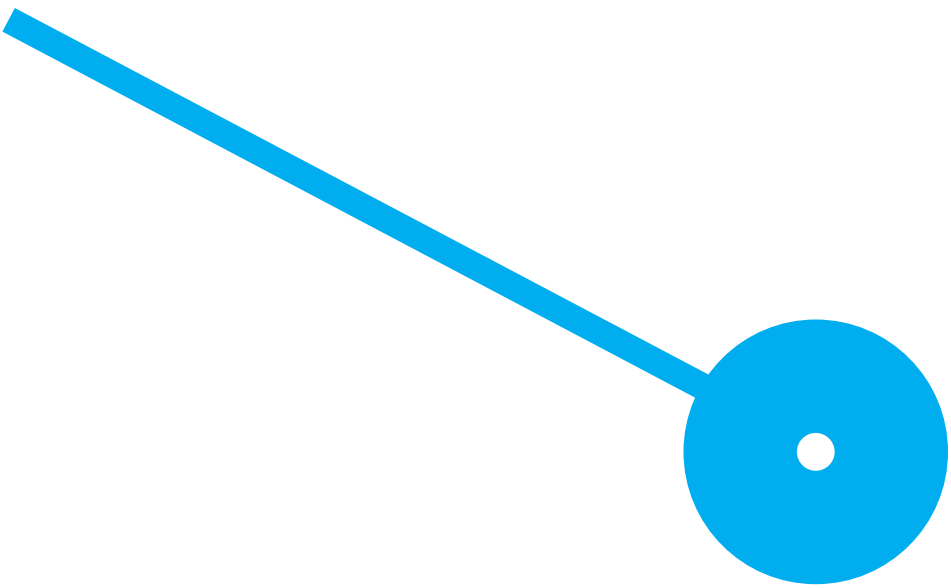
SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

MASTER
Engenharia Informática

Multimedia data extraction and analysis tool: focus on video and image processing

João Miguel Teixeira Bragança

DECEMBER/2024



SCHOOL OF
MANAGEMENT AND
TECNOLOGY
POLYTECHNIC OF
PORTO

MASTER
Engenharia Informática

Multimedia data extraction and analysis tool: focus on video and image processing

João Miguel Teixeira Bragança
8190555

Advisor(s)

PhD, Fábio André Souto Da Silva

Dissertation submitted in fulfilment of the requirements for the Master's degree in Engenharia Informática in the School of Management and Technology of the Polytechnic of Porto.

DECEMBER/2024

Integrity Statement

I, **João Miguel Teixeira Bragança**, student nº **8190555**, of the Master's Degree in **Engenharia Informática** of the School of Management and Technology of the Polytechnic of Porto, declare that I have not plagiarized or self-plagiarized, therefore the work entitled "**Multimedia data extraction and analysis tool: focus on video and image processing**" is original and of my own authorship, not having been used previously for any other purpose. I further declare that all sources used are cited, in the text and in the final bibliography, according to the referencing rules adopted in the institution.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Fábio, for his guidance, patience, and support throughout this journey. Your wisdom and availability were fundamental to the success of this work.

Next, I am deeply grateful to my family, especially my father, mother, and brothers, José and Luís, for all their patience and unconditional support throughout this journey. Your constant encouragement was crucial in helping me face the challenges of this journey.

My heartfelt thanks go to my wonderful girlfriend, Ana, for being by my side throughout this important phase. Your sharp eye and thoughtful proofreading feedback on this document were invaluable, and your support gave me the strength to keep going, even through the toughest moments. I'm truly grateful for everything.

To my friends, I offer my sincere thanks for your companionship and for being a shoulder to lean on in times of despair. Without you, this journey would have been much harder.

I would also like to thank my work colleagues at DEUS for their help, advice, and for contributing to my growth both professionally and personally during this phase.

Finally, I would like to thank myself for the dedication and commitment I maintained, even as a working student. In moments of discouragement, I never gave up, and that perseverance allowed me to achieve this goal.

Abstract

In today's digital landscape, the rapid growth of multimedia content, particularly from influencers, has created a critical need for advanced monitoring tools. Building on previous research in multimedia data analysis, this dissertation proposes the development of a tool for extracting and analysing multimedia data to detect violations in influencer-produced content. The tool leverages pre-trained models such as Whisper.AI for speech recognition, YOLOv8 for object detection, and EasyOCR for Optical Character Recognition (OCR). Additionally, sentiment analysis models are employed and tested, with YOLOv8 further trained for specific tasks such as logo detection, ensuring adaptability to various use cases.

The objective of this dissertation is to design a versatile and customisable tool capable of performing precise content analysis, including object detection, speech transcription, OCR, sentiment analysis, image classification and logo detection. The solution will be cloud-based, ensuring scalability and the ability to process large datasets.

Keywords: Multimedia data analysis, Object detection, Speech recognition, Sentiment analysis, Content monitoring, Logo detection, Image Classification, Optical Character Recognition

Resumo

No panorama digital atual, o rápido crescimento de conteúdos multimédia, particularmente de *influencers*, criou uma necessidade crítica de ferramentas avançadas de monitorização. Com base em investigações anteriores na análise de dados multimédia, esta dissertação propõe o desenvolvimento de uma ferramenta para extrair e analisar dados multimédia, com o objetivo de detetar violações em conteúdos produzidos por *influencers*. A ferramenta utiliza modelos pré-treinados, como o Whisper.AI para reconhecimento de fala, o YOLOv8 para deteção de objetos e o EasyOCR para extração de texto. Adicionalmente, são utilizados e testados modelos de análise de sentimentos, sendo o YOLOv8 ainda treinado para tarefas específicas, como a deteção de logótipos, assegurando a adaptabilidade a vários casos de uso.

O objetivo principal desta dissertação é conceber uma ferramenta versátil e personalizável, capaz de realizar análises de conteúdo precisas, incluindo deteção de objetos, transcrição de fala, reconhecimento ótico de caracteres, análise de sentimentos, classificação de imagens e deteção de logótipos. A solução será baseada na *cloud*, garantindo escalabilidade e a capacidade de processar grandes volumes de dados.

Keywords: Análise de dados multimedia, Deteção de objetos, Reconhecimento de fala, Análise de sentimentos, Monitorização de conteúdo, Deteção de logotipos, Classificação de imagens, Reconhecimento ótico de caracteres

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Context and Motivation

In recent years, the exponential growth of multimedia content generated on digital platforms, particularly within the influencer community, has created an urgent demand for advanced tools capable of monitoring and analysing such material. Governmental and non-governmental organisations, alongside regulators, advertisers, market researchers, and influencer management bureaus, are increasingly focused on scrutinising influencer content and advertisements that appear online. Platforms such as YouTube, Instagram, and TikTok, along with brands and companies, face significant challenges in ensuring compliance with regulations, as they work to detect potential infringements such as the use of unauthorised logos, inappropriate content, or misleading advertisements. According to data from Statista (Figure 1.1), the total volume of data created and replicated globally has surged dramatically, growing from just 2 zettabytes in 2010 to 64.2 zettabytes in 2020. This upward trend shows no sign of slowing, with estimates suggesting it will reach a staggering 181 zettabytes by 2025. To put these figures in perspective, a single zettabyte is equivalent to 1,024 exabytes, or approximately one billion terabytes. This explosive growth in data creation reflects the increasingly digital nature of society, where content generation from platforms like YouTube, Instagram, and TikTok continues to accelerate rapidly. As a result, the need for sophisticated tools to monitor and analyse this vast amount of multimedia content has never been greater, especially in sectors like influencer marketing, where compliance and content regulation are paramount.

As more content is consumed on mobile devices, traffic has also seen an exponential rise. According to Statista (Figure 1.2), mobile data traffic surged from 11.51 exabytes per month in 2017 to 77.49 exabytes per month in 2022.

The influencer marketing industry continues to grow rapidly, with the market size expected to reach $24 billion by 2024 (Figure 1.3). This highlights the economic importance of effective content monitoring to ensure regulatory compliance and protect brand integrity.

The sheer volume of content, combined with the complexity of international regulations, necessitates innovative approaches to data extraction and content analysis. Effective monitoring of digital content is crucial for these stakeholders to ensure regulatory compliance and maintain market integrity. This research, informed by Phan et al.'s (2022) exploration

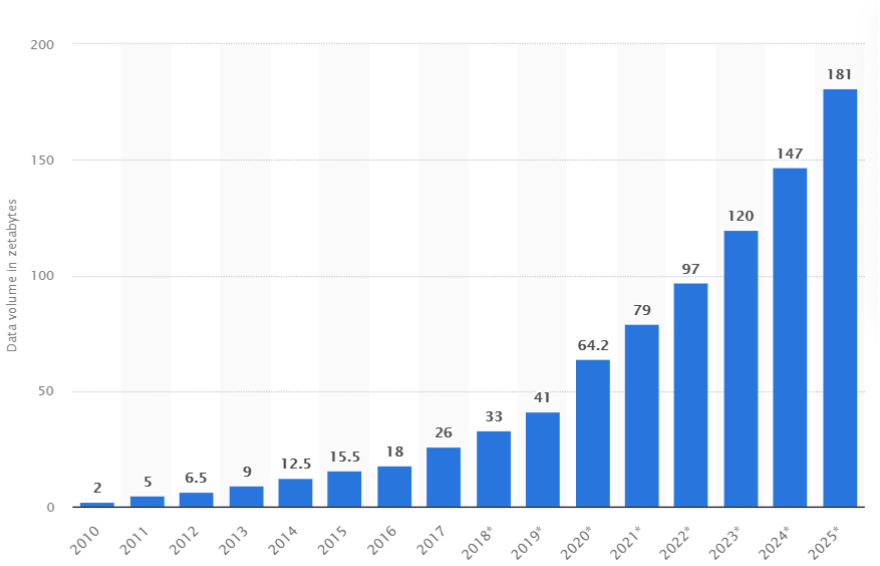


Figure 1.1: Amount of data created, consumed, and stored 2010-2025 [1].

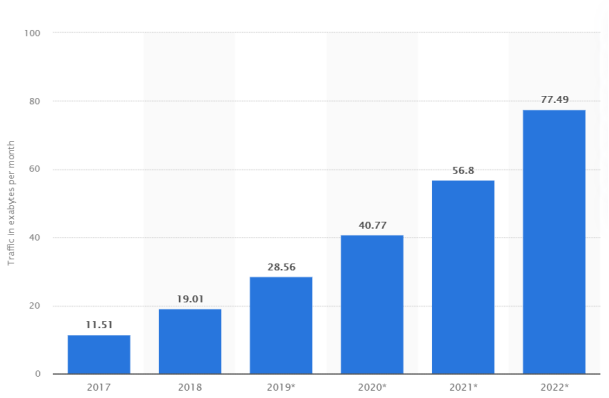


Figure 1.2: Global mobile data traffic 2017-2022 [2].

of content-based video big data retrieval, addresses the growing demand for sophisticated surveillance and analysis of influencer-generated content across platforms [10].

This dissertation builds upon my previous work [11], which explored the use of object detection and speech recognition to identify potential content violations. While that study demonstrated the technical feasibility of extracting significant insights, such as object presence and transcriptions, there remains a need for a more comprehensive solution that can be adapted to various regulatory and market needs.

This research goes beyond mere data extraction, addressing the complex challenges of monitoring multimedia content in the digital age. Effective social media monitoring tools are essential for managing and interpreting the vast amounts of data generated by users daily [12], and sophisticated algorithms and data processing techniques are required to handle this complexity [13]. Previous studies have underscored the importance of information extraction from unstructured data, particularly in identifying sponsored content and other regulatory violations [14].

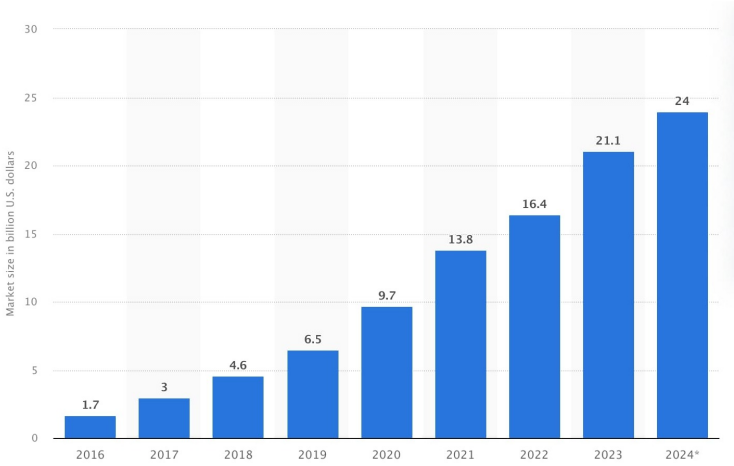


Figure 1.3: Worldwide Influencer Marketing Market Size 2016-2024 [3].

1.2 Objectives

The objective of this dissertation is to develop and demonstrate a tool designed to extract and analyse multimedia content, with a specific focus on video and image processing. This tool will employ a combination of pre-trained and trained models, such as YOLOv8 for object detection and image classification, Whisper.AI for automatic speech recognition, sentiment analysis, EasyOCR for optical character recognition, and YOLO for logo detection [15].

By providing a comprehensive proof of concept, this research aims to showcase the technical feasibility of accurately identifying key elements within content, such as objects, logos, people, text, transcriptions, and sentiment. The tool will assist with regulatory compliance by detecting potential violations, such as unauthorised use of logos or inappropriate content, while also enabling deeper insights for content moderation. This system can further be applied to automatically flag, review, and moderate user-generated content on social media platforms, ensuring that harmful or inappropriate material is identified and addressed swiftly.

One of the primary innovations of this tool is its adaptability: it can be customised to handle specific tasks like logo detection or sentiment analysis based on user requirements. Additionally, its cloud-based infrastructure ensures scalability, enabling the tool to efficiently process large datasets, making it suitable for a range of applications in the ever-growing field of digital content monitoring.

1.3 Research Methodology

This study employs an action research approach, which is ideal for the iterative exploration and validation of the developed technological solution. Action research is characterised by a cyclical process involving planning, acting, observing, and reflecting, as illustrated in Figure 1.4 [16]. This method aligns well with the goal of developing and assessing a tool for multimedia data extraction, integrating pre-existing technologies.

During the planning phase, objectives were established for developing and integrating advanced pre-existing tools for object detection, speech recognition, sentiment analysis, and

OCR. An in-depth review of the latest advancements in these technologies was conducted to ensure the proposed solution aligned with the state of the art and met the requirements for multimedia content monitoring and data extraction [17].

In the action phase, these pre-existing models were applied to the research context, with efforts focused on developing a comprehensive tool that integrates these technologies. Leveraging the robust capabilities of the tools, the system was customised to extract relevant data from video and image content. During this stage, adjustments were made to tailor specific models for precise detection and analysis of content.

The observation phase was crucial for collecting and analysing data on the tool's performance in practical scenarios. Testing was conducted on various multimedia content, evaluating the accuracy of object detection, speech transcription, text recognition, and the effectiveness of sentiment analysis. Performance metrics, such as speed and scalability, were also carefully examined [18].

Finally, the reflection phase enabled analysis of findings, assessing the tool's implementation success and identifying areas for improvement. The iterative nature of action research ensures that insights gained during each phase feed back into the planning and development cycles, allowing for continuous refinement of the tool and its applications (see Figure 1.4) [19]. Future adjustments, such as fine-tuning the models or expanding the system's capabilities, are recognised as necessary steps to further enhance the tool's accuracy and scalability.

Figure 1.4: Action Research methodology steps.

1.4 Document Structure

This dissertation is structured into seven chapters that guide the reader through the context, development, and evaluation of the proposed tool. The Introduction chapter provides an overview of the research, outlining the context and motivation behind the study, particularly the growing need for advanced content monitoring tools in the influencer-driven digital landscape. It also presents the study's objectives, research methodology and bibliography.

The State of the Art chapter reviews key technologies relevant to this research. It covers object detection, speech recognition, sentiment analysis, image classification, OCR and Microservices offering a critical reflection on the current advancements in these fields and highlighting how they relate to multimedia content analysis.

The Related Tools and Studies chapter surveys the existing tools and approaches that tackle similar problems, positioning the tool developed in this research within the broader landscape of current technologies and research in content monitoring and multimedia data extraction.

The System Design and Implementation chapter details the architecture of the developed tool, including the integration of YOLOv8 for object detection, Whisper.AI for speech recognition, sentiment analysis with Cross-lingual Model - Robustly Optimized BERT Approach (XLM-RoBERTa), and EasyOCR for OCR. It also includes a step-by-step breakdown of the YOLOv8 training process. Additionally, the chapter discusses the cloud infrastructure implemented to ensure scalability and efficiency in handling large datasets.

The Performance Evaluation and Results chapter presents the tool's performance results, evaluating the individual components: object detection, speech recognition, and sentiment analysis. The final section of this chapter integrates these components and assesses the tool's performance on real-world multimedia content, particularly influencer videos from different platforms.

The Conclusion chapter summarises the key findings, reflecting on the success and limitations of the developed tool. It also discusses future research directions and potential improvements to enhance the tool's performance and applicability in broader contexts.

Finally, the Bibliography section collates all the references cited across the chapters, serving as a comprehensive resource for further exploration of the topics discussed in this dissertation.

Chapter 2

State of the art

In the realm of Artificial Intelligence (AI), Machine Learning plays a pivotal role, acting as the backbone for significant advancements in various subfields. This technology is crucial for the effective and in-depth analysis of complex data, including videos and associated metadata. Within this context, three specific fields of AI stand out: Computer Vision, Deep Learning, and Natural Language Processing (NLP) [20].

Computer Vision enables machines to interpret visual content, making it essential for information extraction from videos, a task that is becoming increasingly vital across numerous sectors [21]. Deep Learning enhances this by teaching machines to recognise intricate patterns in data, supporting both visual content analysis and intelligent decision-making. Meanwhile, Natural Language Processing allows machines to understand and process human language, which is critical for handling text in metadata and extracting insights from user interactions. These advancements are increasingly applied to content moderation, where AI systems automatically detect harmful or inappropriate content, ensuring a safer digital environment [22].

Together, these AI technologies, powered by Machine Learning (ML), are central to driving innovations, offering sophisticated solutions for data analysis and metadata extraction in our increasingly digital and content-rich era.

2.1 Object Detection

Object Detection is a widely researched area in Computer Vision, focusing on identifying and locating objects within images or videos. The primary challenge lies in accurately detecting various objects in different environments and under varying conditions, such as changes in size, lighting, or occlusion. Numerous models are available today, each with different characteristics and performances, making them more or less suitable depending on the specific problem or use case.

As a critical component of computer vision, object detection enables machines to perceive and interpret the world through images and videos. Its applications span multiple industries, from autonomous driving to healthcare and surveillance, making the accurate detection of objects essential. The evolution of object detection models has seen significant progress, particularly with the advent of deep learning techniques, which have

drastically improved accuracy and performance in detecting and classifying objects under challenging conditions.

