-powered **shoplifting prevention**, which monitors stores in real time, sends instant alerts to staff, and deters recurring theft, as well as **store analytics**, turning surveillance and POS data into actionable insights to optimize operations, inventory, and the overall shopping experience.

Figure 1 :Anavid's logo

3.Specifications:

3.1.Problem Statement:

Anavid Shop Analytics is a platform that collects in-store data in real time, enabling to improve the customer experience and increase customer satisfaction. So this platform assures in the first place the collection of the data , it integrates seamlessly with your existing infrastructure, such as CCTV

cameras, POS systems, and other in-store technologies. It collects real-time data on customer behavior, sales trends, and operational performance.Then,once the data is collected, it analyzes the information to identify patterns, trends, and opportunities that might otherwise go unnoticed and as a final step delivers clear, actionable reports and recommendations. From optimizing inventory to improving staff schedules, these insights help you make data-driven decisions that boost efficiency, sales, and customer satisfaction. So for that, we need to annotate our collected real-time data to use in the AI-based analysis to gather the number of visitors, average visit duration, or even perform gender classification.

3.2.Project Overview:

The main objective throughout my internship at Anavid was to develop a fully functional annotation pipeline using a Deep Learning[2] model combined with a Vision-Language Model (VLM)[1], implemented in Python with Visual Studio Code (VSCode)[3] as the IDE, in order to annotate data collected from cameras and specify key attributes, primarily age and gender. Following the development, I focused on better understanding and researching Vision-Language Models (VLMs) to optimize and refine the pipeline.

3.3.Technologies employed:

During my two-month internship, my main objective was to leverage Vision-Language Models (VLMs) [1] for image annotation in shop analytics. To achieve this, I combined state-of-the-art deep learning models for object detection, specifically YOLO [4], with advanced VLMs to analyze visual content and generate semantic descriptions. This integration allowed for automated extraction of meaningful attributes from images, such as person identification, demographic traits, and contextual information, producing both annotated images and structured JSON outputs for downstream analytics. This approach

demonstrates the power of combining computer vision and natural language understanding to enable scalable and intelligent image annotation for retail environments.

3.3.1.Object Detection-YOLO:

Deep learning[2] is a subset of machine learning that uses multilayered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain. Some form of deep learning powers most of the artificial intelligence (AI) applications in our lives today.

One of the most important tasks in computer vision is object detection, which involves locating and identifying items in an image or video. In contrast to image classification, which gives an image a single label, object detection gives each detected object its spatial coordinates (bounding boxes) along with its class label, enabling analysis of visual data at a more detailed level.

YOLO[4], which stands for You Only Look Once, is a family of real-time object detection models. Object detection is a computer vision task that uses neural networks to localize and classify objects in images. CNNs (Convolutional Neural Networks) form the base of any YOLO model, which is why YOLO is considered a deep learning model.

Image localization is the process of identifying the correct location of one or multiple objects using bounding boxes, which correspond to rectangular shapes around the objects. This process is sometimes confused with image classification or image recognition, which aims to predict the class of an image or an object within an image into one of the categories or classes.

YOLO was chosen for this project because of its ability to perform single-shot detection, providing both speed and accuracy in real-time applications, and because pre-trained models are widely available, facilitating implementation and fine-tuning.

The illustration below corresponds to the visual representation of the previous explanation. The object detected within the image is a “Person.”

Figure 2 : How YOLO works

3.3.2.Vision Language Models (VLMs):

Vision Language Models (VLMs)[1] are AI models that fill the gap between computer vision and natural language processing (NLP). These models are designed to understand and generate language based on visual inputs which helps them to perform a range of tasks such as describing images, answering questions about them and even creating images from textual descriptions because they are trained on large datasets which are pairs of images and their textual descriptions. VLMs learn to connect visual features with corresponding language allowing them to "see" and "understand" the world in a way that combines both vision and language.

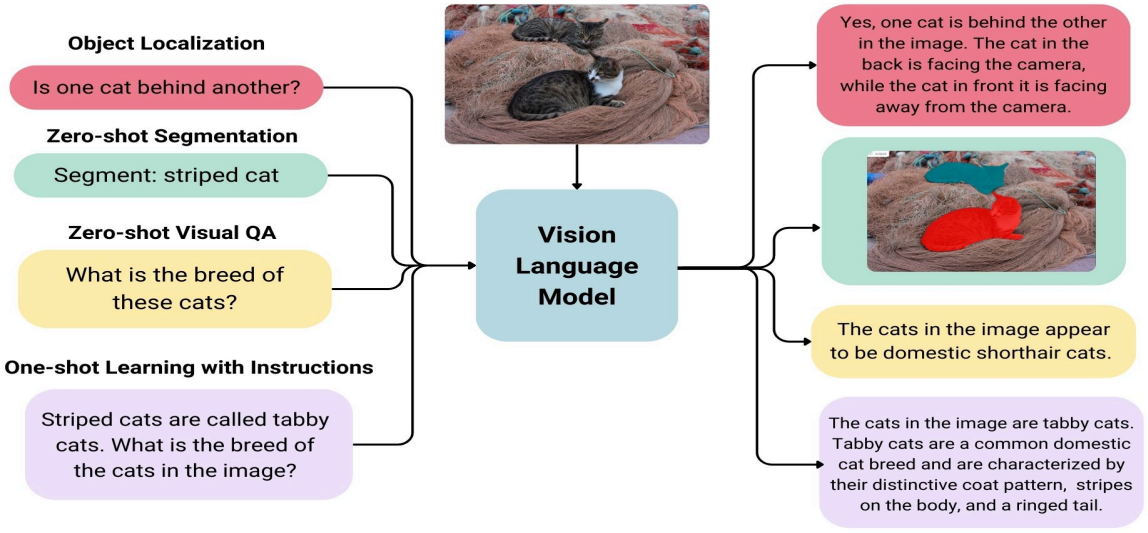

Figure 3 : Capabilities of a Vision Language Model

There are many types of VLMs[1] depending on how they handle the interaction between images and text:

1. Vision-to-Text Models:

Vision-to-text models focus on generating textual descriptions or answering questions based on visual inputs. Like:

- **Image Captioning:** The model generates natural language descriptions of an image. It processes visual features to produce relevant text that describes the scene.
- **Visual Question Answering (VQA):** These models take an image and a question about that image as input and provide a text-based answer.

2. Text-to-Vision Models:

Text-to-vision models generates images from textual descriptions. These models translate natural language input into visual representations which can be used for various creative and practical applications. Some key applications include:

- **Text-to-Image Generation:** These models take a text description and generate an image based on it. For example, given the prompt "A sunset over the ocean" the model will generate an image of a sunset scene.
- **Text-Driven Image Manipulation:** These models modify existing images based on text instruction such as changing the background to a sunset or adjusting colors.

3. Cross-Modal Retrieval Models

They are designed for tasks where one type of data like text or images is used to search for data in the other datatype. These models allow users to perform tasks such as:

- **Image Search Using Text:** This allows users to search for images based on textual queries. For example, entering "a mountain view" into a search engine could retrieve images of mountains.
- **Text Search Using Images:** These models take an image as input and retrieve relevant text such as descriptions or articles about the object in the image.

4.Internship Report:

This internship was carried out at Anavid over a period of two months, from June 30, 2025, to August 29 , 2025.The following figure illustrates the progress of the project throughout the internship period using a Gantt Chart.

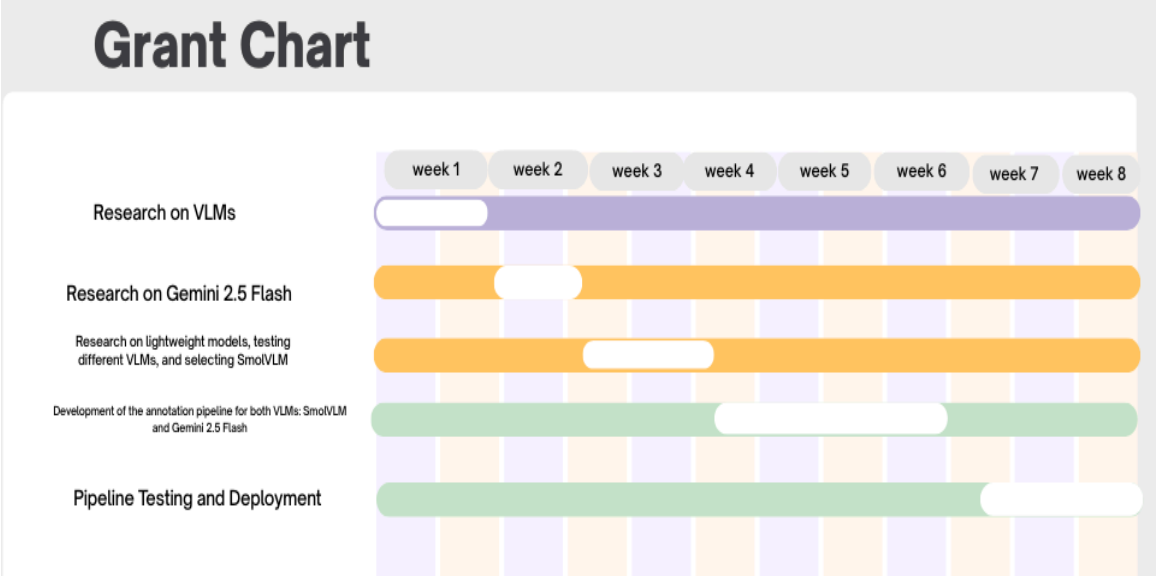

Figure 4 : Gantt chart

5.Accomplished Work:

5.1.Methodology:

In this section, I describe the methodology adopted for the project, including the approaches, tools, and procedures used to achieve the objectives.

During this project ,I proposed a pipeline of image annotation combining YOLOV8+VLMs[1] .For VLMs , I chose to work with Gemini 2.5 Flash Then, I conducted a research into lightweight VLMs to enable the deployment of annotation models on local servers, reducing dependency on external services. Since Gemini 2.5 Flash is free via the API key provided by Google, it is, however, limited to a certain number of uses per day which lead to choosing SMOLVLM as a lightweight model.