2.1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNN) are a class of Deep Learning algorithms widely used for the analysis of visual images. They are especially powerful for tasks involving recognition, classification, and image processing [23]. CNNs consist of convolutional layers that use filters to highlight important features in images, such as edges or patterns, and Rectified Linear Unit (ReLU) layers, which introduce non-linearity into the network. This non-linearity enables the network to learn and represent more complex patterns in the data. Pooling layers reduce the spatial dimensions (i.e., width and height) of the feature maps, condensing the information to focus on the most important features. After processing through the previous layers, the fully connected layer integrates all learned features for tasks like classification or recognition [24]. These components can be arranged in various configurations depending on the specific requirements and objectives of the task at hand. This flexibility allows CNN architetures to be customised and optimised for different challenges in object detection, addressing the unique aspects of each application. As illustrated in Figure 2.1, a typical CNN consists of convolutional layers, ReLU activations, pooling layers, and fully connected layers. These components are crucial for effectively learning and classifying visual patterns.

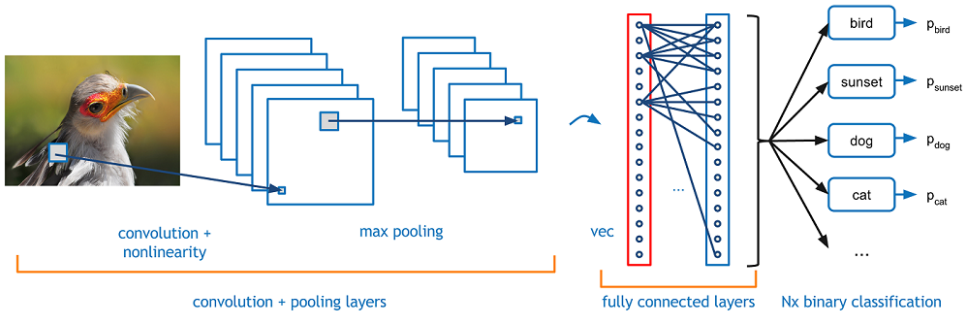


Figure 2.1: A typical CNN architecture showing the convolution, ReLU, pooling layers, and fully connected layers.

2.1.2 R-CNN and Variants

Region-based Convolutional Neural Network (R-CNN), marks the beginning of using CNNs in object detection. It identifies potential object regions in an image and then classifies each region using a CNN. Despite its high accuracy, R-CNN is computationally intensive because it processes each region proposal separately [4]. Improving upon this, Fast R-CNN applies a single CNN to the entire image, creating a convolutional feature map. Region proposals are then classified using this shared map, significantly enhancing efficiency and speed [5].

Figure 2.2 illustrates the steps of the R-CNN model, while Figure 2.3 depicts the Fast R-CNN architeture, from input image to detection output.

Further advancing the field of object detection, the Faster R-CNN model revolutionises the process by incorporating a Region Proposal Network (RPN). This network efficiently

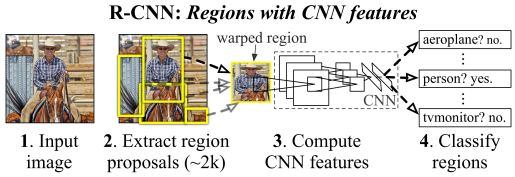


Figure 2.2: R-CNN model steps [4].

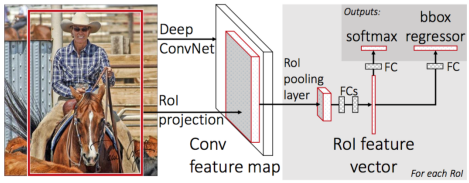


Figure 2.3: Fast R-CNN architecture [5].

generates precise region proposals directly from the convolutional feature maps. The seamless integration of the RPN accelerates the proposal generation phase while maintaining the high accuracy characteristic of the R-CNN family. As a result, Faster R-CNN stands out as a remarkably proficient model for real-time object detection [25].

2.1.3 SSD (Single Shot MultiBox Detector)

The SSD, streamlines the object detection process by eliminating the need for separate region proposals. Instead, it divides the image into a grid and predicts both the presence of objects and their bounding boxes for each grid cell in a single pass, significantly speeding up the detection process [6]. Figure 2.4 illustrates the layers of the SSD model.

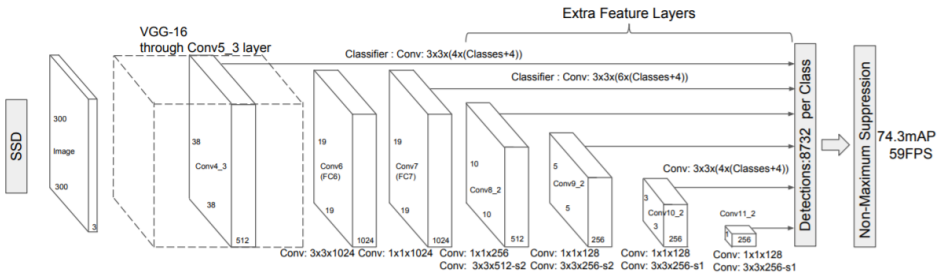


Figure 2.4: SSD architecture [6]

2.1.4 YOLO (You Only Look Once)

YOLO, revolutionises object detection by dividing the image into a grid, with each cell predicting objects within its bounds. This method enables rapid and accurate detections, though it may overlook smaller or overlapping objects. Conceived by Joseph Redmon and colleagues, YOLO assigns each grid cell the task of generating bounding boxes and confidence scores for object presence and box accuracy. It employs non-maximum suppression to filter out overlapping boxes, ensuring only the most accurate predictions remain. With each new version of YOLO, the model architecture has been refined to improve detection accuracy while reducing computational complexity. YOLOv3 introduced multiple scales for object detection, improving performance on smaller objects. YOLOv5 focused on optimising model size and inference speed, while YOLOv8 further refines these characteristics, achieving higher accuracy with fewer parameters.

Evolving from its original version to YOLOv8, the algorithm has seen significant improvements in both precision and speed. Performance charts within the article compare YOLO

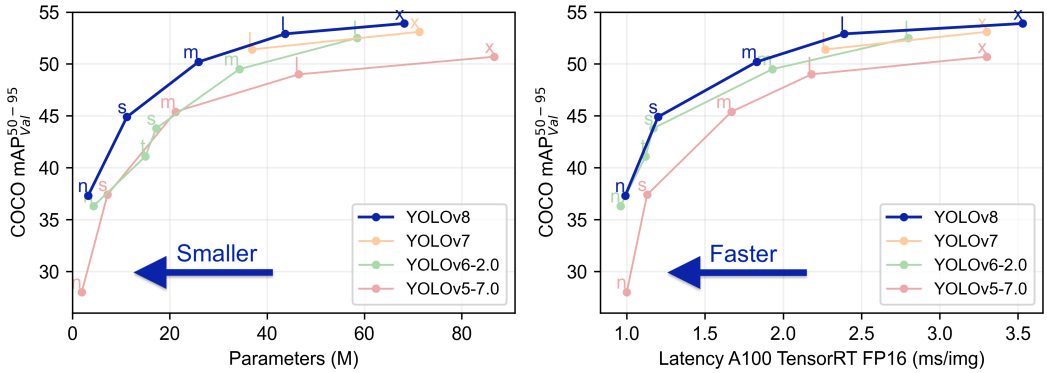


Figure 2.5: Compares the performance of YOLO versions 5 through 8, illustrating the trade-off between model size and accuracy on the COCO dataset (left) and the inference speed on an NVIDIA A100 TensorRT FP16 (right).

versions 5 through 8, showing how YOLOv8 delivers higher accuracy with fewer parameters and quicker inference times, emphasising its refined efficiency and effectiveness in object detection.[26]. See Figure 2.5 for performance comparison. Each iteration has been designed to be lighter and faster, optimising computational resource usage. This makes YOLO applicable across a wide range of devices with varying processing powers, from high-end servers to edge devices in the Internet of Things (IoT) ecosystem [27][28][29].

2.2 Image Classification

Image classification is a task within the field of computer vision, where the objective is to categorise an image into one of several predefined classes. This task is central to many applications, from medical imaging to self-driving cars, and has seen significant advancements with the rise of deep learning techniques.

At the core of image classification is the ability to automatically recognise and label objects or scenes depicted in an image. Unlike object detection, where the model identifies and localises multiple objects within an image, classification focuses on assigning a single label that best represents the content of the entire image. The primary challenge lies in accurately distinguishing between similar classes and ensuring robust performance across diverse environments, lighting conditions, and image qualities.

2.2.1 Convolutional Neural Networks (CNNs)

The introduction of Convolutional Neural Networks (CNNs) revolutionised image classification tasks. CNNs are designed to automatically learn hierarchical features from raw pixel data by applying a series of convolutional filters. These filters detect low-level features such as edges and textures in the early layers, while deeper layers capture high-level semantic information like shapes and objects.

As mentioned in the previous section, a typical CNN architecture consists of convolutional layers, activation functions (such as ReLU), pooling layers, and fully connected layers. Each of these components plays a crucial role in learning and distilling features from images to classify them accurately. The features learned by CNNs are highly effective in

recognising complex patterns, making CNN's the go-to architecture for image classification tasks.

The architecture of CNNs has evolved significantly. Early models like AlexNet [30] were among the first deep networks to achieve impressive results on the ImageNet dataset. This success was followed by more sophisticated architectures such as VGGNet [31] and ResNet [32], which introduced deeper networks while addressing issues like vanishing gradients through the use of skip connections.

2.2.2 Transfer Learning

A key advancement in image classification is the use of transfer learning, where pre-trained models are fine-tuned on specific tasks. Transfer learning leverages models that have been trained on large datasets, such as ImageNet, which contains millions of labelled images across thousands of classes. By fine-tuning these pre-trained models on smaller, task-specific datasets, researchers can achieve high accuracy with reduced training time and less data.

Popular pre-trained models used for transfer learning in image classification include:

- **ResNet:** Known for its residual connections, ResNet has become a popular choice for transfer learning due to its ability to train deep networks without encountering issues like vanishing gradients;
- **EfficientNet:** EfficientNet is a newer architecture that balances accuracy and computational efficiency, making it ideal for tasks requiring high performance on limited resources [33];
- **Inception-v3:** Known for its innovative use of factorised convolutions, Inception-v3 offers a high level of accuracy while maintaining manageable computational demands [34].

2.2.3 Advanced Techniques: Data Augmentation and Regularisation

To improve the performance and robustness of image classification models, various techniques, such as data augmentation and regularisation, are employed.

Data augmentation involves artificially increasing the size and variability of the training dataset by applying transformations such as rotation, flipping, scaling, and cropping. This helps the model generalise better by learning to recognise objects in diverse orientations and conditions.

Regularisation techniques, such as dropout and batch normalisation, are used to prevent overfitting, particularly in deep networks. Dropout randomly deactivates a subset of neurons during training, helping the model become less dependent on specific features, thus improving generalisation.

2.2.4 Challenges and Applications of Image Classification

Despite its successes, image classification still faces several challenges:

- **Class imbalance:** In many datasets, some classes may be overrepresented, while others are underrepresented. This can lead to biased models that perform poorly on minority classes.
- **Generalisation to new data:** While models can achieve high accuracy on training datasets, ensuring they generalise well to unseen data remains a significant challenge, especially when dealing with real-world conditions that differ from the training environment.
- **Adversarial attacks:** Small perturbations in input images, often imperceptible to humans, can drastically change a model's predictions. Developing models that are robust to such adversarial attacks is an ongoing area of research.

Image classification is widely applied across various industries:

- **Healthcare:** In medical imaging, CNNs are used to classify diseases from X-rays, CT scans, and Magnetic Resonance Imaging (MRI) scan, aiding in early diagnosis and treatment planning.
- **Autonomous Vehicles:** Image classification helps self-driving cars identify pedestrians, traffic signs, and other vehicles, playing a crucial role in ensuring road safety.
- **Agriculture:** In precision farming, image classification is used to monitor crop health, identify pests, and optimise irrigation strategies.
- **Retail:** Retailers use image classification to enhance visual search capabilities, enabling customers to find products based on images instead of text-based queries.

2.3 Speech Recognition

Speech recognition, also referred to as Automatic Speech Recognition (ASR) or speech-to-text, is a technology that converts spoken language into written text. It has become an essential tool in various domains, from transcription services to voice-controlled systems, enhancing accessibility and enabling new forms of human-computer interaction. In recent years, advancements in artificial intelligence, particularly deep learning, have revolutionised speech recognition, making it more accurate and versatile in handling diverse audio conditions and languages.

The process typically begins with speech enhancement, where background noise is reduced, followed by feature extraction from the voice signal. The next stage involves acoustic modelling, where the audio features are mapped to phonetic units. This leads to phonetic unit recognition and, finally, the conversion of speech into text, as illustrated in Figure 2.6.

2.3.1 Traditional vs Deep Learning Approaches

Early speech recognition systems relied heavily on acoustic models and Hidden Markov Models (HMMs), which required handcrafted features and were particularly sensitive to background noise and variability in speaker accents [35]. These systems were most effective in controlled environments with limited vocabulary. However, their performance diminished in real-world scenarios with spontaneous or conversational speech due to their inability to generalise across varied environments.

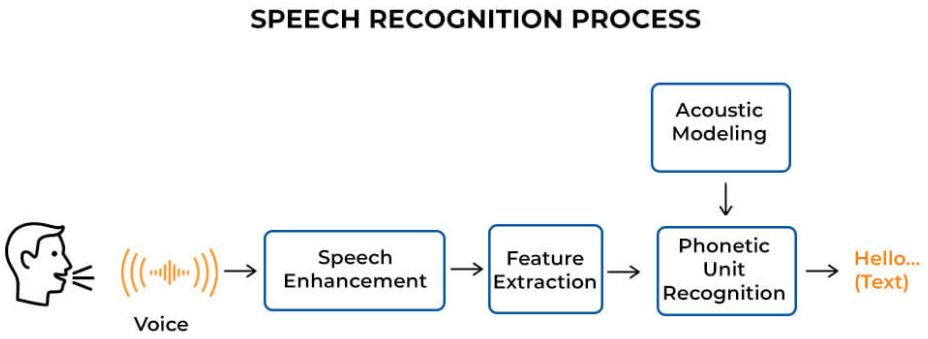


Figure 2.6: Speech Recognition Process Flow (adapted from [7])

The introduction of deep learning marked a significant breakthrough in speech recognition. Models such as Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) allowed for better handling of temporal sequences by capturing context over time. Unlike traditional systems, deep learning models automatically learn feature representations from raw audio data, significantly reducing the need for manual feature engineering and improving robustness to noise, speaker variation, and spontaneous speech [36].

A further evolution came with the adoption of transformer-based models, which utilise self-attention mechanisms to manage long-range dependencies in speech. Transformers, such as those used in Whisper.AI, have dramatically improved transcription accuracy even in complex audio environments by effectively modelling the dependencies within speech sequence [37]. These models are also highly parallelisable, enabling efficient processing of large datasets.

2.3.2 Whisper.AI: A Modern Solution for Speech Recognition

Whisper.AI, developed by OpenAI, represents the cutting edge of modern speech recognition systems. It leverages a transformer-based architecture and is trained on a diverse dataset that includes multilingual and noisy audio samples, making it one of the most versatile systems for handling real-world audio conditions [8]. Whisper.AI's large-scale training enables it to accurately transcribe speech even with background noise, varying accents, and other challenges that typically degrade traditional models.

What distinguishes Whisper.AI from earlier models is its ability to integrate seamlessly into multimedia processing workflows. It can handle both locally stored and cloud-based video files, transcribing audio in real-time or batch mode without the need to separate audio and video. This capability allows for streamlined workflows in industries such as media production, content moderation, and accessibility services.

Additionally, Whisper.AI excels in multilingual transcription and speech translation, making it a valuable tool for global applications. The system's multitask training format enables it to switch between transcription and translation tasks with ease, as shown in Figure 2.7. This flexibility allows Whisper.AI not only to convert speech to text but also to translate non-English speech directly into English.

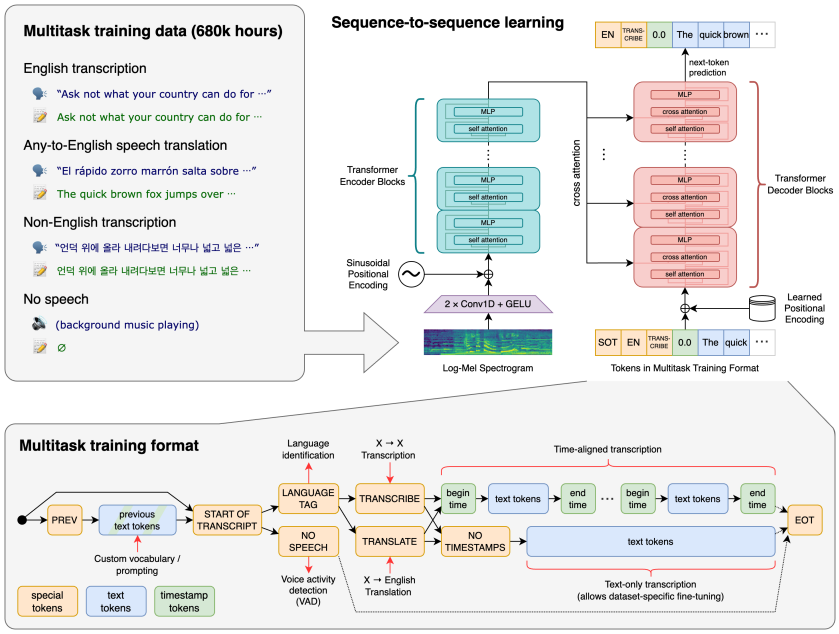


Figure 2.7: Architecture and workflow of Whisper.AI, illustrating the multitask training format, the sequence-to-sequence learning model, and the diverse use cases for transcribing and translating speech data [8].

2.4 Sentiment Analysis

Sentiment analysis, or opinion mining, is a task in NLP aimed at extracting information, such as opinions and emotions, from text. With the increasing need to understand public sentiment, sentiment analysis has found widespread use in domains such as social media monitoring, customer feedback analysis, content moderation, and financial market predictions [38].

Over the years, approaches to sentiment analysis have evolved, transitioning from rule-based methods to advanced deep learning models that offer higher accuracy and better generalisation across diverse datasets.

2.4.1 Traditional Lexicon-Based and Machine Learning Methods

Earlier sentiment analysis methods primarily relied on lexicon-based techniques, where sentiment dictionaries like Sentiment Lexical Resource for Opinion Mining (SentiWordNet) or Affective Norms for English Words (AFINN) mapped words to specific sentiments [39]. While lexicon-based approaches such as Valence Aware Dictionary and sEntiment Reasoner (VADER) [40] were efficient for analysing short, informal text like social media posts, they often struggled with contextual understanding and nuances such as sarcasm. Lexicon-based models are computationally lightweight and interpretable but generally lack the sophistication to handle complex sentence structures or polysemous words.

Machine learning approaches, using classifiers such as Naive Bayes and Support Vector Machine (SVM), further improved sentiment analysis by learning from labelled datasets. Feature extraction techniques like Term Frequency-Inverse Document Frequency (TF-

IDF) and n-grams[1] allowed these models to capture contextual clues. However, these methods still required manual feature engineering and struggled with context-dependent sentiment, limiting their scalability in real-world applications [41].