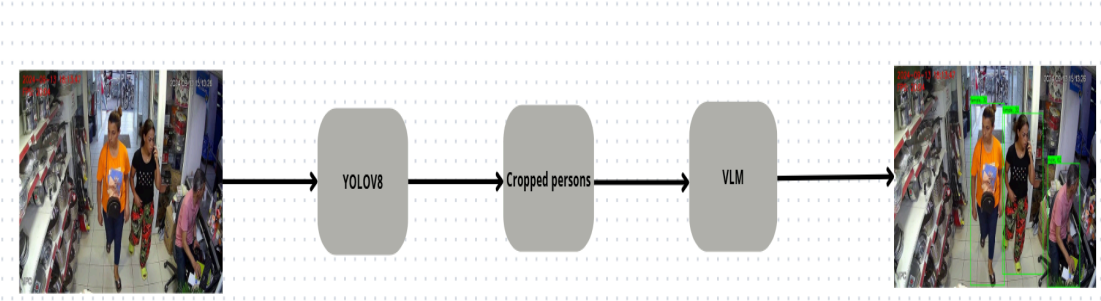

Figure 5 : Suggested Pipeline

In the next section , I will provide a detailed examination of the different model architectures.

5.1.1.YOLOV8:

YOLO [4] architecture is similar to GoogleNet. As illustrated below, it has 24 convolutional layers, four max-pooling layers, and two fully connected layers.

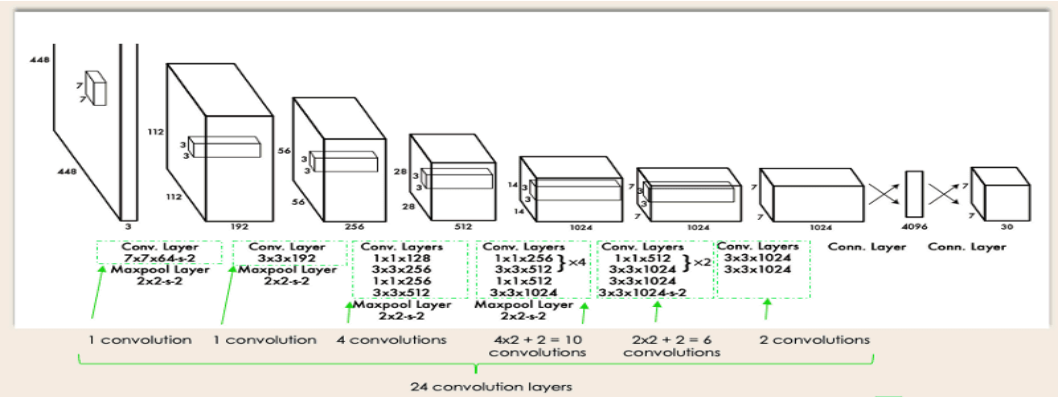

Figure 6 : Architecture of YOLO

YOLOv8[5] was released by Ultralytics on January 10th, 2023, offering cutting-edge performance in terms of accuracy and speed.

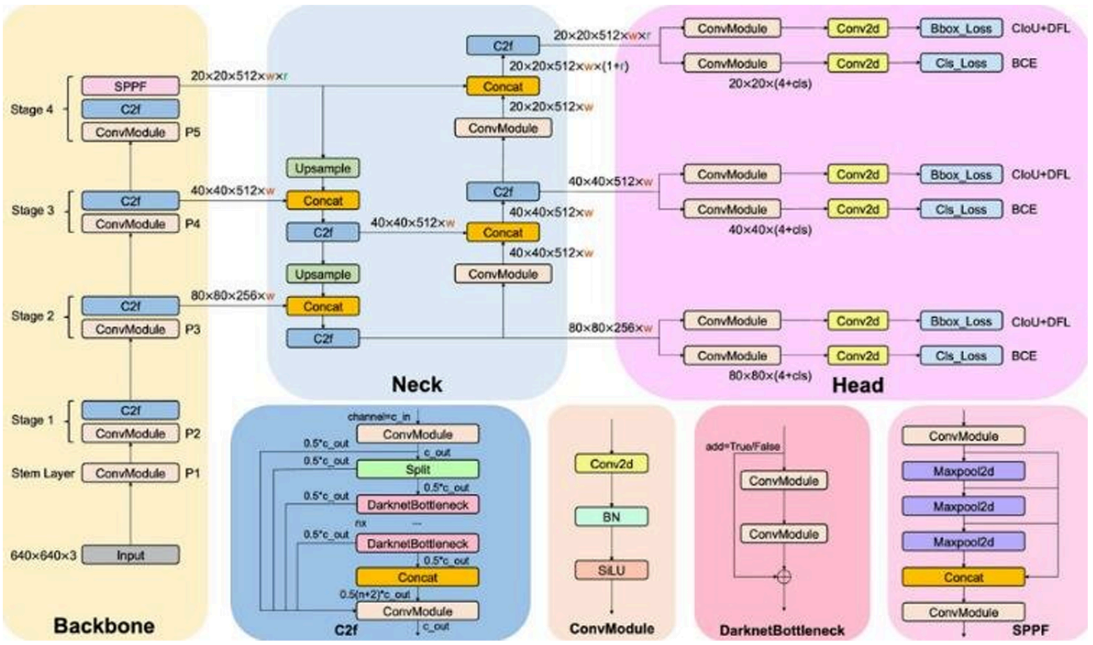

Figure 7 : Architecture of YOLOv8 Source :yolov8-architecture-kamal-sai-tillari

So , let's describe YOLOV8[5] architecture. It consists of several key components that process the input image step-by-step.

A **ConvModule** typically consists of the following layers in sequence:

1. Convolution Layer (Conv2d):

- Applies a 2D convolution operation on the input tensor.
- This operation extracts spatial features by sliding a kernel (filter) over the input tensor.
- Key parameters include Kernel size, which determines the receptive field (e.g., 1x1, 3x3).Stride: Determines the step size for sliding the kernel (e.g., 1 or 2).Padding: Ensures the output dimensions align with requirements.