2.4.2 Deep Learning and Pre-Trained Models

The advent of deep learning marked a turning point for sentiment analysis, with models like CNNs and LSTMs enabling better handling of complex sentence structures and long-range dependencies. CNNs, initially designed for image recognition, were adapted to extract sentiment features from local phrases [42]. Meanwhile, LSTMs and Bidirectional Long Short-Term Memory (Bi-LSTM) networks excelled at processing sequential data, capturing relationships between words over longer contexts [43].

A significant advance came with the introduction of pre-trained language models like Word2Vec[2] [44] and Global Vectors for Word Representation (GloVe) [45], which enabled word embeddings that capture semantic relationships between words. These models allowed words with similar meanings to have similar vector representations, improving performance of sentiment analysis tasks by providing better contextual understanding of the text.

Bidirectional Encoder Representations from Transformers (BERT) has been especially transformative for sentiment analysis due to its ability to understand context from both directions in a sentence, capturing deeper semantic meaning. Pre-trained on vast amounts of unlabelled text and fine-tuned on task-specific datasets, BERT outperforms traditional methods on benchmark sentiment analysis datasets such as the Stanford Sentiment Treebank (SST) [46] and Internet Movie Database (IMDB) movie reviews [47].

2.4.3 Transformer-Based Models Beyond BERT

Building on the success of BERT, newer transformer-based models like Robustly Optimized BERT Pre-training Approach (RoBERTa) [48] and A Lite BERT for Self-supervised Learning of Language Representations (ALBERT) [49] have pushed the boundaries of sentiment analysis by improving model robustness and efficiency. RoBERTa extended BERT by training on larger datasets with more sophisticated training techniques, while ALBERT reduced the model size while maintaining strong performance, making it more accessible for real-world applications.

These models have shown remarkable improvements in handling more nuanced sentiment tasks, especially when dealing with long and complex text, where understanding sentiment contextually across multiple sentences is crucial.

2.4.4 Multimodal Sentiment Analysis and Speech Integration

The rise of multimodal sentiment analysis has broadened the scope of sentiment analysis by integrating audio, video, and text data. Technologies like Whisper.AI enable accurate speech-to-text transcription, which can then be processed by sentiment analysis models, allowing deeper insights into video content, such as social media vlogs or customer review videos [8]. By combining these transcriptions with pre-trained language models like

[1] n-grams: Continuous sequence of n items from a given sample of text or speech

[2] Word2Vec: Continuous bag-of-words and skip-gram models for learning word representations

BERT, it is possible to capture not only the textual sentiment but also non-verbal cues, providing a more comprehensive understanding of user sentiment [50].

This multimodal approach is especially useful in domains like influencer marketing, where visual, verbal, and textual elements all contribute to the overall sentiment conveyed by the speaker or content creator. By integrating models like Whisper.AI for speech-to-text and Contrastive Language-Image Pretraining (CLIP) for video analysis, sentiment analysis has expanded beyond text to encompass entire multimedia experiences, offering richer insights into public opinion and consumer behaviour.

2.5 Optical Character Recognition

Optical Character Recognition (OCR) is a technology in computer vision that enables machines to automatically recognise and extract textual information from images, scanned documents, or videos. It plays a key role in various applications, such as digitising printed documents, automatic number plate recognition, and assistive technologies for visually impaired individuals [51].

2.5.1 Evolution and Advances in OCR Technology

Early OCR systems relied on template matching and handcrafted rules to recognise individual characters, which were limited to specific fonts and high-quality images [52]. The introduction of machine learning, and more recently, deep learning, has drastically improved OCR systems accuracy and their ability to handle diverse fonts, languages, and noisy images [53, 54].

Modern OCR systems leverage deep learning models, particularly CNNs and RNNs, to automatically learn hierarchical features from raw pixel data. These models can process complex images, making OCR more robust in handling low-resolution, noisy, or skewed text [23]. A crucial technique used in deep learning-based OCR is Connectionist Temporal Classification (CTC) loss, which enables models to recognise sequences of characters without requiring precise alignment between the input image and the output sequence [55].

2.5.2 EasyOCR: A Modern OCR Tool

EasyOCR, developed by the Jaided AI team, is a widely-used OCR library built on PyTorch. It supports over 80 languages, making it versatile for both printed and handwritten text recognition [9]. The framework leverages CNNs and LSTMs for text detection and recognition, enabling it to process multilingual text, including non-Latin scripts such as Chinese, Korean, and Arabic [56].

EasyOCR's modular design allows for the integration of state-of-the-art models for text detection and recognition, providing developers with a flexible and efficient tool for OCR tasks. Figure 2.8 illustrates the EasyOCR framework, showcasing its interchangeable modules for incorporating new models.

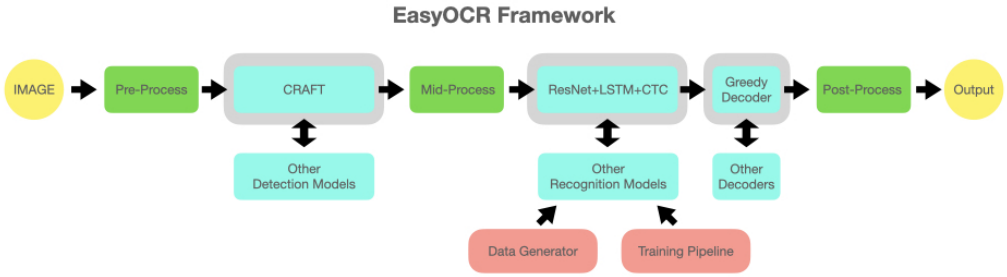


Figure 2.8: EasyOCR Framework, illustrating its modular architecture with placeholders for interchangeable detection and recognition models [9].

2.5.3 Applications of OCR

OCR technology has diverse applications across multiple industries, enhancing automation, accessibility, and data processing capabilities. Some key use cases include:

- **Document Digitisation**: OCR is used to convert printed or scanned documents into searchable, editable text, facilitating large-scale data storage and retrieval [57].
- **License Plate Recognition**: Traffic systems leverage OCR for vehicle identification, assisting with toll collection, law enforcement, and smart city initiatives [29].
- **Assistive Technologies**: OCR systems combined with text-to-speech technology can aid visually impaired individuals by reading printed material or street signs aloud.
- **Multilingual Text Processing**: EasyOCR's support for various languages makes it ideal for global markets, enabling the extraction of text from product packaging, advertisements, and signs in multiple languages [56].
- **Invoice and Receipt Scanning**: OCR streamlines business processes by automating the scanning and processing of receipts, invoices, and financial documents, reducing manual data entry [29].

With the growing demand for real-time OCR in videos and live feeds, the next frontier involves integrating OCR with real-time object detection systems and improving its accuracy in complex environments, such as low-light or noisy backgrounds. As new text detection and recognition models emerge, platforms like EasyOCR are well-positioned to rapidly integrate these advancements, further enhancing the accuracy and efficiency of OCR systems in diverse use cases.

2.6 Microservices and Orchestration

Microservices is an architectural style that structures an application as a collection of loosely coupled services, where each service is designed to perform a specific business function. These services can be developed, deployed, and scaled independently, offering several advantages over traditional monolithic architectures. In contrast to monolithic systems, where all functionalities are integrated into a single application, microservices enable scalability, fault isolation, and flexibility in technology choices [58].

2.6.1 Key Characteristics of Microservices

Microservices architectures have several key characteristics that contribute to their widespread adoption:

- **Decoupling:** Microservices are independent, meaning changes to one service do not necessarily impact others, simplifying updates, maintenance, and testing.
- **Scalability:** Each microservice can be scaled independently, allowing for more efficient use of resources based on demand [59].
- **Autonomy:** Development teams can work on different microservices concurrently, improving development speed and agility [60].
- **Resilience:** Because services are isolated, failures in one microservice do not cascade through the entire system, increasing overall system reliability [59].

2.6.2 Orchestration in Microservices

Microservices communicate with each other through different interaction patterns: coreography and orchestration. In the choreography approach, services communicate directly with each other through events, with no central control. However, in the orchestration approach, a central entity called an orchestrator manages the workflow, ensuring the correct sequence of service interactions. See Figure 2.9 for a representation of orchestration in a microservices architecture.

Orchestration offers several benefits, including centralised control of workflows, simplified error handling, and better monitoring. The orchestrator controls the sequence in which services are invoked, managing dependencies between them [61] [62].

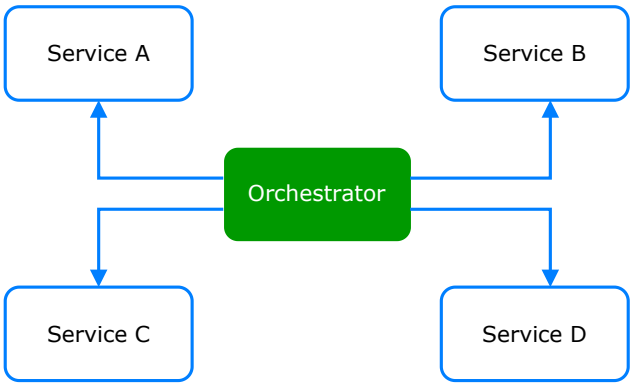


Figure 2.9: Microservices Orchestration.

2.6.3 Advantages of Using an Orchestrator

Adopting an orchestrator provides several advantages for microservices architectures:

Centralised Control: The orchestrator manages service interactions, ensuring that services are called in the correct order and that workflows execute as expected. This is particularly useful for complex workflows involving multiple services.

Error Handling: Orchestrators can simplify error handling by rerouting or retrying service calls in case of failures, improving the system's resilience and robustness.

Monitoring and Logging: Orchestrators offer enhanced monitoring and logging capabilities, providing better visibility into the state of the system. This helps track service performance, detect issues early, and optimise workflows.

The orchestration model ensures consistent interaction between services, particularly when workflows are complex or involve multiple steps. This is beneficial for applications that require tight coordination between services, such as business logic workflows or long-running processes.

2.7 Critical Reflection

The state of the art in machine learning and artificial intelligence presented in this chapter outlines a diverse range of technologies, each with unique strengths and challenges. In reviewing the various subfields—Object Detection, Speech Recognition, Sentiment Analysis, and Optical Character Recognition—it is clear that significant advancements have been made, particularly with the integration of deep learning and transformer-based models. However, each field also faces ongoing challenges, particularly in real-time performance, resource optimisation, and handling complex, diverse data sources.

Object detection has seen rapid development, primarily driven by the use of CNNs and their advanced iterations such as R-CNN, Fast R-CNN, Faster R-CNN, and YOLO. Among these, YOLO stands out due to its exceptional speed and efficiency in real-time object detection. Its architecture, designed to process images in a single pass, makes it significantly faster than traditional region proposal-based methods like R-CNN. This speed advantage is particularly beneficial for applications requiring real-time decision-making, such as autonomous driving or surveillance systems.

One of the key reasons for selecting YOLO in this study is the extensive community support, comprehensive documentation, and availability of tutorials that simplify the process of training YOLO models for specific tasks. This accessibility makes it easier for developers and researchers to fine-tune YOLO for custom tasks, offering a practical advantage in deployment scenarios that require rapid prototyping and customisation.

Despite its advantages, YOLO has limitations. The model tends to struggle with smaller or overlapping objects, and while later versions like YOLOv8 have improved in this regard, there are still cases where its accuracy may fall short compared to slower but more precise models like Faster R-CNN. Therefore, while YOLO excels in speed and ease of use, it may not always be the optimal choice for tasks where detecting minute details or handling dense object scenes is critical.

In speech recognition, systems like Whisper.AI have set new standards with their transformer-based architectures. Whisper.AI's ability to handle diverse languages and accents, combined with its integration into multimedia workflows, makes it a powerful tool for speech-to-text applications. However, like many deep learning systems, the computational cost and resource requirements of transformer models like Whisper.AI remain high. For applications in resource-constrained environments, such as edge devices or mobile platforms, these models may not always be practical. Additionally, Whisper.AI's performance in

real-time scenarios, where low-latency transcription is required, could be further optimised. Despite these limitations, its versatility across languages and accents positions it as a leading solution for complex speech recognition tasks.

Sentiment analysis has evolved significantly with the introduction of pre-trained transformer models such as BERT and RoBERTa. These models evolved into a highly accurate and context-aware field of NLP, especially in capturing nuanced sentiment. Their ability to process both short and long texts, while accounting for context and semantics, marks a substantial improvement over traditional lexicon-based approaches like VADER, which struggle with complex language structures. However, the computational demands of fine-tuning large models like BERT can be prohibitive, particularly for real-time sentiment analysis in dynamic applications such as social media monitoring or customer feedback analysis. Additionally, these models often operate as black boxes, making interpretability and explainability a challenge, especially in industries like finance or healthcare where decision-making transparency is crucial.

Optical Character Recognition (OCR) has also made significant strides, particularly with the introduction of deep learning-based models like EasyOCR. Supporting over 80 languages and capable of processing both printed and handwritten text, EasyOCR is a flexible tool for applications ranging from document digitisation to assistive technologies for the visually impaired. Its lightweight nature and ease of integration make it an appealing choice for developers. However, like many OCR systems, EasyOCR struggles with highly distorted or low-quality text, particularly in noisy images or low-light conditions. Future work should focus on improving the robustness of OCR systems in handling challenging environments.

Image classification, powered by CNNs and enhanced by transfer learning, has similarly benefited from deep learning, with models like ResNet, EfficientNet, and Inception-v3 enabling efficient and accurate classification across various industries. YOLO Classification, leveraging YOLO architecture's strengths in speed and efficiency, is also emerging as a promising approach for real-time image classification, making it suitable for applications like surveillance and industrial automation. However, challenges such as class imbalance, generalisation to new data, and robustness against adversarial attacks still need to be addressed.

One promising approach to managing the growing complexity and diversity of AI models and workflows is the adoption of microservices architectures. Microservices enable each component or model in an AI system to be developed, deployed, and scaled independently. This is particularly useful in applications that combine multiple machine learning models (e.g., object detection, speech recognition, sentiment analysis) into a single system. By decoupling these models into separate services, microservices provide scalability and fault tolerance, ensuring that each model can be maintained or updated without affecting others. Orchestration, where a coordinator manages the interactions between services, ensures smooth integration, especially in complex workflows.

In this context, adopting a microservices architecture seems to be an effective approach for handling different AI models efficiently and ensuring scalability, which is critical in resource-constrained or real-time environments.

A key observation across all the fields discussed is the trade-off between accuracy and computational efficiency. While deep learning and transformer-based models offer un-

paralleled accuracy, their high resource requirements make them challenging to deploy in real-time or resource-constrained environments. As AI continues to evolve, addressing this trade-off between performance and efficiency will be crucial for pushing these technologies forward.

Additionally, while AI technologies are highly advanced, future research should focus on improving explainability and interpretability. In critical applications like healthcare, finance, or law enforcement, AI systems must not only be accurate but also transparent in their decision-making processes. Futhermore, there is a growing need for models that can generalise better to low-resource environments or languages, particularly in the case of sentiment analysis and OCR.

In conclusion, the technologies discussed in this chapter offer powerful tools for addressing complex real-world problems. YOLO's speed and ease of use make it a compelling choice for real-time object detection and classification tasks, while models like Whisper.AI and EasyOCR provide versatile solutions for speech recognition and text extraction, respectively. However, ongoing research is required to overcome the limitations of these systems, particularly in handling complex data, improving efficiency, and enhancing interpretability.

Chapter 3

Related Tools, Techniques, and Studies

This chapter explores existing studies and tools relevant to multimedia data extraction, analysis, and influencer content monitoring. We begin by reviewing significant academic research in the field, followed by an examination of the most prominent tools available today for multimedia and influencer-related tasks.

3.1 Related Studies

One of the essential studies that has contributed to the field is the development of content-based video retrieval systems that can extract valuable information from videos has become more apparent. In Phan et al.'s work on video retrieval, deep learning techniques such as Faster R-CNN ResNet and SSD MobileNet V2 were utilised to extract important features like subtitles, speech, and objects from video frames [10]. Their work highlights the importance of using distributed systems such as Apache Spark to optimise processing times, reducing the cost and resources typically associated with large-scale video analysis.

Similarly, Rangaswamy et al. explored the intersection of sentiment analysis and metadata extraction from YouTube videos. Their research focused on using sentiment analysis to classify video content based on comments and metadata, providing a way to identify potentially harmful or illicit content [63]. This study demonstrates the growing importance of analysing multimedia content for sentiment, especially given the rise of social media platforms like YouTube.

Another relevant study in this domain is the exploration of sponsored content in YouTube videos using information extraction techniques. Santos Rodrigues et al. presented a method that identified sponsored content based on audio transcriptions and keyword analysis [14]. The research highlights the role of digital influencers in marketing and how extracting valuable information from video content can assist businesses in decision-making processes related to advertising campaigns.

In addition to these studies, Kamis and Goularas compared various deep learning techniques for sentiment analysis using Twitter data. Their work evaluated the performance of CNNs and LSTMs in sentiment classification tasks. They demonstrated that while

these models offer high accuracy, there is still room for improvement, particularly in NLP techniques applied to social media content.

Moreover, social media has proven to be a valuable source of data for understanding public sentiment and trends. In a study by Štajner et al., interactive diversity analysis was conducted using data extracted from social media posts. Their research focused on reputation management and the role of sentiment analysis in evaluating public opinion. This work underscores the potential of sentiment analysis tools to extract actionable insights from vast amounts of unstructured data, which is increasingly relevant in today's digital economy.

Finally, Soliev et al. explored the future of video data extraction using Python and AI for video-based information recognition. Their study provided practical examples of how Python libraries such as OpenCV and deep learning models like YOLO can be applied to real-world problems, such as license plate recognition and text extraction from videos [64]. Their work highlights the versatility of AI in addressing various challenges related to video content analysis.

3.2 Current Landscape of Online Tools

In today's digital age, multimedia data extraction and analysis have become essential for understanding influencer activities and content across various social media platforms. Tools in this space not only enable influencer monitoring and compliance but also allow for the detailed extraction and analysis of data from videos and images. These tools utilise AI and machine learning algorithms to streamline tasks such as object detection, speech recognition, and sentiment analysis, making them invaluable for brands and businesses seeking to optimise their digital strategies. This section provides an overview of some of the most prominent online tools, highlighting both those tailored for influencer monitoring and those specialised in extracting and analysing multimedia content from videos and images.

3.2.1 Specialised Tools for Influencer Monitoring and Campaign Insights

In the rapidly evolving landscape of influencer marketing, brands and organisations require powerful tools to monitor influencer activity, ensure compliance with advertising regulations, and gain insights into the effectiveness of campaigns. Several platforms have emerged to meet these needs, offering AI-driven solutions that streamline influencer discovery, content tracking, and performance analysis across major social media platforms. These tools not only facilitate compliance with industry standards but also provide in-depth metrics to help brands optimise their marketing strategies. Below, we explore some of the leading platforms designed specifically for influencer monitoring and campaign insights: Infludata, Influencer Monitor, and Storyclash, each offering unique features tailored to influencer marketing needs.

Infludata offers a comprehensive AI-powered platform for discovering and analysing influencers across various social media platforms. Its key features include:

- **Discover Creators and Brands:** Infludata's AI-powered category search and summaries help users find the perfect creators for their brand by accurately target-

ing niches. The tool summarises profiles based on content and identifies the ideal category match.

- **Audience and Analytics Insights:** Infludata provides over 50 metrics for evaluating creators, including audience location, demographics, and platform performance. It also offers insights into the authenticity of follower engagement and the success of past campaigns.
- **Tracking, Reporting, Analytics, and Social Listening:** Brands can monitor their mentions, posts, stories, reels, and engagement in real-time, staying up to date with their social media performance. The platform summarises comments, provides positivity rates, and even detects controversial topics (e.g., "Shit-Storm Detection") [65].
- **All-in-One Social Media Platform:** Infludata supports analytics for Instagram, YouTube, and TikTok, offering specialised metrics such as:
  - **Instagram:** Analytics for over 15 million influencers, including audience insights and content exploration by hashtags and mentions.
  - **YouTube:** Performance analysis and audience insights for over 100k YouTube Shorts creators and 5 million regular YouTube creators, with sentiment analysis for YouTube comments.
  - **TikTok:** Detailed analysis of over 10 million TikTok creators worldwide, with audience insights and growth tracking based on keywords and hashtags.

Influencer Monitor is used by self-regulatory organisations, industry associations, creator agencies, and brands to manage influencer marketing compliance and content monitoring. Its key features include:

- **Target and Calibrate:** Users can track specific creator channels and adjust content filters based on their monitoring needs, ensuring focus on relevant content.
- **Detect and Act:** Influencer Monitor detects advertising content and ensures compliance with advertising rules. Users can reach out to parties generating problematic content and report violations to the public or relevant industry bodies.
- **Social Media Monitoring:** Influencer Monitor tracks content across platforms like YouTube, Instagram, and TikTok, with daily updates. The platform supports sorting by descriptions, captions, speech, and on-screen text or underage children [66].
- **Analytics and Reporting:** The platform provides detailed statistics and content filters to support informed decision-making, improving the efficiency of influencer marketing compliance.

Storyclash is a platform designed to monitor influencer marketing campaigns and social media activity across platforms. Its primary features include:

- **Monitoring and Reporting on Influencer Collaborations:** Storyclash enables users to create reports and measure the impact of ongoing influencer collaborations, such as product launches and awareness campaigns.

- **Instagram Stories and Text Recognition:** Storyclash is particularly effective at monitoring Instagram Stories, preserving them beyond the 24-hour lifecycle. It uses text recognition technology in images and videos to ensure brands catch all mentions.
- **Influencer Performance Analysis:** The tool evaluates influencer content quality, tracking metrics like fan growth and analysing both organic and sponsored posts across all platforms [67].

This subsection examined two interconnected yet distinct areas: influencer content monitoring, which focuses on regulatory compliance, and campaign analysis or influencer discovery, aimed at marketing optimisation. Tools like Influencer Monitor are designed to ensure adherence to advertising regulations and content standards, while platforms such as Infludata and Storyclash provide insights into campaign performance and assist brands in discovering the ideal influencers for their strategies.

While these tools share common processes, such as data scraping and analysis, their objectives differ. Influencer Monitor prioritises compliance, whereas Infludata and Storyclash focus on marketing insights. However, there is a growing overlap between these domains, highlighting the potential for integration between compliance monitoring and marketing analytics, particularly in real-time applications.

As influencer marketing continues to grow and the volume of multimedia data increases exponentially, the need for specialised tools will also expand. Future trends may include more advanced AI-driven techniques to offer scalable, real-time solutions. General-purpose AI platforms like Azure AI Video Indexer, IBM Watson, and Google Cloud AI already provide powerful solutions for multimedia data extraction and analysis, with features such as speech-to-text, object detection, and sentiment analysis. These platforms, although designed for broader purposes, illustrate how AI-driven technologies can enhance both influencer content monitoring and campaign analysis.

3.2.2 Multimedia Data Extraction and Analysis

Several tools have been developed to automate and improve multimedia data extraction and analysis. These tools leverage cutting-edge AI technologies to perform tasks like speech recognition, object detection, and sentiment analysis across various content types. Tools such as IBM Watson and Google Cloud AI offer pre-built Application Programming Interface (API) for tasks such as speech recognition, sentiment analysis, and video classification. While these platforms provide scalable solutions, they often come with limitations in terms of customisation and the need for large-scale resources. For example, the Google Cloud Video Intelligence API provides automated video labelling and scene detection [68], while Watson's Natural Language Understanding service offers robust sentiment analysis capabilities across multiple domains [69]. These tools are suitable for general-purpose applications but may lack the fine-tuning capabilities needed for niche applications like influencer content monitoring, which requires tailored models and real-time data extraction insights.

The Google Cloud Vision API is another essential tool in multimedia data analysis. It offers a comprehensive set of features for image analysis, including object detection, label detection, and OCR. This API enables developers to automate image tagging, detect faces, landmarks, logos, and even classify explicit content. A key advantage of Google Cloud

Vision is its ability to scale for large image databases while maintaining high levels of accuracy and efficiency. It is particularly useful for projects where image metadata needs to be generated automatically, or when objects within an image need to be identified for further analysis. The API has been successfully applied in many use cases, from retail product recognition to real-time image search in media archives. Figures 3.1, 3.2, 3.3, and 3.4 provide examples of how the API processes images to extract valuable insights.

Figure 3.1: Google Cloud Vision API OCR result example.

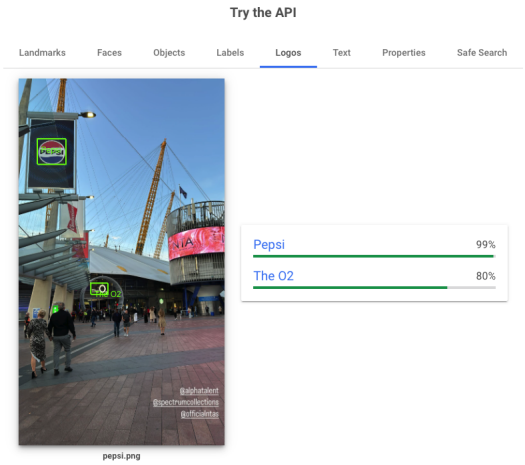


Figure 3.2: Google Cloud Vision API logo detection example.

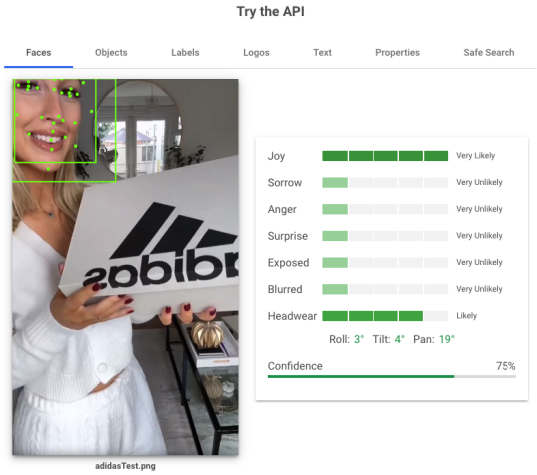


Figure 3.3: Google Cloud Vision API facial recognition example.

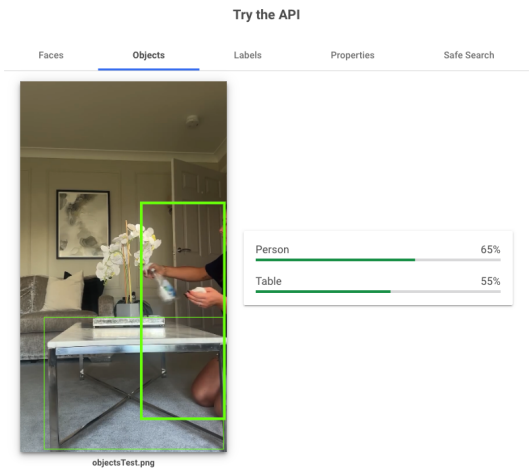


Figure 3.4: Google Cloud Vision API objects detection example.

Amazon Rekognition stands out as a versatile tool in the realm of multimedia data extraction and analysis. With its ability to process both images and videos, Rekognition leverages pre-trained models to perform a wide range of tasks, including object detection, facial analysis, and activity recognition. One of its key strengths is its facial recognition capabilities, allowing for the detection, analysis, and comparison of faces - particularly useful for identity verification in security applications. Additionally, Rekognition can estimate a person's age, which is valuable for ensuring compliance with regulations in content involving children. Its ability to detect unsafe content, such as violence or explicit material, makes it a critical tool for content moderation across social platforms and media outlets.

Rekognition also supports real-time video analysis, enabling it to track people, objects, and activities across video frames, making it a valuable tool for security and surveillance applications. Its ability to detect text in images and videos (OCR) adds further utility, as does its capability to recognise celebrity faces for entertainment or media-related applications. The high scalability and integration with AWS infrastructure make it suitable for large-scale projects requiring fast processing and robust analysis.

Figures 3.6 and 3.5 showcase examples of Rekognition's facial analysis and comparison features.

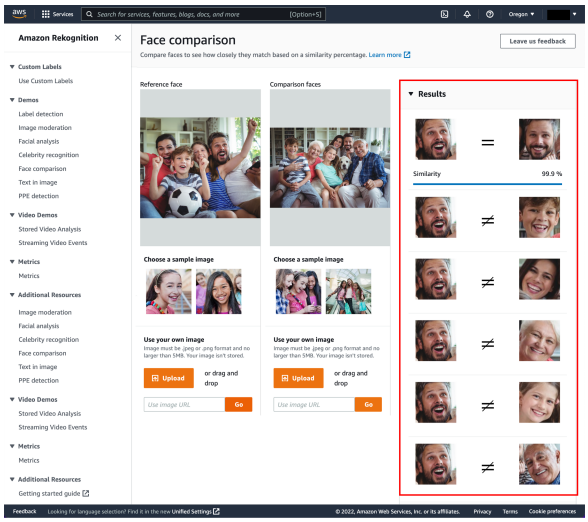


Figure 3.5: Amazon Rekognition face comparison feature.

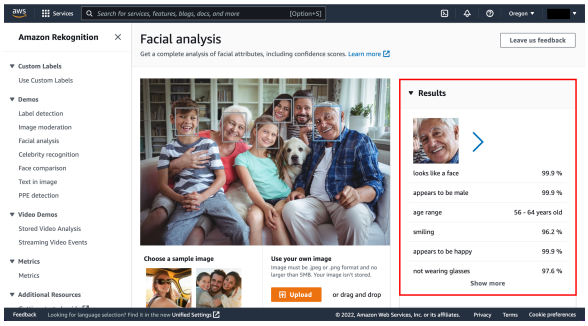


Figure 3.6: Amazon Rekognition face analysis feature.

Azure AI Video Indexer is another powerful tool that allows users to extract insights from video content using AI technologies like speech-to-text, facial recognition, object detection, and sentiment analysis. It offers advanced features for identifying people, locations, emotions, and key topics within videos [70]. Azure Video Indexer is particularly effective for large-scale multimedia projects that require automated content analysis and metadata extraction. However, like other general-purpose tools, it may lack the specific customisations needed for influencer content monitoring. A free trial of the application, along with its interface and features, can be seen in Figures 3.7 and 3.8.

3.2.3 Content Tagging and Retrieval Systems

Content tagging and retrieval systems have revolutionised the way multimedia data is processed, enabling the automatic classification and identification of elements within images and videos. These systems are intrinsic to modern platforms like Google Photos, Facebook, and iCloud, providing users with seamless ways to manage and search large amounts of multimedia content.

Platforms such as Google Photos employ sophisticated AI models to automatically tag images with attributes like locations, objects, or activities without user intervention. By utilising deep learning models such as CNNs, Google Photos can tag and retrieve photos based on recognised patterns, making search highly efficient. Similarly, Facebook uses its AI framework, DeepFace, to tag individuals in photos, making facial recognition a core

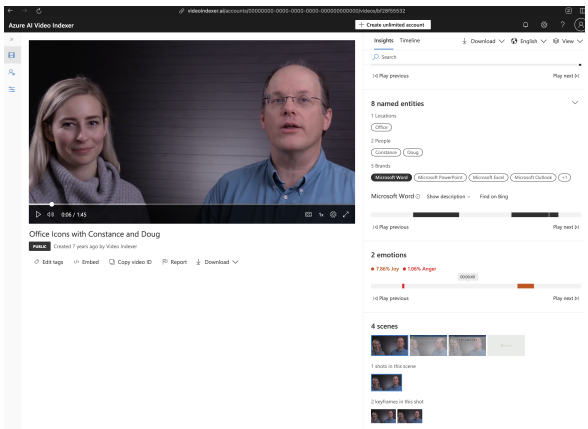


Figure 3.7: Azure Video Indexer Interface with named entities, emotions, and scene breakdown.

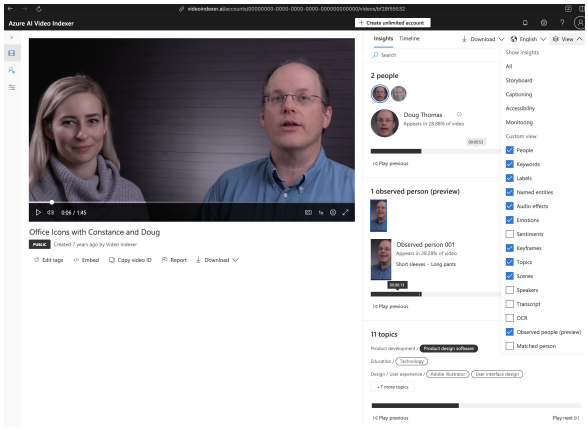


Figure 3.8: Azure Video Indexer Interface showing insights on detected people and topics in the video.

feature of social interactions on the platform (Figure 3.10)[71]. The combination of facial recognition and content tagging has enhanced how users search and retrieve personal and social media content.

Moreover, iCloud Photos integrates AI-powered tagging systems that allow for facial recognition and object detection. iCloud automatically categories photos based on people, objects, and scenes, enabling users to organise and search their content effectively (Figure 3.9).

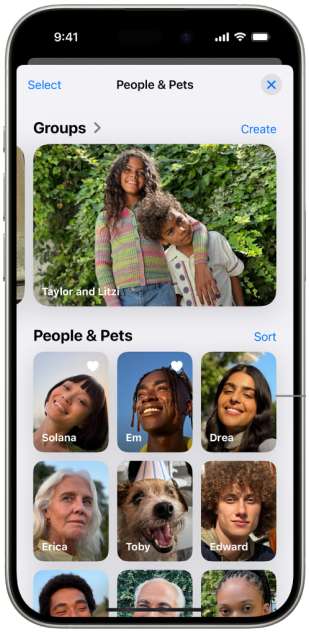


Figure 3.9: People and Pets identification by iCloud.

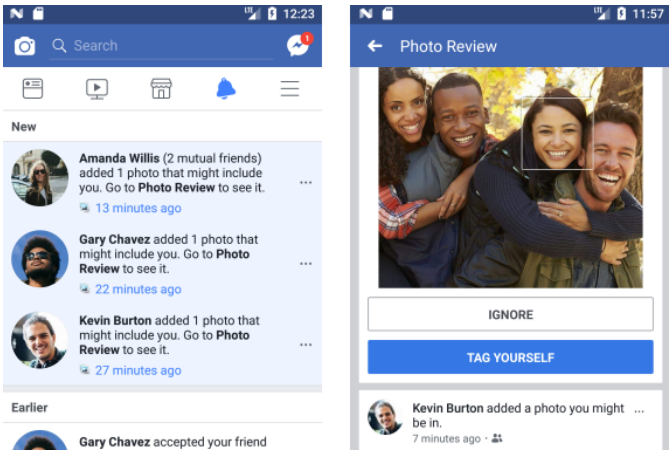


Figure 3.10: Facebook tagging face recognition.

Chapter 4

System Design and Implementation

This chapter outlines the design and implementation of the proposed system, which is a proof of concept (POC) developed using a microservices-based architecture. The system aims to extract and analyse multimedia content, including videos and images, through various AI-driven services such as object detection, logo detection, image classification, speech recognition, OCR, and sentiment analysis.

The system is built in Python and designed with scalability and flexibility in mind, using a modular approach where each service functions independently. A user-friendly frontend, created with Streamlit, allows users to upload media files and select which services to run on the content. Depending on the type of content—image or video—the system processes the data by uploading it to AWS S3 and publishing tasks to a coordinator, which manages the execution of specific processing services. Each service downloads the content from AWS S3, processes it, and stores the results in a MongoDB instance.

The following sections provide a detailed overview of the system architecture, cloud infrastructure, and scalability.

4.1 Overall System Architecture

The proposed system is built using a microservices-based architecture, ensuring modularity, flexibility, and scalability to handle large datasets and complex multimedia processing tasks. Each microservice operates independently, allowing for seamless scaling, load balancing, and fault tolerance. By decoupling tasks into distinct services and employing message queues for inter-service communication, the system is highly resilient and can process tasks concurrently, reducing bottlenecks and improving overall performance.

The overall structure of the system is depicted in Figure 4.1, showing the relationships between the user interface, microservices, data store, and file storage components.

At the core of the application is the user interface, developed with Streamlit, which provides an intuitive and interactive platform for users to upload multimedia content (videos or images) and select various analysis services. These services include advanced AI models for object detection, image classification, logo detection, optical character recognition, speech recognition, and sentiment analysis. A summary of the system's AI services and their corresponding functionalities is provided in Table 4.1.

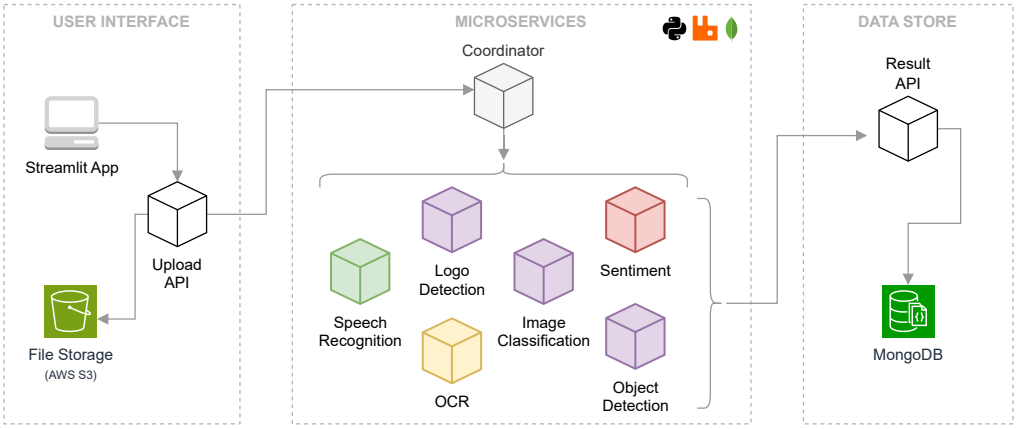


Figure 4.1: System Architecture

4.1.1 User Interaction and Workflow

The Streamlit user interface simplifies interaction between users and the backend microservices by offering the following functionalities:

- **Upload**: Users can upload multimedia content in the form of videos or images.
- **Service Selection**: Users have the flexibility to choose which analysis services to apply to their content. Available services include object detection, optical character recognition (OCR), speech recognition, logo detection, image classification, and sentiment analysis.
- **Custom Settings**: For video uploads, users can configure specific parameters, such as the frame rate for extracting individual frames. This ensures that the content is pre-processed according to the user's needs.
- **Language Selection**: Users can optimise the OCR and Whisper speech recognition services by specifying the target languages for text or audio extraction.