2. Batch Normalization (BN):

- Normalizes the output of the convolution layer to stabilize training.

- Reduces internal covariate shift and accelerates convergence.
- Ensures each feature map has zero mean and unit variance.

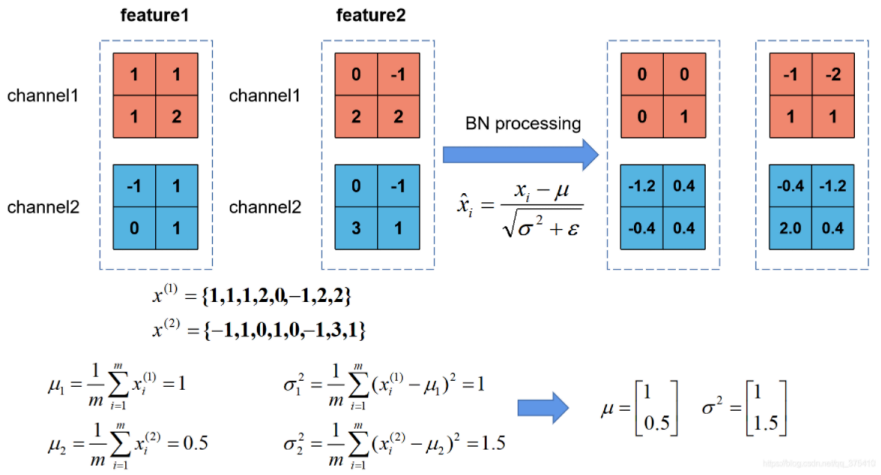

Figure 8 : Batch Normalization

3. Activation Function (SiLU):

- Applies the Sigmoid Linear Unit (SiLU) activation function, also known as Swish.
- Introduces non-linearity to the model, helping it learn complex patterns.
- SiLU is chosen because it is smooth, differentiable, and empirically effective for deep networks.

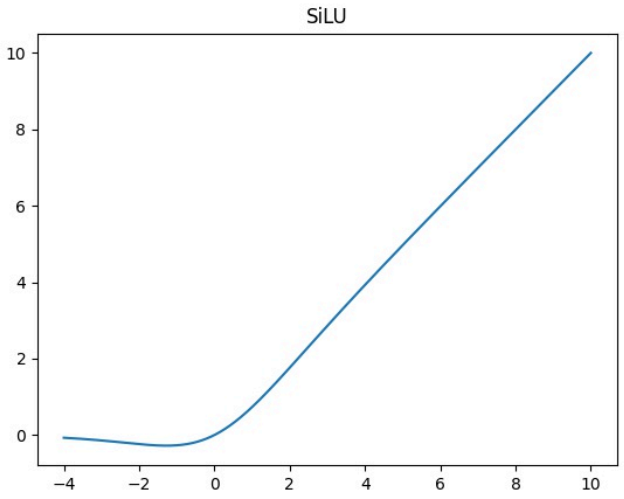

Figure 9 : SiLu Function

The **DarknetBottleneck** module typically contains the following components:

1. Input: The input feature map is passed to both: A main branch for feature transformation. A skip connection to preserve the original features.
2. Main Branch: The main branch applies two convolutional layers with a residual connection: 1x1 Convolution: Reduces the number of channels (dimensionality reduction). This step is computationally efficient and prepares the features for further processing. 3x3 Convolution: Applies spatial filtering to extract meaningful patterns. The number of output channels matches the input to preserve dimensions.
3. Skip Connection: The input feature map is directly added to the output of the main branch after processing. This is achieved through element-wise addition, which enables the model to learn residual features effectively.
4. Activation: Each convolutional layer is followed by Batch Normalization (BN) and the SiLU activation function. SiLU introduces non-linearity, enabling the module to capture complex relationships in the data.

The **C2F module** consists of the following components:

Input Splitting:

- The input tensor is split into two parts: One part (50%) is passed through the main branch, which contains multiple DarknetBottleneck blocks. The other part (50%) is directly forwarded to the output without modification (this acts as a residual connection).

DarknetBottleneck Blocks:

- The main branch contains a sequence of DarknetBottleneck blocks, which are lightweight blocks designed to extract and refine features.
- Each DarknetBottleneck consists of: A 1x1 convolution to reduce feature dimensions. A 3x3 convolution to process the features. A skip connection that adds the input back to the processed features.

Concatenation:

- The processed features from the DarknetBottleneck blocks are concatenated with the unprocessed input (from the residual connection).

Output Convolution:

- The concatenated features are passed through a final convolutional layer (ConvModule) to produce the output feature map with the desired channel depth.

The **SPPF module** consists of the following components:

1. Input: A feature map is taken as input, typically from the deepest layer of the backbone (Stage 4 in YOLOv8).

2. Convolution (ConvModule): Applies a ConvModule (Convolution + Batch Normalization + SiLU activation) to process the input feature map and reduce its channel dimensions, preparing it for pyramid pooling.
3. Max Pooling (3x3 Kernel, Stride 1): Applies three consecutive max-pooling operations with: A 3x3 kernel size. A stride of 1. These pooling layers operate on the same input feature map, creating feature maps with slightly different receptive fields.
4. Concatenation: The outputs of the three max-pooling layers are concatenated along the channel dimension with the original feature map. This combination aggregates spatial information at multiple scales.
5. Final Convolution (ConvModule): A final ConvModule refines the concatenated features, ensuring the output dimensions are as required for the next layers.