Once the user has uploaded the content and selected the desired services, the system initiates the processing pipeline. The upload service uploads media files (either images, video frames, or the entire video) to an `AWS S3 bucket`, which serves as the primary storage for all media content. This design leverages cloud storage for efficient data management while supporting distributed processing across microservices.

Table 4.1: Summary of AI Microservices and Their Functions

Service	**Functionality**	**Task**
yolo-service	Object Detection	Detect objects in images and video frames
yolo-cls-service	Image Classification	Classify images into categories
yolo-logo-service	Logo Detection	Detect specific logos in media
whisper-service	Speech Recognition	Convert audio to text
ocr-service	Optical Character Recognition	Extract text from images and video frames
sentiment-service	Sentiment Analysis	Analyse the sentiment of extracted text
upload-service	Upload Files	API for uploading files and trigger coordinator
coordinator-service	Manage services	Controls and manage services
result-service	Stores Data	API for storing service results

4.1.2 Processing Workflow and Orchestration

After the media files are successfully uploaded to the S3 bucket, a message is dispatched to the `coordinator-service`, which orchestrates the distribution of tasks. As depicted in Figure 4.2, the sequence diagram outlines the detailed flow of operations, showing how each task is routed to the appropriate microservice. The `coordinator-service` ensures that YOLO handles object detection, Whisper.AI converts speech into text, and EasyOCR processes text extraction from images. Additionally, the XLM-Roberta sentiment analysis service evaluates the text generated by Whisper to determine the sentiment in the media content. This structured sequence allows for efficient parallel processing across services, optimising the overall workflow.

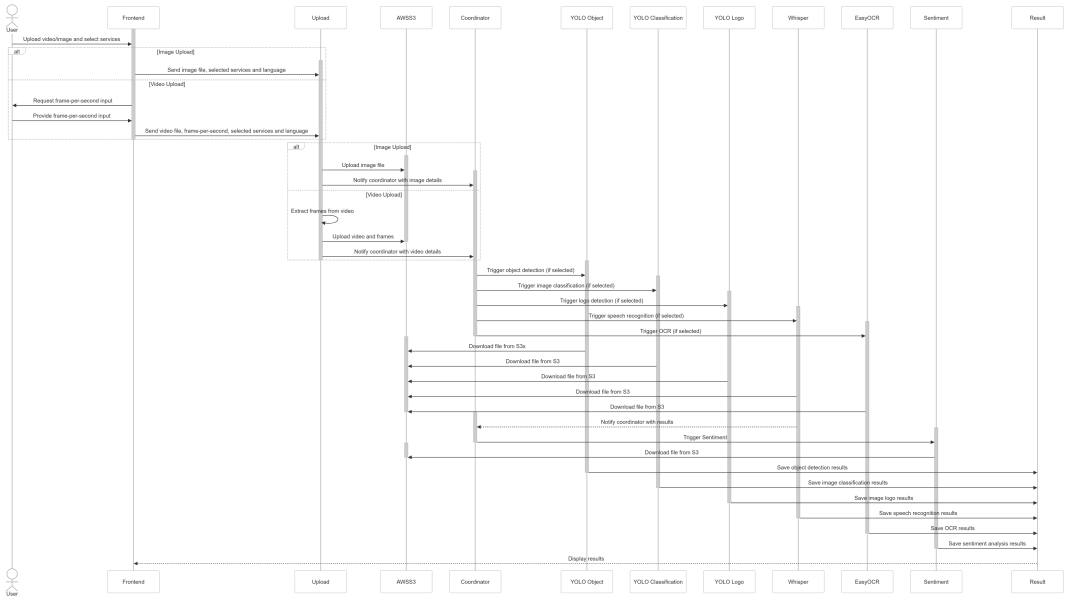


Figure 4.2: Sequence diagram showing the process of uploading and processing a file.

Each microservice in the architecture follows this sequence:

1. Downloads the required files from AWS S3 to a temporary folder;
2. For video content, sorts the frames into the correct sequence;
3. Processes the data according to its specific function (e.g., object detection, speech recognition, etc.);
4. Sends the processed results and status back to the `result-service`, which stores the results in a MongoDB database.

The system's architecture promotes flexibility by allowing each service to scale independently. For instance, if the system experiences increased demand for speech-to-text conversions, multiple instances of the Whisper microservice can be launched without affecting the object detection service. This ensures efficient resource utilisation and supports high-throughput processing.

As illustrated in Figure 4.3, the layered architecture highlights the different system layers - from input and processing to storage and presentation - ensuring a modular and scalable design.

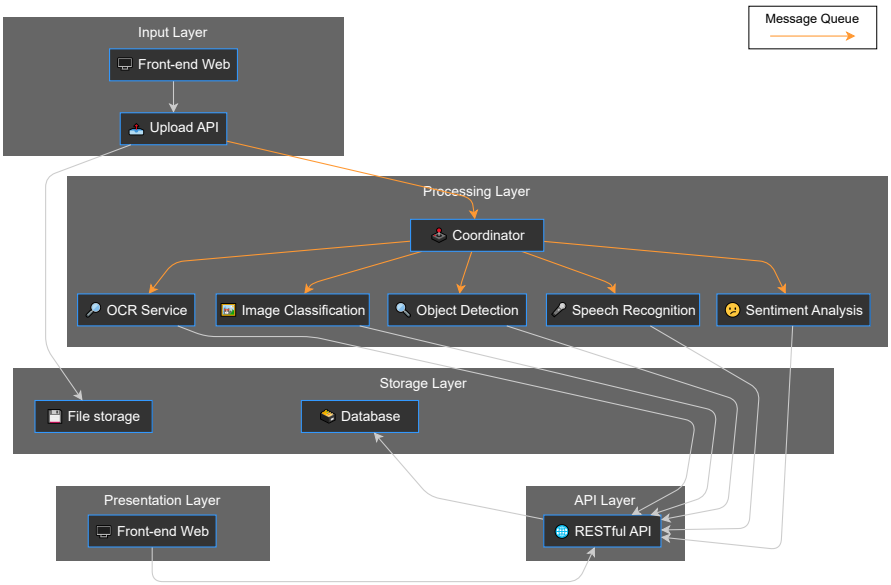


Figure 4.3: Layered Application Diagram

4.1.3 User Feedback and Progress Tracking

In addition to service orchestration, the application provides users with real-time feedback on the status of their submitted tasks. A comprehensive table displays metadata for the uploaded files (e.g., file type, frame rate, selected languages) and tracks the progress of each selected service (e.g., completed, processing, failed). This feedback mechanism ensures that users can monitor the status of their tasks and receive results as soon as they are available. A more detailed explanation of the user interface and feedback features can be found in Section 4.7, where the various components and functionalities of the application's user interface are described in depth.

4.1.4 Decoupled Architecture for Scalability and Fault Tolerance

The microservices architecture is designed with a decoupled approach, where each service handles a specific task independently. This separation of concerns makes the system highly modular and scalable. For instance, if one service experiences high demand or requires maintenance, it does not affect the functionality of other services. Additionally, the use of message queues for communication between the upload service, coordinator, and processing microservices ensures that tasks are distributed asynchronously, avoiding potential bottlenecks.

By leveraging cloud-native technologies such as AWS S3 for storage and Docker for containerised deployments, the system can scale horizontally, meeting increased demand by simply adding more instances of the necessary microservices. This design makes the system well-suited for processing large-scale multimedia datasets efficiently.

The frontend was initially built for testing the application and is designed to receive one video or image at a time. However, after uploading one video or image, the system is

ready to accept another upload, even if the services for the previous upload have not yet finished processing. This allows for continuous uploads without waiting for ongoing tasks to complete.

To fully integrate this application into a production environment for handling higher volume of content, it would be necessary to implement a queue for managing multiple file requests. This queue would regulate the inflow of media files and distribute them across the necessary services, ensuring that the system can efficiently process multiple uploads concurrently without overloading the microservices.

4.2 Object Detection with YOLOv8

When this project began, YOLOv8 was the latest version of the YOLO models, known for their real-time processing capabilities and balance between speed and accuracy. YOLOv8 brought significant advancements in terms of detection precision, ease of use, and efficiency across various tasks like object detection, classification, and segmentation.

Throughout the development of this thesis, however, YOLO has continued to evolve, and by the time of writing, version 11 (YOLOv11) had already been released [72]. This rapid advancement demonstrates YOLO's powerful and ever-growing capabilities in computer vision. Each new iteration builds on the previous versions, introducing refined architectures and improving performance metrics.

In the system, YOLOv8 plays a central role in detecting objects and analysing images. Three distinct services were implemented to cater different detection needs: the `yolo-service` for general object detection, the `yolo-cls-service` for image classification, and the `yolo-logo-service`, which is fine-tuned for detecting logos. Further details on the logo detection training process are covered in the next Section 4.2.2

One of the key advantages experienced with YOLOv8 was its flexibility and ease of training, which enabled impressive results to be achieved within a short time frame. The continuous improvements made to YOLO and its widespread adoption underscore its significance as a powerful tool in AI-driven computer vision. A more in-depth discussion of the results produced by these services is provided in the section on YOLO results.

To further illustrate the range of YOLOv8's capabilities, the following tables list the available YOLOv8 models for object detection and image classification Table 4.2 and 4.3.

Table 4.2: YOLOv8 Models Performance (COCO)

Model	Size (px)	mAP (%)	CPU ONNX (ms)	A100 TRT (ms)	Params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

4.2.1 Using Pre-Trained YOLOv8 Models

YOLOv8 simplifies the process of deploying object detection models by offering pre-trained versions that can be used directly for a wide range of tasks. These models,

Table 4.3: YOLOv8 Classification Models (ImageNet)

Model	Size (px)	Acc1 (%)	Acc5 (%)	CPU ONNX (ms)	A100 TRT (ms)	Params (M)	FLOPs (B)
YOLOv8n-cls	224	69.0	88.3	12.9	0.31	2.7	4.3
YOLOv8s-cls	224	73.8	91.7	23.4	0.35	6.4	13.5
YOLOv8m-cls	224	76.8	93.5	85.4	0.62	17.0	42.7
YOLOv8l-cls	224	76.8	93.5	163.0	0.87	37.5	99.7
YOLOv8x-cls	224	79.0	94.6	232.0	1.01	57.4	154.8

pre-trained on large datasets like Common Objects in Context (COCO), are able to recognise a variety of common objects right out of the box. For developers looking to integrate object detection into their applications, these pre-trained models can be implemented with minimal setup, making them an ideal solution for scenarios where fast deployment is crucial.

To use a pre-trained YOLOv8 model, developers can simply load the model and run predictions on new images or videos, allowing the model to detect objects without additional training. This is particularly useful when there is no need for task-specific customisation. However, for more specialised use cases, such as detecting specific logos or classifying images into custom categories, fine-tuning the model might be required. YOLOv8 makes it easy to transition from using a general-purpose pre-trained model to training a custom model for specific tasks. By leveraging transfer learning, users can adapt the pre-trained model to new datasets with minimal effort.

In this application, the YOLO-related services follow a similar structure: the `yolo-service` (for object detection), `yolo-logo-service` (for logo detection), and `yolo-cls-service` (for image classification). Each service operates with the same approach, with the only differences being the specific model loaded and the structure of the results returned after processing. This allows for easy scaling and modification of the services to suit different tasks while maintaining a consistent workflow.

The following code demonstrates how to load a YOLOv8 model, predict and show the results. The Listing 4.1 showcases the simplicity of working with pre-trained YOLOv8 models in Python.

```
1  from ultralytics import YOLO
2
3  # Load the pre-trained YOLOv8 model
4  model = YOLO("yolo8n.pt")
5
6  # Perform object detection on an input image
7  results = model("path/to/image.jpg")
8
9  # Visualize the detection results on the image
10 results[0].show()
```

Listing 4.1: Using YoloV8

Message Structure for YOLO Services

Each of the YOLO-related services receives a message from the coordinator indicating whether the task involves processing an image or a video. The message structure is simple and consistent, with only the path to the relevant media file (either the image or the directory containing the extracted frames from the video) being passed along with the `item_id`. The services then use this information to download and process the appropriate media.

For an image, the message includes the path to the image, as shown in Listing 4.2.

```
{
  "item_id": "2b7d9785-b5c2-4f64-b6b7-74f86e2882b9",
  "image_path": "images/2b7d9785-b5c2-4f64-b6b7-74f86e2882b9/imageExample.png"
}
```

Listing 4.2: Message for Image Processing

For a video, the message includes the path to the directory containing the extracted frames, as shown in Listing 4.3.

```
{
  "item_id": "0dbb7728-d60f-459f-a270-ea96b4440d2c",
  "frames_path": "video/0dbb7728-d60f-459f-a270-ea96b4440d2c/frames/"
}
```

Listing 4.3: Message for Video Processing

The process is identical for other services, such as the `yolo-logo-service` and `yolo-cls-service`, where the only change is the model loaded and how the results are structured for each type of detection task (objects, logos, or classifications). This consistent messaging format ensures that the microservices are modular and can be easily scaled or modified for future requirements.

Expected Results for YOLO Services

An example of the expected output is presented below for the `yolo-cls-service` as well as both the `yolo-service` and `yolo-logo-service`. When processing videos, the system analyses each frame individually, meaning there will be a separate result object for each frame in the video.

```
{
  "yolo_cls_result": [
    {
      "top1_class_idx": 906,
      "top1_class_name": "Windsor_tie",
      "top1_confidence": 0.4241333603858948,
      "top5_classes_indices": [906, 617, 834, 783, 794],
      "top5_class_names": ["Windsor_tie", "lab_coat", "suit", "screw", "shower_curtain"],
      "top5_confidences": [0.4241333603858948, 0.0652686357498169, 0.04082207099192814,
          0.023162393953985852, 0.02084171213209629]
    }
  ]
}
```

Listing 4.4: YOLO-CLS Classification Results Example

```
{
  "yolo_result": [
    {
      "box": [
        482.1490478515625,
        485.848876953125,
        719.68798828125,
        935.8724365234375
      ],
      "confidence": 0.37620508670806885,
      "class": 0,
      "label": "person"
    }
  ]
}
```


Listing 4.5: YOLO Object Detection Results Example

4.2.2 Training YOLOv8 for Logo Detection

One of the objectives of this project is to enable the introduction of new models in a seamless manner while demonstrating how to train these models for specific tasks like logo detection. YOLOv8's flexibility makes it particularly suitable for this. By leveraging the model's pre-trained weights, users can fine-tune the model on custom datasets, reducing the time and computational resources required for training while achieving high accuracy for specialised tasks.

The following flowchart (Figure 4.4) outlines the step-by-step process of training YOLOv8 for logo detection. Each stage in the flow corresponds to a crucial part of the training process.

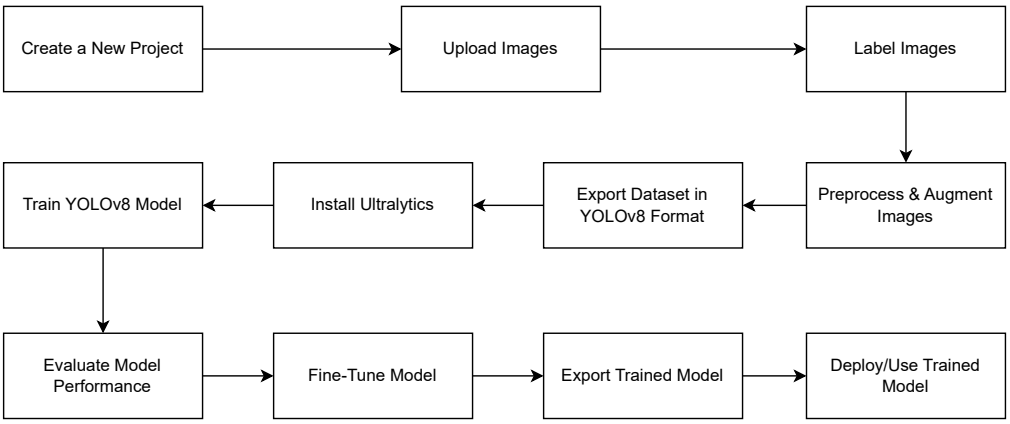


Figure 4.4: Training process flow for YOLOv8 logo detection model.

To begin the process of training YOLOv8 for logo detection, the first step involves selecting a dataset that contains labelled images of logos. This dataset can either be sourced from existing public collections or created manually. In this project, Roboflow was used, a platform that simplifies the process of dataset creation and management by providing tools for image labelling, including drawing bounding boxes around objects of interest, such as logos [73]. See Figure 4.5 for a Roboflow project overview.

Other platforms, like Ultralytics Hub, offer similar functionalities, allowing users to upload datasets and train either with Ultralytics Cloud, Google Colab or with your own agent.

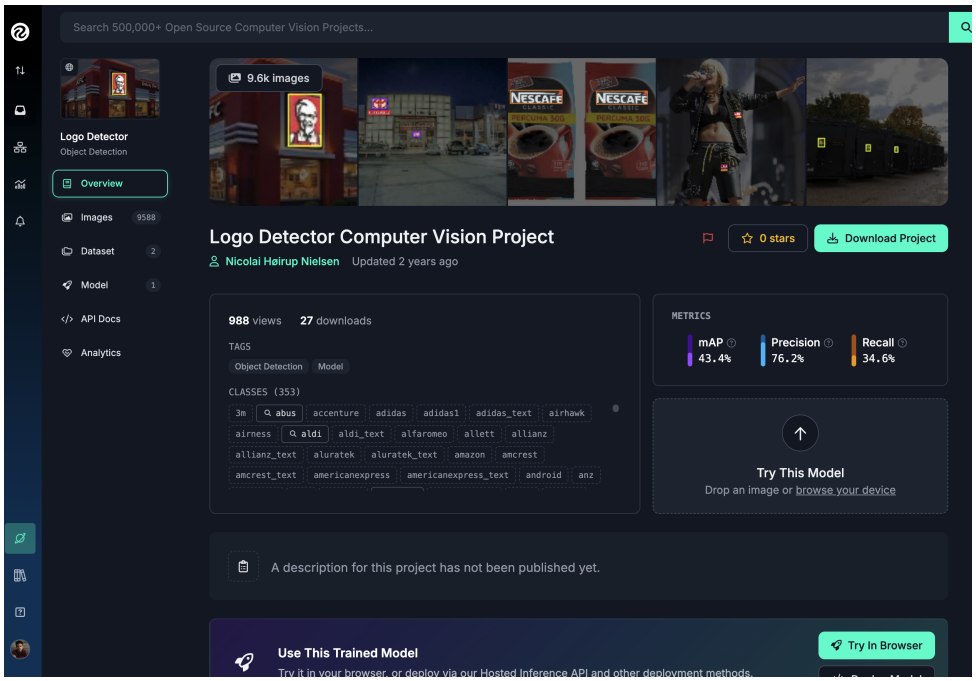


Figure 4.5: Roboflow Logo Detector project overview.