The BackBone module consists of the following components**:**

Stem Layer:

- Purpose: Extract initial low-level features and reduce spatial dimensions.
- Input Dimension: 640×640×3 (RGB image).
- Output Dimension: 320×320×64(feature map 1)

Stage 1:

- Purpose: Extract shallow features and further reduce spatial dimensions.
- Input Dimension: 320×320×64
- Output Dimension: 160×160×128(feature map 2)

Stage 2:

- Purpose: Capture mid-level features with more abstraction.
- Input Dimension: 160×160×128

- Output Dimension: 80×80×256(feature map 3)

Stage 3:

- Purpose: Learn higher-level features (e.g., object contours and boundaries).
- Input Dimension: 80×80×256
- Output Dimension: 40×40×512(feature map 4)

Stage 4:

- Purpose: Extract the most abstract and global-level features.
- Input Dimension: 40×40×512
- Output Dimension: 20×20×512(feature map 5)

The Neck Module aggregates features from different stages of the Backbone using feature fusion techniques (e.g., upsampling and concatenation). This helps combine fine-grained details from shallow layers and abstract information from deeper layers, improving the model's ability to detect objects of varying scales.

Input Features:

- From the Backbone: P3 (80×80×256),P4 (40×40×512) and P5(20×20×512)

Upsample and Concatenate:

- Step 1: Upsample P5 (20×20×512) to 40×40×512 and concatenate it with P4 (40×40×512)
- Output: 40×40×512
- Step 2: Apply a convolutional transformation and upsample to 80×80×512, then concatenate it with P3 (80×80×256).

- Output: 80×80×256

Downsample and Concatenate:

- Step 3: Downsample the 80×80×256output back to 40×40×512, and concatenate it with the intermediate 40×40 feature map.
- Output: 40×40×512
- Step 4: Downsample further to 20×20×512, and concatenate it with the intermediate 20×20 feature map.
- Output: 20×20×512.

Finally, **The Head Module** is responsible for generating the final predictions for object detection. It processes the multi-scale feature maps from the Neck to predict bounding boxes, object confidence scores, and class probabilities.

Key Features of YOLOv8

- **Advanced Backbone and Neck Architectures:** YOLOv8 employs state-of-the-art backbone and neck architectures, resulting in improved feature extraction and object detection performance.
- **Anchor-free Split Ultralytics Head:** YOLOv8 adopts an anchor-free split Ultralytics head, which contributes to better accuracy and a more efficient detection process compared to anchor-based approaches.
- **Optimized Accuracy-Speed Tradeoff:** With a focus on maintaining an optimal balance between accuracy and speed, YOLOv8 is suitable for real-time object detection tasks in diverse application areas.

So I chose to work with YOLOV8 because it introduces a more modular and flexible design, allowing easier customization and fine-tuning. Built-in support for various tasks beyond object detection, such as segmentation and pose estimation. In addition YOLOV8 presents multiple Lightweight versions of the models so I used YOLOv8n which is the

smallest version which runs extremely fast for real-time detection on edge devices, with lower accuracy compared to larger YOLOv8 models, making it ideal when speed and low memory usage are more important than maximum precision especially that our pipeline might be implemented on a security camera .

5.1.2.GEMINI 2.5 Flash :

Gemini 2.5 Flash[6] is the next iteration in the Gemini 2.0 series of models, a suite of highly-capable, natively multimodal, reasoning models. Gemini 2.5 Flash is Google's first fully hybrid reasoning model. It is a fast and efficient Vision-Language Model by Google, built for real-time image and text understanding. Compared to earlier versions, it offers faster response times, better visual reasoning, and supports longer context—making it ideal for advanced AI agents and interactive applications.

Gemini 2.5 Flash takes as inputs text strings (e.g., a question, a prompt, document(s) to be summarized), images, audio, and video files, with a 1M token context window and gives as outputs: Text, with a 64K token output.

Gemini 2.5 Flash is well suited for applications that require:

- cost-efficient thinking.
- well-rounded capabilities.

Capability **Benchmark**[2]		**Gemini 2.5 Flash GA**	**Gemini 2.5 Flash Preview** (04-17) Thinking	**Gemini 2.0 Flash** Non-thinking	**OpenAI o4-mini**	**Claude 3.7 Sonnet** 64k Ext. thinking	**Grok 3 Beta** Ext. thinking	**DeepSeek R1**
Reasoning & knowledge **Humanity's Last Exam (no tools)**		11.0%	12.1%	5.1%	14.3%	8.9%	—	8.6%*
Science **GPQA diamond**	single attempt (pass@1)	82.8%	78.3%	60.1%	81.4%	78.2%	80.2%	71.5%
	multiple attempts	—	—	—	—	84.8%	84.6%	—
Mathematics **AIME 2025**	single attempt (pass@1)	72.0%	78.0%	27.5%	92.7%	49.5%	77.3%	70.0%
	multiple attempts	—	—	—	—	—	93.3%	—
Code generation **LiveCodeBench v5**	single attempt (pass@1)	63.9%	63.5%	34.5%	—	—	70.6%	64.3%
	multiple attempts	—	—	—	—	—	79.4%	—
Code editing **Aider Polyglot**		61.9% / 56.7% whole / diff	51.1% / 44.2% whole / diff	22.2% whole	68.9% / 58.2% whole / diff	64.9% diff	53.3% diff	56.9% diff
Agentic Coding **SWE-Bench Verified**		60.4%	—	—	68.1%	70.3%	—	49.2%
Factuality **SimpleQA**		26.9%	29.7%	29.9%	—	—	43.6%	30.1%
Factuality **FACTS Grounding**		85.3%	—	84.6%	62.1%	78.8%	74.8%	56.8%
Visual reasoning **MMMU**	single attempt (pass@1)	79.7%	76.7%	71.7%	81.6%	75.0%	76.0%	no MM support
	multiple attempts	—	—	—	—	—	78.0%	no MM support
Image understanding **Vibe-Eval (Reka)**		65.4%	62.0%	56.4%	—	—	—	no MM support
Long context **MRCR v2**	128k (average)	74%	—	36%	49%	—	54%	45%
	1M (pointwise)	32%	—	6%	—	—	—	—
Multilingual performance **Global MMLU (Lite)**		88.4%	88.4%	83.4%	—	—	—	—

Figure 10 : Comparison between different models

As you can see from the image above , Gemini 2.5 Flash demonstrated strong performance across a range of models especially in the task of Visual reasoning and Image understanding which are essential for our data annotation .

5.1.3.SMOLVLM2:

SmolVLM2[7], developed by engineers on the Hugging Face TB Research team, is part of the "Smol Models" initiative. This initiative is focused on making "efficient and lightweight AI models that can run effectively on-device while maintaining strong performance." SmolVLM2 is capable of both image and video understanding.

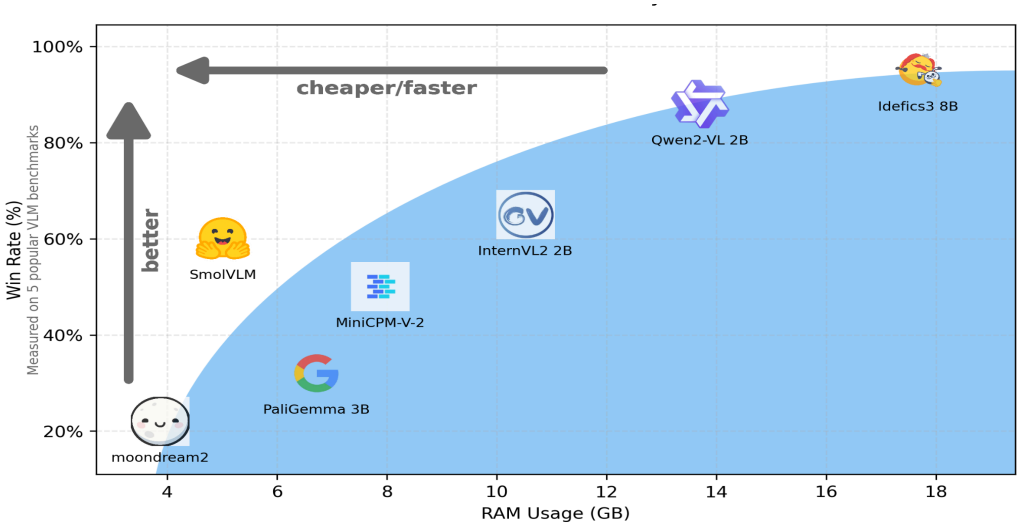

Figure 11 : SMOL Vision Model Ecosystem

SmolVLM2[7] comes in three sizes: 256M, 500M, and 2.2B. The larger the model, the better performance you can expect on your tasks.

SmolVLM2[7] was tested on various multimodal tasks. Here were our findings in comparison to other multimodal models available at the time of the model's release.Based on the figure, we can observe that SmolVLM2[7] performs very well across multiple tasks, ranking just below GPT-4.5 in accuracy. Considering its strong performance, we chose SmolVLM2[7] for our project because it is a lightweight model, does not require an API key, and is freely available, making it practical for local use without additional infrastructure or cost.

Question	SmolVLM2	PaliGemma	Moondream 2b	GPT-4.5
How many coins do I have? (VQA)				
Which movie is this scene from? (VQA)				
Read text from the picture (Document OCR)				
What is the price of Pastrami Pizza? (Document VQA)				
How much tax did I pay? (Document VQA)				
Find the dog. (Zero-Shot Object Detection)				
Read the serial number. (OCR)				

Figure 12 : comparison with other multimodal models

Architecture:

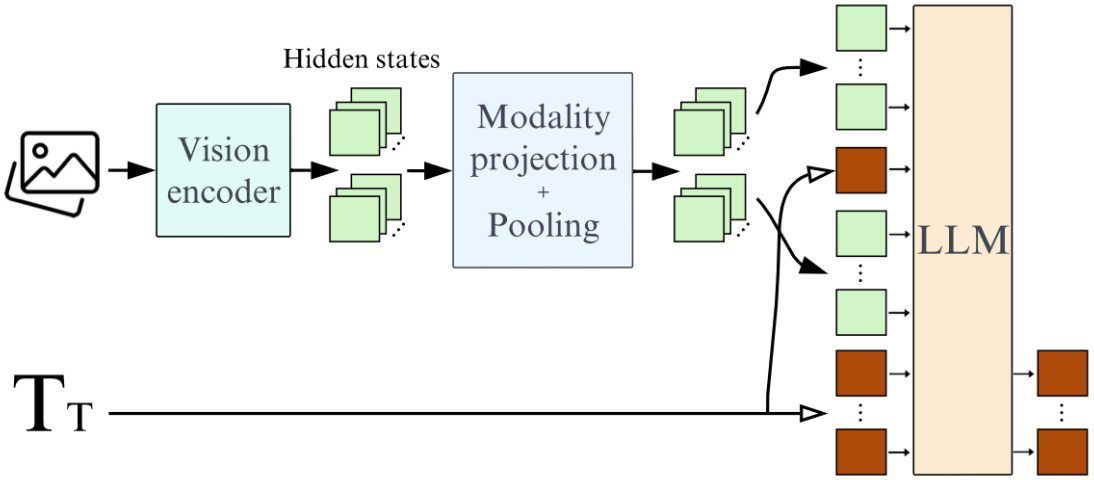

Figure 13 : Architecture of SMOLVLM

For SmolVLM, the architecture closely follows the architecture from Idefics3, to the point that it uses the same implementation in transformers. There are, however a few key differences:

- Llama 3.1 8B was replaced with SmolLM2 1.7B as the language backbone.
- It more aggressively compresses the patched visual information by reducing the information by 9x using the pixel shuffle strategy, compared to 4x with idefics3.
- it uses patches of 384*384, instead of 364x364, because 384 is divisible by 3, which is necessary for our pixel shuffle strategy to work.
- For this, it changes the vision backbone to use shape-optimized SigLIP with patches of 384x384 pixels and inner patches of 14x14.

For the data annotation , I choose to work with smolvlm2 2.2B[8] which is a lightweight multimodal model designed to analyze video content. The model processes videos, images, and text inputs to generate text outputs - whether answering questions about media files, comparing visual content, or transcribing text from images. Despite its

compact size, requiring only 5.2GB of GPU RAM for video inference, it delivers robust performance on complex multimodal tasks. This efficiency makes it particularly well-suited for on-device applications where computational resources may be limited.

6.Results:

Thanks to the developed pipeline, we can now efficiently annotate image data using the combined YOLOv8 [5]+ Vision-Language Model (VLM)[1] architecture. Since our images are sourced from a retail store surveillance camera, we designed a dynamic Region of Interest (ROI) filtering mechanism. This component allows either a static ROI or a dynamic filter that removes small or distant objects based on their median height (with a configurable ratio threshold), ensuring that the detection focuses on relevant individuals in the scene.

YOLOv8 [5]is responsible for detecting persons and extracting cropped regions corresponding to them. The model demonstrates high accuracy in identifying people, even in crowded or cluttered environments, and achieves fast inference times relative to other detection models, making it suitable for both CPU and GPU deployments. These cropped images are then processed by two VLMs[1] — Gemini VLM [6]and SmolVLM[7]. Both models were selected for their lightweight architectures and CPU-efficient performance, ensuring that the pipeline remains fast and practical even on standard hardware.

To maximize annotation quality, each model in the configuration file has its own dedicated prompt, which was carefully fixed after extensive experimentation. Through iterative testing, we refined these prompts to guide the models in generating consistent, structured, and relevant descriptions of physical attributes.

As a result, the pipeline produces two separate annotation files — one for Gemini[6] outputs and another for SmolVLM[7] outputs. Additionally, each detected person is

associated with its own JSON file containing the attributes generated by each VLM, along with a consolidated file aggregating all results. This modular design allows easy verification, comparison, and further processing of annotations, highlighting the synergy between fast detection (YOLOv8)[5], efficient multi-modal understanding (VLMs)[1], and carefully optimized prompt engineering.

An example of the annotated images is provided below:

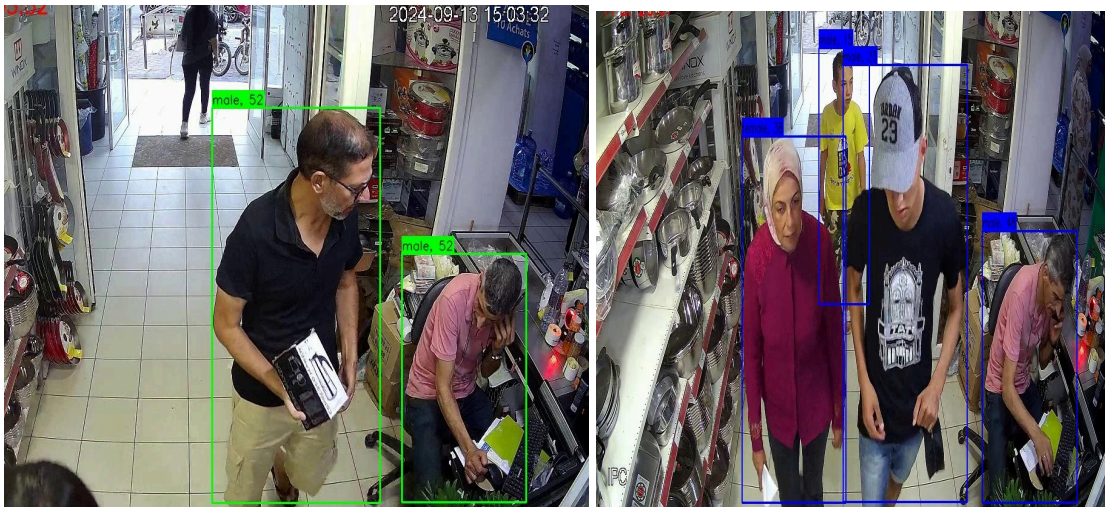

Figure 14 : Example of annotated images

An example of the JSON structure is provided below: it contains facial analysis outputs generated by Vision-Language Models (VLMs)[1] for each detected individual. The outputs include predicted gender ("male" or "female"), age group (from predefined ranges), exact age (integer), and key attributes such as glasses, cap, headscarf, mask, hood, occlusion, facing direction, and seasonal clothing ("summer" or "winter"). Some models also provide reasoning, a short phrase justifying the age and gender classification. These structured outputs enable automated annotation, cross-model comparison, and evaluation of model accuracy in facial analysis applications.