These platforms significantly reduce the overhead associated with dataset preparation, as they provide intuitive interfaces and automation tools for organising, training and deploy models [74]. See Figure 4.6 for a Ultralytics Hub training user interface.

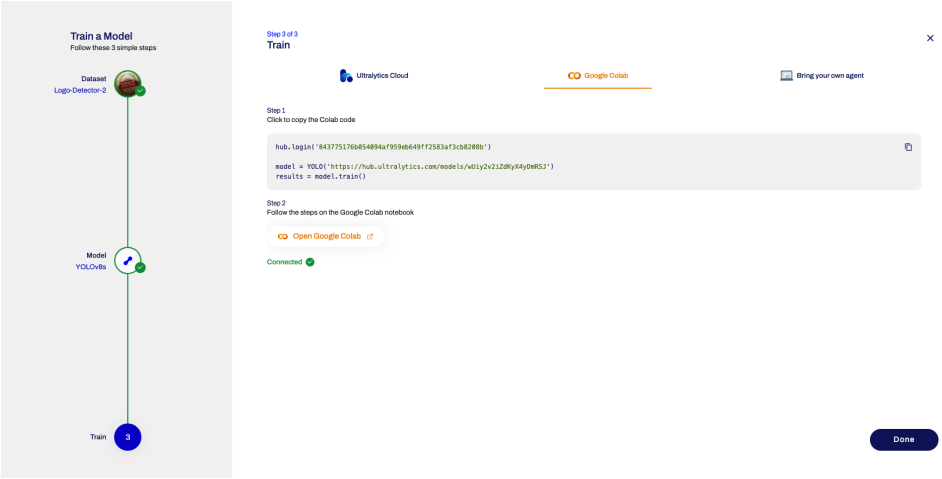


Figure 4.6: Ultralytics Hub UI.

In a custom dataset scenario, each image must be labelled with bounding boxes that identify the logos present, specifying the exact coordinates of the bounding box around the logos. This ensures that the model can learn from the labelled data and develop the capability to recognise logos in unseen images accurately. See example of the labelling process in the Figure 4.7

Once the dataset is ready, the next steps involve preprocessing and augmenting the images. Preprocessing ensures the images are consistent (resizing, normalisation, etc.), while augmentation introduces variability by applying transformations like flipping, cropping,

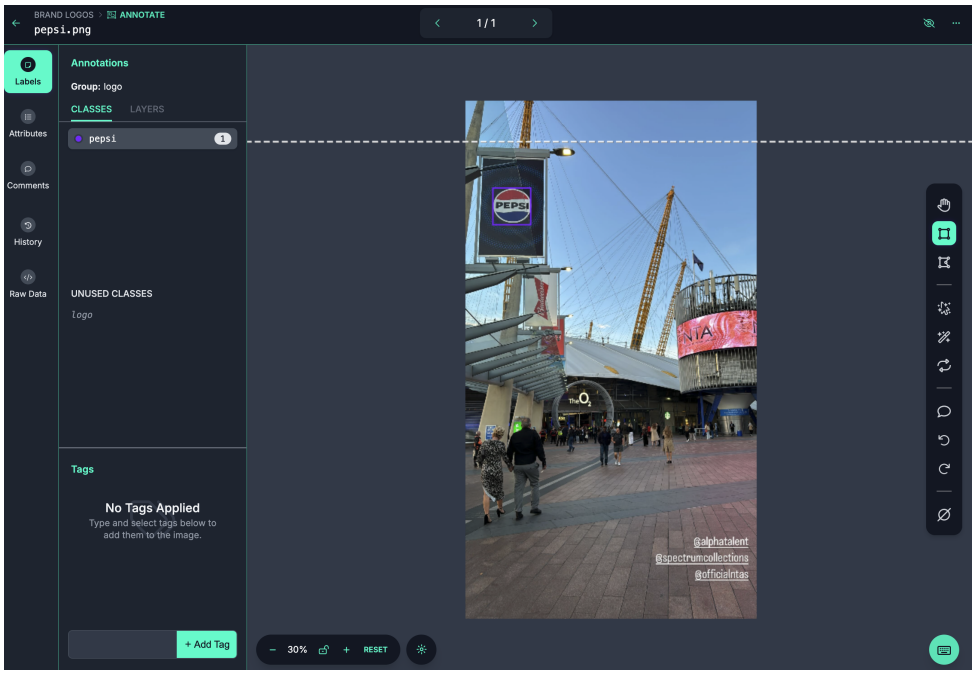


Figure 4.7: Roboflow labelling process.

or adjusting brightness. This step increases the diversity of training data, improving the model's ability to generalise across unseen images.

After the dataset is preprocessed and augmented, it is exported in the YOLOv8 format. The export process can be done via platforms like Roboflow, which provide the necessary format for seamless integration with YOLOv8 training. Figure 4.8 shows an example of the code used to download the dataset from Roboflow into the YOLOv8 format for Google Colab.

Next step involves installing and configuring the necessary dependencies, including Ultralytics and Roboflow, in the environment. This ensures that the tools required for training the YOLOv8 model are properly installed and that the system is ready to begin the training process. Additionally, the environment setup includes verifying the availability of GPU resources, which significantly accelerates the training process (see Figure 4.9).

Once the environment is configured, the next step is to download the labelled dataset. The dataset, which contains images of logos along with their bounding box annotations, is imported into the working environment (see Figure 4.10). Roboflow simplifies this process by providing tools to easily download the dataset in the appropriate format for YOLOv8.

After the dataset is downloaded, the YOLOv8 model is trained using the pre-trained weights provided by Ultralytics. During this step, the pre-trained model is fine-tuned to adapt specifically for logo detection. The training process typically runs for several epochs, with the goal of achieving the desired accuracy (see Figure 4.11). Throughout training, the model learns to identify logos in images by adjusting its weights based on the labelled data.

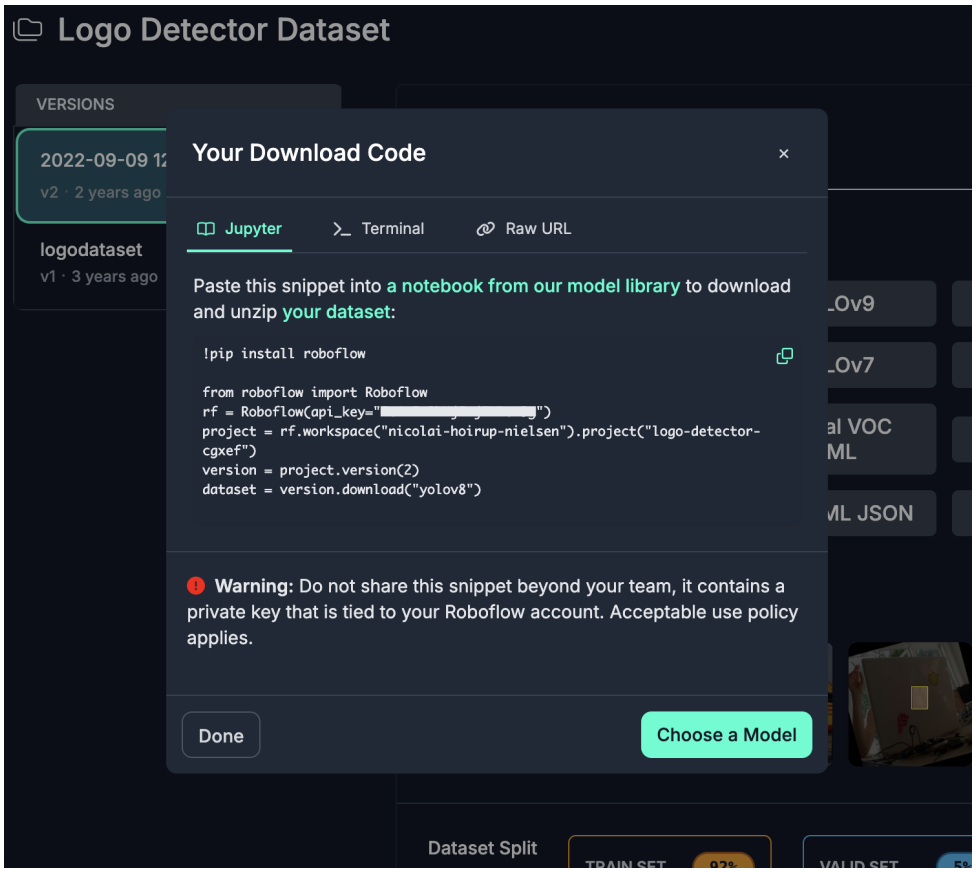


Figure 4.8: Downloading the dataset from Roboflow.

4.3 Speech Recognition with Whisper.AI

Whisper.AI is a state-of-the-art model designed to convert spoken language into text. It supports over 100 languages and offers an option to translate speech into English if required. Whisper can handle diverse accents, dialects, and noisy environments, making it highly effective for transcribing conversations, media files, and live audio streams. In this application, the `whisper-service` is integrated to transcribe audio or video content, allowing users to specify the languages via the frontend for enhanced recognition accuracy.

Whisper provides a range of model sizes that offer different trade-offs between speed, memory consumption, and accuracy. The table below (Table 4.4) summaries the available models, including their parameters, memory requirements, and relative speed during inference:

Table 4.4: Available Whisper Models and Their Specifications

Size	Parameters	English-only Model	Multilingual Model	Required VRAM	Relative Speed
tiny	39 M	`tiny.en`	`tiny`	1 GB	10x
base	74 M	`base.en`	`base`	1 GB	7x
small	244 M	`small.en`	`small`	2 GB	4x
medium	769 M	`medium.en`	`medium`	5 GB	2x
large	1550 M	N/A	`large`	10 GB	1x
turbo	809 M	N/A	`turbo`	6 GB	8x

The ‘.en‘ models are optimised for English-only transcription tasks and tend to perform better than their multilingual counterparts, especially for the `tiny.en` and `base.en` mod-

```
[ ] from ultralytics import YOLO
    import os
    from IPython import display
    display.clear_output()
    !yolo checks

Ultralytics 8.3.7 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 36.3/112.6 GB disk)

OS                  Linux-6.1.85+-x86_64-with-glibc2.35
Environment         Colab
Python              3.10.12
Install             pip
RAM                 12.67 GB
Disk                36.3/112.6 GB
CPU                 Intel Xeon 2.00GHz
CPU count           2
GPU                 Tesla T4, 15102MiB
GPU count           1
CUDA                12.1

numpy               ✅ 1.26.4>=1.23.0
matplotlib          ✅ 3.7.1>=3.3.0
opencv-python       ✅ 4.10.0.84>=4.6.0
pillow              ✅ 10.4.0>=7.1.2
pyyaml              ✅ 6.0.2>=5.3.1
requests            ✅ 2.32.3>=2.23.0
scipy               ✅ 1.13.1>=1.4.1
torch               ✅ 2.4.1+cu121>=1.8.0
torchvision         ✅ 0.19.1+cu121>=0.9.0
tqdm                ✅ 4.66.5>=4.64.0
psutil              ✅ 5.9.5
py-cpuinfo          ✅ 9.0.0
pandas              ✅ 2.2.2>=1.1.4
seaborn             ✅ 0.13.1>=0.11.0
ultralytics-thop    ✅ 2.0.9>=2.0.0
torch               ✅ 2.4.1+cu121!=2.4.0,>=1.8.0; sys_platform == "win32"
```

Figure 4.9: Environment setup showing installed dependencies and available GPU.

```
[ ] from roboflow import Roboflow
    rf = Roboflow(api_key="your-api-key")
    project = rf.workspace("christwork").project("logo-detection-e5fp3")
    version = project.version(3)
    dataset = version.download("yolov8")

loading Roboflow workspace...
loading Roboflow project...
Dependency ultralytics==8.0.196 is required but found version=8.3.7, to fix: `pip install ultralytics==8.0.196`
Downloading Dataset Version Zip in Brand-Logo-recognition---Yolov8-1 to yolov8:: 100%|██████████| 28322/28322 [00:03<00:00, 8935.58it/s]

Extracting Dataset Version Zip to Brand-Logo-recognition---Yolov8-1 in yolov8:: 100%|██████████| 1018/1018 [00:00<00:00, 4015.02it/s]
```

Figure 4.10: Downloading the dataset from Roboflow to the working environment.

els. For tasks requiring transcription of multiple languages, Whisper offers multilingual models that maintain strong accuracy across various languages, with minimal degradation in speed. The `turbo` model is an optimised version of the large model, offering faster transcription with minimal loss in accuracy.

Message Structure for Whisper Service

In the system architecture, the `whisper-service` processes incoming media files after they are uploaded to the system. The service is triggered by a message from the `coordinator-service`, which contains details about the media file and the language to be recognised. The message structure received by the `whisper-service` is shown in Listing 4.6.

```
1 {
2     "item_id": "0dbb7728-d60f-459f-a270-ea96b4440d2c",
3     "video_path": "videos/0dbb7728-d60f-459f-a270-ea96b4440d2c/video.mp4",
4     "languages": ["en"]
5 }
```

Listing 4.6: Message for Whisper Service

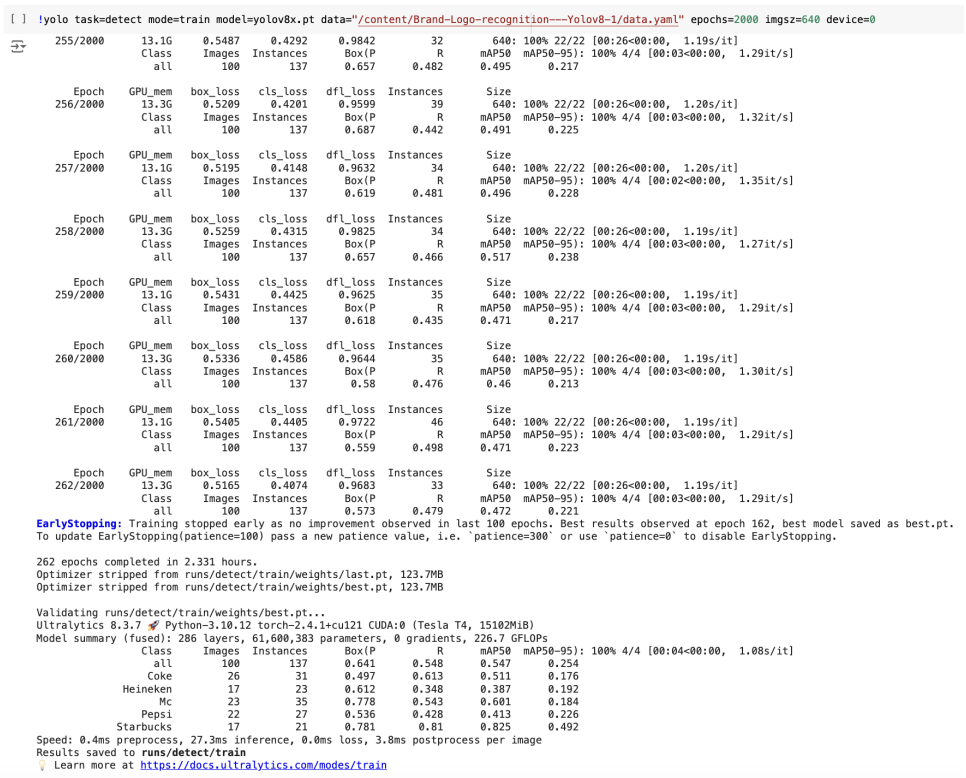


Figure 4.11: Training the YOLOv8 model with logo detection dataset.

The code in Listing 4.7 shows an example of how Whisper.AI tiny model can be used for speech recognition in Python. The language is also been specified for better accuracy.

```
1 import whisper
2
3 model = whisper.load_model("tiny")
4 result = model.transcribe("audio.mp3", language="pt")
5
6 print(result["text"])
```

Listing 4.7: Using Whisper.AI for Speech Recognition

Expected Results for Whisper Service

Once the audio is processed, Whisper returns a detailed transcription result, which includes the text, confidence scores, timing information, and other metadata. An example of the expected output from the `whisper-service` is shown in Listing 4.8, which includes key elements such as the transcribed text, start and end times for each segment, and additional speech-related metrics.

```
{
  "whisper_result": [
    {
      "id": 0,
      "seek": 0,
      "start": 0,
      "end": 4.48,
      "text": "We are family, close net, always together from weekend football games...",
      "tokens": [ ... ],
      "temperature": 0,
      "avg_logprob": -0.36877460344463375,
      "compression_ratio": 1.6904024767801857,
      "no_speech_prob": 0.04385824501514435
    },
    ...
  ]
}
```

Listing 4.8: Whisper Service Output Example

The expected output, as shown above, includes detailed metadata for each segment of transcribed speech, such as the time intervals (`start` and `end`), the transcribed text, and probability metrics such as `no_speech_prob`, which indicates the likelihood of silence during the segment. This structured output allows for further analysis or integration with downstream services.

4.4 Sentiment Analysis with XLM-RobBERTa

For sentiment analysis, the application uses the cardiffnlp/twitter-xlm-roberta-base-sentiment-multilingual model, which is particularly suited for analysing sentiments across multiple languages. XLM-RoBERTa is fine-tuned on Twitter data, making it highly effective for short-form texts like tweets or social media posts commonly produced by influencers. This model predicts three sentiment labels: positive, negative, and neutral, with a high degree of accuracy across different languages [75].