Figure 15: Example of JSON output file

Beyond these per-image JSON files, the pipeline also generates two complementary log files:

- **processing.log**: keeps track of the execution flow, useful for debugging (e.g., if the pipeline crashes or the Gemini API quota is exceeded).

- **progress.json**: stores the last image successfully processed by GeminiVLM. This mechanism enables the pipeline to **resume execution from the last checkpoint** instead of restarting from scratch, ensuring robustness in long annotation tasks.

7.Consolidation of Knowledge:

Table 1: Comparison of Competencies and Skills Acquired

Skills acquired at INSAT	Skills acquired during the internship
• Problem Analysis • Project Management • Computer Vision	• Working withVision Language Models (VLMs) • Coding Best Practices • Research

8.Conclusion :

My internship at Anavid was an excellent opportunity that allowed me to acquire essential skills in artificial intelligence and practical experience in deploying AI solutions. During my time at the company, I worked on an innovative project: building an automated image annotation pipeline by combining YOLOv8[5] for fast and accurate person detection with Vision-Language Models (Gemini VLM [6]and SmolVLM[7]) for rich, multi-modal understanding. I designed a dynamic Region of Interest (ROI) filtering mechanism to focus detection on relevant areas of the scene, and each VLM[1] was carefully configured

with dedicated prompts that were refined through extensive experimentation to ensure consistent and structured outputs.

The pipeline processes surveillance images end-to-end: YOLOv8[5] first detects individuals and extracts cropped regions, which are then analyzed by the VLMs[1] to generate detailed attributes and context descriptions. The results are stored in both individual JSON files per detected person and consolidated annotation files, enabling easy verification and further analysis. Through this project, I successfully integrated multiple AI technologies, optimized them for CPU efficiency, and developed a modular, scalable solution that produces reliable and high-quality annotations, demonstrating the potential of combining detection and multi-modal understanding in practical applications.

This internship gave me the chance to apply the knowledge I had gained during my studies while also developing additional skills. I had the opportunity to collaborate with talented professionals who guided me throughout the project and shared their expertise.

8.1 – Strengths:

Improving productivity:

With the system I developed, it is possible to automate image annotation for large datasets, significantly reducing manual work and accelerating AI model training, which improves workflow efficiency and productivity.

Acquisition of advanced technical skills:

The internship allowed me to acquire advanced technical skills in artificial intelligence, specifically in Deep Learning, by using transfer learning to ensure the efficiency of the automated image annotation pipeline I developed using Vision-Language Models, the Gemini API, and YOLO.

8.2 –Limitations:

Gemini API limitations:

During the project, the usage quota of the Gemini API was a limiting factor. Although

we worked with local lightweight Vision-Language Models such as SmolVLM to address this limitation, Gemini still proved to be faster and produced higher-quality annotations. This restricted the number of images that could be annotated automatically at a time, slowing down the overall process.

Potential improvements:

To overcome this limitation and improve annotation quality, it would be beneficial to integrate the VGG tool for manual corrections and validation. This approach would allow for more accurate annotations while complementing the automated pipeline and reducing dependency on the API quota. Additionally, future developments in tiny VLMs and ultra-lightweight local VLMs, which are currently being researched, could further enhance the pipeline by enabling faster, on-device annotation with minimal resource usage, providing a scalable and cost-effective solution.

Bibliography:

[1]VLMs:Vision Language Models retrieved from https://huggingface.co/blog/vlms, https://www.geeksforgeeks.org/artificial-intelligence/vision-language-models-vlms-explained,

https://www.ibm.com/think/topics/vision-language-models

and https://huggingface.co/blog/vlms-2025 accessed on 25/09/2025.

[2] Deep learning retrieved from https://www.ibm.com/think/topics/deep-learning accessed on 25/09/2025.

[3] VS code: Visual Studio code retrieved from https://code.visualstudio.com/download accessed on 25/09/2025.

[4] Yolo : You Only Look Once retrieved from https://www.datacamp.com/blog/yolo-object-detection-explained ,https://www.geeksforgeeks.org/machine-learning/yolo-you-only-look-once-real-time-object-detection/ and https://viso.ai/computer-vision/yolo-explained/ accessed on 27/09/2025.

[5]YOLOV8 retrieved from https://docs.ultralytics.com/fr/models/yolov8/#performance-metrics and https://www.geeksforgeeks.org/machine-learning/object-detection-using-yolov8/ accessed on 27/09/2025.

[6] Gemini 2.5 Flash retrieved from https://deepmind.google/models/gemini/flash/ accessed on 28/09/2025.

[7] Smolvlm2 retrieved from https://huggingface.co/blog/smolvlm#Architecture and https://roboflow.com/model/smolvlm2 accessed on 28/09/2025.

[8] smolvlm2 2.2B retrieved from https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B accessed on 28/09/2025.