The code below demonstrates how the sentiment analysis service integrates with this model to process text segments transcribed from audio content using Whisper.AI. The sentiment service pulls messages from RabbitMQ, processes them, and sends the results to the `result-service`, which consequently stores them in the database.

```
from transformers import pipeline

# Initialize the XLM-Roberta-based sentiment analysis model
sentiment_analyzer = pipeline("sentiment-analysis", model="cardiffnlp/twitter-xlm-roberta-base-sentiment-multilingual")

def analyze_sentiment(text):
    """Analyze sentiment using the XLM-Roberta model."""
    result = sentiment_analyzer(text)
    return result

# Example usage
sentiment_result = analyze_sentiment("This is amazing!")
print(sentiment_result)
```

Listing 4.9: Using XLM-Roberta for Sentiment Analysis

Message Structure for Sentiment Service

The `sentiment-service` processes messages containing the transcription result from Whisper.AI and outputs sentiment predictions for each text segment. The structure of the message sent to the `sentiment-service` is shown below in Listing 4.10, where the `whisper_result` contains the transcribed text and related metadata from Whisper.

```
{
    "item_id": "0dbb7728-d60f-459f-a270-ea96b4440d2c",
    "whisper_result": [
        {
            "id": 0,
            "seek": 0,
            "start": 0,
            "end": 4.48,
            "text": "We are family, close net, always together from weekend football games...",
            "tokens": [ ... ],
            "temperature": 0,
            "avg_logprob": -0.36877460344463375,
            "compression_ratio": 1.6904024767801857,
            "no_speech_prob": 0.04385824501514435
        }
    ]
}
```

Listing 4.10: Message for Sentiment Service

Expected Results

After processing the text, the `sentiment-service` outputs the sentiment prediction for each text segment. The expected result contains the transcribed segment, a sentiment label (positive, negative, or neutral), and a confidence score. An example of the expected output from the `sentiment-service` is shown in Listing 4.11.

```
{
  "sentiment_result": [
    {
      "segment_text": "We are family, close net, always together from weekend football games...",
      "sentiment": {
        "label": "POSITIVE",
        "score": 0.9996004700660706
      }
    }
  ]
}
```

Listing 4.11: Sentiment Service Output Example

The output includes the following key elements, as shown in Table 4.5:

This structure enables efficient storage and retrieval of both the transcription and the associated sentiment analysis results.

4.5 Optical Character Recognition with EasyOCR

For optical character recognition (OCR), the application utilises EasyOCR, a deep learning-based OCR library that is flexible and supports multiple languages. The `ocr-service` processes images and video frames to extract text and sends the results to the `result-service` for storage. To enhance OCR accuracy and performance, preprocessing techniques such as resizing and denoising are applied to the images. EasyOCR supports multiple languages, which is especially beneficial when handling multilingual content.

Table 4.5: Key Elements of Sentiment Analysis Output

Field	Description
`segment_text`	The transcribed text from Whisper, which is the input for sentiment analysis.
`sentiment`	An object containing the sentiment prediction details, including:
`label`	The predicted sentiment label, such as `"POSITIVE"`, `"NEGATIVE"`, or `"NEUTRAL"`.
`score`	The confidence score of the sentiment prediction, indicating how confident the model is in its prediction.

The code below demonstrates how the OCR service is configured to process a single image, including image preprocessing and text extraction using EasyOCR.

```
import easyocr
import cv2

# Initialize EasyOCR Reader with default languages
ocr_reader = easyocr.Reader(['en'])

def preprocess_image(image_path):
    """Preprocess the given image to optimize OCR performance."""
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    image = cv2.resize(image, (1024, 1024)) # Resize for consistent processing
    image = cv2.medianBlur(image, ksize=1) # Denoise the image
    return image

def perform_ocr(image_path):
    """Run OCR on the given image."""
    image = preprocess_image(image_path)
    result = ocr_reader.readtext(image)
    return result

# Example usage
ocr_result = perform_ocr('image_path.png')
for entry in ocr_result:
    print(entry[1]) # Extracted text
```

Listing 4.12: Using EasyOCR for OCR

In this example, the image is preprocessed by converting it to grayscale, resizing it to a consistent size, and applying denoising before running EasyOCR. This ensures that the model performs more accurately and avoids issues like excessive memory consumption when processing high-resolution images in the Docker container.

Message Structure for OCR Service

The `ocr-service` processes both images and video frames, and the message structure differs slightly based on the type of media being processed. The message contains the `item_id`, the path to the media, and the languages for OCR.

For video frame processing, the message structure is shown in Listing 4.13.

```
{
    "item_id": "0dbb7728-d60f-459f-a270-ea96b4440d2c",
    "frames_path": "videos/0dbb7728-d60f-459f-a270-ea96b4440d2c/frames/",
    "languages": ["en", "es", "pt"]
}
```

Listing 4.13: Message for OCR Service - Video Frames

For image processing, the message structure is shown in Listing 4.14.

```
{
    "item_id": "14f918f5-eb5a-43af-8014-dfa88f452dff",
    "image_path": "images/14f918f5-eb5a-43af-8014-dfa88f452dff/image.png",
    "languages": ["en", "es", "pt"]
}
```

Listing 4.14: Message for OCR Service - Image

Expected Results

After processing the image or video frames, the OCR service outputs a result containing the recognised text. The expected result structure includes the extracted text for each segment, along with the bounding boxes for the detected text regions and confidence. An example of the expected output from the `ocr-service` is shown in Listing 4.15.

```
{
    "ocr_result": [
        {
            "text": "amazon co.uk",
            "confidence": 0.9861529218249994,
            "bounding_box": [
                [10, 20], [300, 20], [300, 60], [10, 60]
            ]
        },
        {
            "text": "Christina",
            "confidence": 0.4042236039445303,
            "bounding_box": [
                [20, 80], [400, 80], [400, 120], [20, 120]
            ]
        }
    ]
}
```

Listing 4.15: OCR Service Output Example

The output contains the following key elements:

- **text**: The extracted text from the image or video frames.
- **confidence**: The confidence value for specific text.
- **bounding_box**: The coordinates of the bounding box that surrounds the detected text in the image or frame.

This structured result allows for easy storage and integration with downstream services.

4.6 Cloud Infrastructure and Scalability

This project is designed with scalability and cloud-native deployment in mind, utilising AWS for file storage, computation, and management of system components. The core services are containerised using Docker, ensuring modularity and isolated environments for each service. This containerisation enables horizontal scaling by simply adding more containers or nodes to handle increased workloads. New services can be integrated into the system by building additional containers, allowing the architecture to remain flexible, scalable, and adaptable to new use cases or AI models.

4.6.1 Leveraging AWS for Scalability

In a production environment, AWS offers a suite of services that make it easier to deploy, manage, and scale the application. The key AWS services that can be integrated into the architecture include:

- **AWS S3**: Used as the primary storage solution for uploaded media files (videos, images, and frames). AWS S3 provides high availability and durability, ensuring that files are securely stored and easily retrievable. The S3 bucket structure, shown in Figure 4.12, organises files into distinct categories (videos, images, frames) for more efficient processing. Furthermore, S3 integrates seamlessly with AWS Lambda and S3 event triggers, enabling automated workflows, such as invoking the `whisper-service` or `yolo-service` when new files are uploaded.
- **AWS Lambda**: To further improve scalability, AWS Lambda can be used to run serverless functions that handle individual processing tasks (e.g., triggering the OCR service when an image is uploaded). By using AWS Lambda, the system can automatically scale up or down based on the workload, eliminating the need to manage infrastructure manually.
- **Amazon ECS or EKS**: These managed container orchestration services enable the automatic deployment and scaling of Docker containers. With Amazon ECS (Elastic Container Service) or EKS (Elastic Kubernetes Service), each microservice (e.g., `whisper-service`, `yolo-service`) runs inside its own container, and the system can horizontally scale by deploying additional containers to handle increased traffic. These services also provide health checks, logging, and monitoring, ensuring that each microservice runs efficiently.
- **Amazon SQS**: For task queue management, AWS SQS (Simple Queue Service) can be used to queue processing tasks (e.g., video frames for object detection or speech-to-text conversion). By integrating SQS with the microservices architecture, tasks can be distributed evenly across containers, improving load balancing and resilience.
- **AWS DynamoDB or MongoDB Atlas**: While the current implementation uses MongoDB for storing processed results, DynamoDB can be considered as an alternative for a fully managed, highly available NoSQL database solution. DynamoDB scales automatically to accommodate growing datasets, and offers built-in backup and encryption, enhancing security and reliability. MongoDB Atlas could also be integrated into AWS infrastructure for a managed MongoDB solution.

4.6.2 Scaling Through Containerisation and Serverless Architectures

The use of containerisation ensures that the system can easily scale by adding more containers for each microservice, enabling parallel processing of multimedia content. New services or models can be added to the system without significant architectural changes. For example, if a new AI model for video summarising is required, a new Docker container can be created, and this service can be seamlessly integrated with the existing infrastructure by routing the relevant tasks to this container.

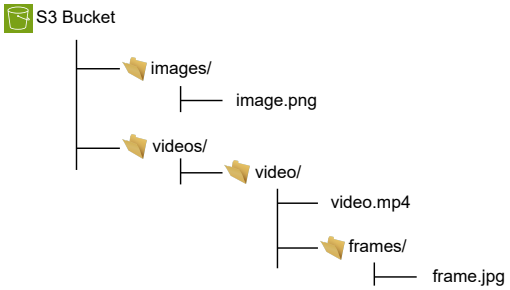


Figure 4.12: S3 Bucket file structure diagram.

Figure 4.13 shows how the system can be extended with additional services (service 1, service 2) to meet growing business requirements or new content analysis needs. By adopting an event-driven architecture using AWS Lambda and SQS, new services can be added dynamically, triggered by user actions or file uploads.

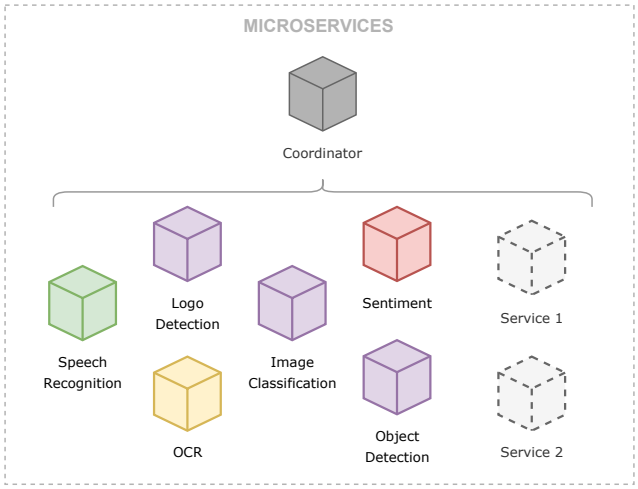


Figure 4.13: Example of adding new services (e.g., video summarising or children detection) to the application.

4.6.3 Future Scalability Improvements

As the system grows and handles more media files, the following enhancements can be considered to further improve scalability and performance:

- **Infrastructure as Code (IaC)**: Tools like AWS CloudFormation or Terraform can be used to define and manage the AWS infrastructure in a declarative format. This ensures consistent environments across development, testing, and production and makes it easy to replicate the infrastructure or roll out updates.
- **Elastic Load Balancing (ELB)**: Integrating ELB ensures that incoming requests are distributed across multiple containers and AWS Lambda functions, improving fault tolerance and ensuring consistent performance as the load increases.

- **Auto-Scaling Groups (ASG)**: For container-based services running on Amazon ECS or EKS, auto-scaling groups can be configured to automatically launch new EC2 instances when CPU or memory usage exceeds predefined thresholds. This guarantees that the system can handle traffic spikes without manual intervention.
- **AWS Fargate**: For truly serverless container management, AWS Fargate can be employed to run containers without provisioning or managing servers. This allows the system to scale seamlessly and eliminates the need to manage container hosts.
- **Advanced Data Analytics**: Leveraging AWS services such as Amazon Athena and Amazon QuickSight could provide real-time insights into how the system is performing, which media types are being processed the most, and the overall efficiency of the AI models in production. These insights can inform future optimizations.

This cloud-based architecture, combining Docker containers, AWS services, and MongoDB for data storage, ensures the application is both flexible and scalable. The integration of AWS Lambda, S3, and ECS/EKS provides the infrastructure needed to handle an increasing number of media files and growing user demands without sacrificing performance or reliability. By adopting this modular and scalable approach, the application is well-prepared to integrate new AI models and services, as well as scale effortlessly to meet future demands.

4.7 Application User Interface

To test the application, a user-friendly interface was built using Streamlit, with the `streamlit-service` responsible for its operation. The interface provides an intuitive environment for uploading files and controlling key parameters. On the left-hand side, users can find a sidebar that allows for file uploads and input of several values such as the services to run—`yolo`, `yolo-cls`, `yolo`-logo, `whisper` or `ocr`. When `whisper-service` is selected, `sentiment-service` for sentiment analysis is automatically included since it depends on the whisper results.

If the uploaded file is a video, the interface additionally requests the frame interval (in seconds) for splitting the video into frames. This option is only available for video uploads, ensuring that the frames are processed by the corresponding services such as YOLO or OCR. Additionally, the user can specify the languages identified in the file to optimise the performance of the selected models. Fig 4.14 illustrates the file upload process.

Once a file is uploaded, a new entry appears in a table view that lists the uploaded files, visible in the center of the main page. This table provides users with an overview of the upload status, the progress of each selected service, and key metadata of the uploaded file. Fig 4.15 illustrates the uploaded file with ongoing processing.

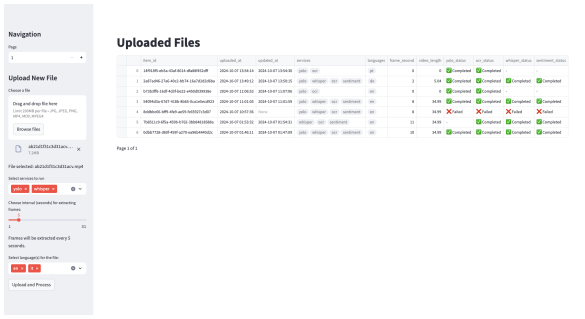


Figure 4.14: Interface for uploading new files with service options and file metadata displayed in the table view.

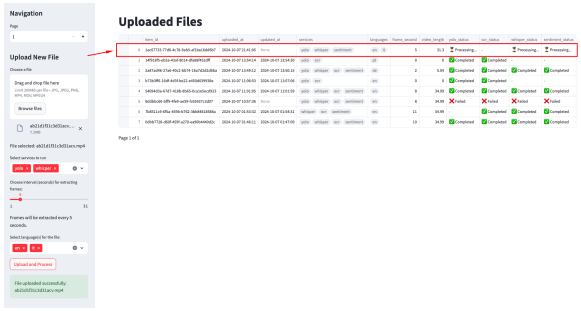


Figure 4.15: Uploaded file entry showing ongoing processing with the status of selected services.

To review the combined model results, the user must select a specific entry from the table, after which the results will be displayed. If the file is a video, a timeline view appears, showing frames extracted at regular intervals. By clicking on any frame, the user can review the results for that specific frame. Fig 4.16 demonstrates the timeline interaction for video files.

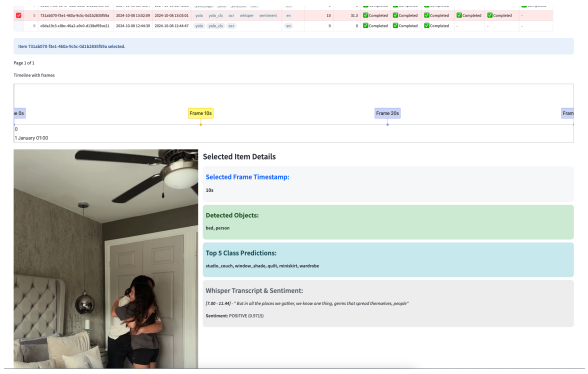


Figure 4.16: Display of selected video entry with corresponding results and a visual timeline of the file's processing stages.

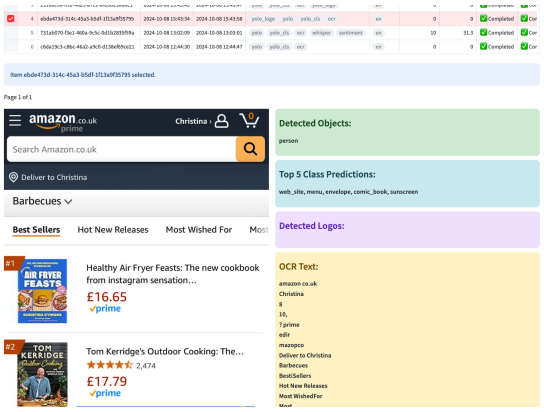


Figure 4.17: Display of selected image entry with corresponding results.

For image uploads, the original image is presented alongside the results from `yolo-service`, `yolo-cls`, `ocr-service` and `yolo-logo-service`. These services currently handle image processing, but additional models can be integrated. Fig 4.17 shows an example of how results are displayed for images.

Chapter 5

Performance Evaluation and Results

5.1 Object Detection Performance

Several tests were conducted to evaluate the performance of YOLOv8 models and to determine their suitability for application in the context of this thesis. The goal was to assess how different YOLOv8 variants—ranging from lightweight models like YOLOv8n to more complex models like YOLOv8x—performed under controlled conditions by processing individual frames from a YouTube video.

In Table 5.1, the models' operational efficiency is outlined, highlighting key metrics such as speed and detection accuracy. The *yolov8n* model, shown in Figure 5.1, excels in fast-paced environments where speed is critical. Meanwhile, the *yolov8x* model, depicted in Figure 5.2, showcases its strength in more complex tasks where higher accuracy is essential. These results not only guided the development of the YOLO-based services in this thesis but also demonstrated the adaptability of YOLO models for various use cases, from real-time detection to detailed content analysis. The tests allowed for a deeper understanding of how to leverage these models in practical applications, further validating their effectiveness for the project's objectives.

Table 5.1: Comparison of performance metrics across different YOLOv8 models.

Model	Time Elapsed (s)	Preprocess (ms)	Inference (ms)	Postprocess (ms)
yolov8n	0.53	10.1	82.8	10.2
yolov8s	0.59	3.8	98.5	9.2
yolov8m	0.72	4.0	160.1	8.3
yolov8l	0.84	3.1	222.3	1.8
yolov8x	1.05	1.8	294.7	1.8

5.2 Speech Recognition Accuracy

Our evaluation of Whisper.AI's transcription accuracy included several trials using English-language YouTube videos. The aim was to compare Whisper.AI's capabilities against YouTube's native transcription service. For these tests, we used the z '.en' model, which is optimised for English transcription. Whisper.AI offers various models with different

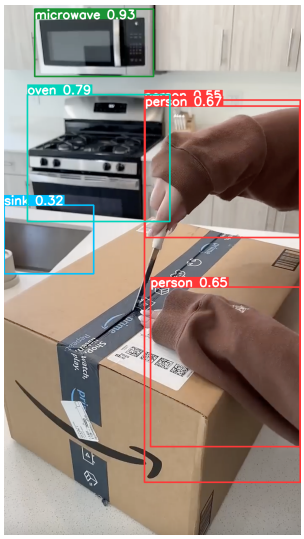


Figure 5.1: Detection using YOLOv8n.

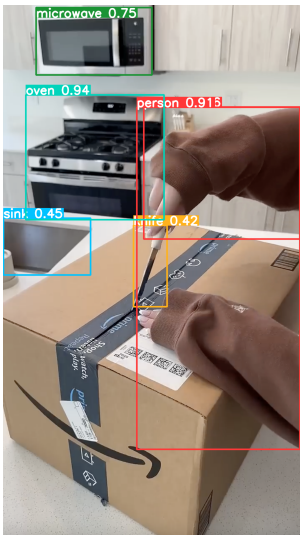


Figure 5.2: Detection using YOLOv8x.

configurations to cater to specific requirements, making it highly adaptable for transcription tasks [8].

The process involved running the media file through a chosen model to generate an initial transcription for analysis. We then compared this output to a reference transcript produced by a language specialist, focusing on details such as compound word spacing or the use of hyphens. While such nuances might be disregarded in a broader analysis, leading to a lower word error rate, our focus was on precise accuracy.

The Table 5.2 below compares the transcription results across multiple dimensions, including the total word count, substitutions, deletions, insertions, overall errors, correctly transcribed words, word error rate, and model execution time. Both mp3 and mp4 formats were tested, allowing us to assess the models' performance across different media types.

In our detailed analysis of a YouTube short, we found that YouTube's native transcription closely matched Whisper.AI's results when using the *base.en* model, as outlined in Table 5.2. The *medium.en* model, applied to an mp3 file, achieved near-perfect accuracy, with only minor issues such as extra spaces between words or slight punctuation errors. This suggests that both the file format and model choice influence the quality and speed of the transcription. Notably, Whisper.AI has the advantage of including punctuation, which significantly improves the readability of its transcripts. Additionally, pronunciation and dialect variations can also affect the accuracy, further highlighting Whisper.AI's robustness and its added benefit of multilingual transcription capabilities.

The ability to obtain timestamps for each transcription segment is a key feature, offering precise references to the corresponding moments in the video where transcriptions occur, as illustrated in List 5.1. This timestamping feature allows users to map content back to specific points in a video, making it easier to navigate and analyse the video's progression.

Moreover, by integrating advanced models for sentiment analysis on each transcribed segment, as discussed in Section 5.3, we can gain a deeper understanding of the emotional tone of the content. This approach enhances the overall data analysis, enabling the

Table 5.2: Transcription accuracy comparison of various models and methods for a specific YouTube video.

Source	Model	Words	Sub.	Del.	Ins.	Errors	Correct	WER %	Time (s)
Original	N/A	74	0	0	0	0	74	0.00	N/A
YouTube	N/A	74	3	1	0	4	70	5.41	N/A
Whisper (mp3)	tiny.en	74	4	1	0	5	69	6.76	1.285
Whisper (mp4)	tiny.en	74	4	1	0	5	69	6.76	1.194
Whisper (mp3)	base.en	74	3	1	0	4	70	5.41	2.234
Whisper (mp4)	base.en	74	3	1	0	4	70	5.41	2.220
Whisper (mp3)	small.en	74	3	1	0	4	70	5.41	5.984
Whisper (mp4)	small.en	74	4	1	0	5	69	6.76	5.417
Whisper (mp3)	medium.en	74	2	1	0	3	71	4.05	15.907
Whisper (mp4)	medium.en	74	3	1	0	4	70	5.41	15.178

extraction of sentiment-related insights that are contextually rich and actionable. An example of applying sentiment analysis to such segments can be found in the work of [63], where the emotional undertones of video content are explored, further showcasing the value of sentiment analysis in multimedia processing.

```
{
    "id": 0,
    "seek": 0,
    "start": 0.0,
    "end": 1.76,
    "text": " You love this iPhone trick!",
    "tokens": [50363, 921, 1842, 428, 7133, 6908, 0, 50451],
    "temperature": 0.0,
    "avg_logprob": −0.25819649015154156,
    "compression_ratio": 1.568,
    "no_speech_prob": 0.03296058252453804
}
```

Listing 5.1: Example of a Whisper.AI transcription segment

5.3 Sentiment Analysis Model Comparison

In this section, we evaluate three sentiment analysis models: VADER, BERT Multilingual, and XLM-RoBERTa. Each model was tested on several influencer-style sentences to determine which is best suited for analysing content produced by influencers. The sentences cover positive, negative, and neutral sentiments commonly found in influencer posts. Below are brief introductions to the models and the results of our analysis.

VADER is a rule-based sentiment analysis model that is optimised for social media text. It uses a combination of lexical features and heuristics to determine sentiment, making it fast and computationally lightweight. The model provides four scores: negative, neutral, positive, and a compound score that represents the overall sentiment. VADER is highly effective for general sentiment analysis, especially in short, informal texts like tweets or influencer posts. However, its rule-based nature may limit its ability to capture more nuanced emotions or context-dependent sentiments in complex content [40].

The second model, `bert-base-multilingual-uncased-sentiment`, is based on the BERT architecture and has been fine-tuned for sentiment analysis on product reviews in six languages: English, Dutch, German, French, Spanish, and Italian. It predicts sentiment on a 5-star rating scale, ranging from 1 (very negative) to 5 (very positive). This model was

fine-tuned on large sets of product reviews, making it suitable for multilingual environments and related sentiment analysis tasks. However, its focus on product reviews may limit its applicability to other contexts, such as social media posts where the language might be less structured or more informal [76].

The final model, `cardiffnlp/twitter-xlm-roberta-base-sentiment-multilingual`, is a fine-tuned version of the XLM-RoBERTa model, trained specifically for sentiment analysis on multilingual tweets. It predicts three sentiment labels: positive, negative, and neutral. XLM-RoBERTa excels in handling multilingual content, making it especially useful in analysing influencer posts that may contain multiple languages. Its fine-tuning on social media data like tweets ensures that it is well-suited for short-form text, but it may also struggle with longer or more contextually rich content [75].

We tested the models on the following influencer-related sentences:

- "I absolutely love how this product has transformed my skincare routine!" (positive)
- "This new workout routine is amazing, I feel stronger and more energised every day." (positive)
- "Honestly, this collaboration feels like a cash grab, it's really disappointing." (negative)
- "The product didn't work at all for me, complete waste of money." (negative)
- "I'm partnering with this brand for their new launch, stay tuned for updates." (neutral)
- "Here's the link to my latest blog post, check it out when you have time." (neutral)

The results of these tests, including sentiment predictions and processing times, are summarised in the table below:

Model	**Sentence**	**Sentiment Score**	**Time Elapsed (ms)**
VADER	POSITIVE	'neg': 0.0, 'neu': 0.653, 'pos': 0.347, 'compound': 0.6989	0.124
VADER	POSITIVE	'neg': 0.0, 'neu': 0.5, 'pos': 0.5, 'compound': 0.8748	0.126
VADER	NEGATIVE	'neg': 0.218, 'neu': 0.438, 'pos': 0.344, 'compound': 0.2516	0.098
VADER	NEGATIVE	'neg': 0.203, 'neu': 0.797, 'pos': 0.0, 'compound': -0.4215	0.089
VADER	NEUTRAL	'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0	0.080
VADER	NEUTRAL	'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0	0.093
BERT Multilingual	POSITIVE	'label': '5 stars', 'score': 0.936	67.25
BERT Multilingual	POSITIVE	'label': '5 stars', 'score': 0.923	38.65
BERT Multilingual	NEGATIVE	'label': '1 star', 'score': 0.479	36.13
BERT Multilingual	NEGATIVE	'label': '1 star', 'score': 0.884	38.39
BERT Multilingual	NEUTRAL	'label': '5 stars', 'score': 0.612	38.37
BERT Multilingual	NEUTRAL	'label': '5 stars', 'score': 0.441	38.28
XLM-RoBERTa	POSITIVE	'label': 'positive', 'score': 0.989	75.48
XLM-RoBERTa	POSITIVE	'label': 'positive', 'score': 0.990	61.97
XLM-RoBERTa	NEGATIVE	'label': 'negative', 'score': 0.983	46.76
XLM-RoBERTa	NEGATIVE	'label': 'negative', 'score': 0.951	50.22
XLM-RoBERTa	NEUTRAL	'label': 'positive', 'score': 0.953	51.07
XLM-RoBERTa	NEUTRAL	'label': 'positive', 'score': 0.582	360.60

Table 5.3: Sentiment Analysis Results for Influencer Content Sentences

From these results, we can conclude that VADER is the fastest model by far, with sub-millisecond execution times, making it suitable for real-time applications. However, since VADER is an English-only model, it may not perform well in multilingual contexts,

which is a limitation for analysing influencer content from diverse regions. Additionally, VADER's rule-based nature may limit its ability to capture subtle or context-dependent sentiments, which are often present in influencer-generated content.

BERT Multilingual performs well with clear positive or negative sentiments but struggles with neutral sentiments, likely due to its training focus on product reviews. Although this model is well-suited for multilingual environments, it also uses a star rating system (from 1 to 5 stars), which may not be the best fit for the nuanced analysis required in influencer content. The star-based scale could oversimplify the complex range of sentiments typically found in social media and influencer communications.

XLM-RoBERTa shows a high level of accuracy across positive, negative, and neutral sentiments, though there is a tendency to slightly overrate neutral statements as positive. However, this model still handles sentiment analysis with fewer critical errors compared to BERT Multilingual, making it the most reliable model for diverse influencer content. Its ability to process multiple languages accurately is also a significant advantage for analysing influencer content across various cultural and linguistic contexts.

Given the accuracy and flexibility of XLM-RoBERTa, it appears to be the best fit for analysing influencer content, where context and language diversity are critical factors.

5.4 OCR Performance Evaluation

In this section, we present the results of Optical Character Recognition (OCR) tests conducted on three images. The tests were carried out with and without preprocessing to compare the performance and accuracy of the OCR model. We filtered out results with a confidence score below 0.5 to eliminate noise and improve accuracy. The table below summarises the Word Error Rate (WER), number of found characters, and other relevant metrics.

Additionally, during the execution of the OCR service, it was observed that without preprocessing, the Docker container would frequently crash due to excessive memory consumption when running on Central Processing Unit (CPU). Preprocessing techniques such as resizing and denoising were applied to prevent these issues. While preprocessing slightly affected the execution time, it stabilised the system, allowing for smooth processing.

Figures 5.3 and 5.4 show Image 1 before and after preprocessing, respectively.

Image	**Preprocessing**	**Found Characters**	**WER (%)**	**Subs**	**Del**	**Ins**	**Elapsed Time (s)**
Image 1	Without	132	2.27%	0	1	0	8.61
Image 1	With	81	7.41%	3	2	1	7.93
Image 2	Without	69	7.25%	2	0	3	5.95
Image 2	With	62	6.45%	0	2	2	6.73
Image 3	Without	18	5.56%	0	0	1	6.78
Image 3	With	4	0.00%	0	0	0	7.10

Table 5.4: OCR Results: Comparison of Preprocessing on Found Characters, WER, and Elapsed Time

The table above provides a comparison of OCR results based on the number of characters found, WER, and elapsed time. In Image 1, preprocessing caused a drop in the number of

Figure 5.3: Image 1: Before Preprocessing

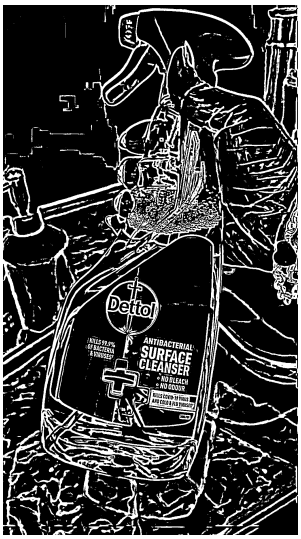


Figure 5.4: Image 1: After Preprocessing

found characters and increased the WER from 2.27% to 7.41%. This suggests that some characters may have been filtered out during preprocessing.

In Image 2, preprocessing improved the WER slightly, dropping from 7.25% to 6.45%, but it also resulted in a slight decrease in the number of characters found. The preprocessing also removed some noise from the image, leading to fewer insertions and substitutions.

For Image 3, preprocessing resulted in a significant improvement, reducing the WER to 0.00%, though fewer characters were found. This highlights the benefits of preprocessing for certain types of images, where noise may otherwise result in erroneous OCR outputs.

While preprocessing slightly improved the overall execution time in some cases, the most significant benefit was the stability it provided to the system. Further investigation could explore the use of a GPU to improve performance, as this would likely reduce processing times significantly.

5.5 Integrated Results on Real-World Content

The application was tested on various real-world videos and images sourced from influencer content across popular platforms such as YouTube, Instagram, and TikTok. These platforms were selected due to the high volume of visual, textual, and spoken content they feature, which makes them ideal for evaluating the system's multiple services. By combining object detection, logo identification, speech recognition, sentiment analysis, and OCR, the application provides valuable insights into how products and brands are being promoted by influencers.

The integration of these services allows for a deep analysis of multimedia content, detecting logos, products, spoken words, and sentiments in influencer posts. The following sections break down the results by platform, with examples for each showcasing how the system extracts and consolidates meaningful data.

5.5.1 Instagram Story Analysis

Instagram stories often contain rich visual content combined with text, emojis, and branded logos. We tested the system on three different Instagram stories, where influencers shared moments featuring products and promotional content.

Object Detection and Logo Identification: In each of the stories, the YOLOv8 model successfully identified multiple objects, such as cups, dining tables, and brand logos. In particular, the logo detection module highlighted brand logos like Pepsi, which provides quick insights into the promotional content featured within the stories.

OCR: Text Extraction from Visual Content: The OCR service accurately extracted text embedded in the stories, including user captions describing their experiences (e.g., a chest infection and the medications used). It also captured promotional elements such as timestamps and influencer commentary, which are often essential for understanding the full scope of influencer marketing.

Examples of Results

- **Instagram Story 1:**
  - **Detected Objects:** cup, dining table, book.
  - **Top Class Predictions:** coffee_mug, pencil_sharpener, water_bottle.
  - **OCR Extracted Text:** "Awake for half the night with this chest infection."

  Medications such as Amoxicillin and Strepsils were also detected in the scene.
- **Instagram Story 2:**
  - **Detected Objects:** person, dining table, sink.
  - **Top Class Predictions:** toilet_tissue, hand_blower, paper_towel.
  - **Whisper Transcript & Sentiment:** A positive sentiment was detected alongside the transcript "it's where we connect, laugh, and make memories."
- **Instagram Story 3:**
  - **Detected Logos:** Pepsi.
  - **Top Class Predictions:** basketball, horizontal_bar, volleyball.

  This story featured an influencer at a stadium, with logo detection accurately identifying the Pepsi logo in the background.

These results demonstrate the effectiveness of using object detection, logo identification, and OCR for analysing Instagram stories. Figure 5.8 illustrates the analysis for all three stories.

5.5.2 TikTok Analysis

TikTok videos often combine dynamic visual content with captions, hashtags, and brand logos to enhance engagement. We tested the system on three different TikTok videos, where influencers showcased products and promotions.

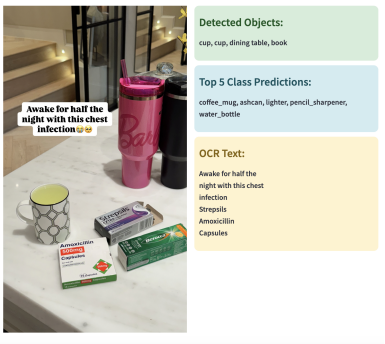


Figure 5.5: Instagram Story 1

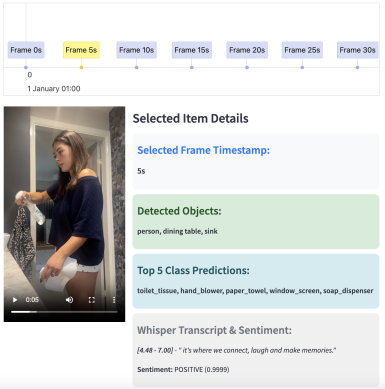


Figure 5.6: Instagram Story 2

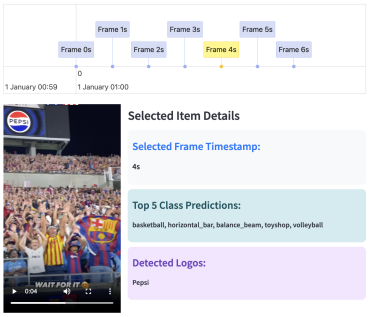


Figure 5.7: Instagram Story 3

Figure 5.8: Instagram Story Analysis: Health, Home, and Stadium content with detected objects, logos, and OCR results.

Object Detection and Logo Identification: In each TikTok video, the YOLOv8 model successfully identified objects such as bottles and people. Additionally, the logo detection module recognised prominent brand logos such as Coke and K18, providing valuable insights into the visual marketing content.

OCR: Text Extraction from Video Content: The OCR service extracted text from the video frames, including hashtags, captions, and brand names, which are critical for understanding the influencer's message and promotional details.

Examples of Results

- **TikTok Video 1:**
  - **Detected Objects:** bottle, person.
  - **Top Class Predictions:** pop_bottle, bottlecap, water_bottle, vending_machine.
  - **Detected Logos:** Coke.
- **TikTok Video 2:**
  - **Detected Objects:** person, bottle, potted plant, chair.
  - **Top Class Predictions:** barber_chair, soap_dispenser, washbasin.
  - **Whisper Transcript & Sentiment:** A positive sentiment was detected with the transcript "do at home so I kindly agreed with him and let him demonstrate."
  - **OCR Extracted Text:** K18.
- **TikTok Video 3:**
  - **Detected Objects:** person, bottle.
  - **Top Class Predictions:** pill_bottle, lotion, sunscreen.

- **Whisper Transcript & Sentiment:** Another positive sentiment was detected with the transcript "want to head and request it a blowout, a silk press and then a soft curl."
- **OCR Extracted Text:** K18, molecular repair.

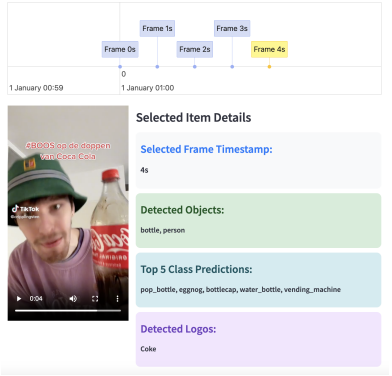


Figure 5.9: TikTok Video 1

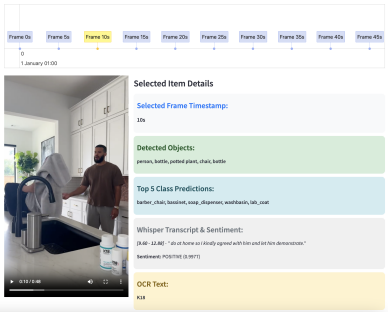


Figure 5.10: TikTok Video 2

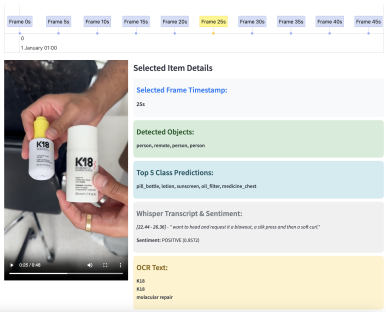


Figure 5.11: TikTok Video 3

Figure 5.12: TikTok Story Analysis: Detected objects, logos, and OCR results from three different TikTok videos.

These results demonstrate the efficiency of using object detection, logo recognition, and OCR for analysing TikTok videos. Figure 5.12 showcases the analysis of three TikTok videos with detected content.

5.5.3 YouTube Reel Analysis

YouTube Reels often feature short, engaging videos with embedded text, product showcases, and brand logos. We tested the system on two different YouTube Reels, where influencers promoted technology products.

Object Detection and Logo Identification: The YOLOv8 model identified objects such as people and remote controls in the videos. Additionally, the logo detection module successfully identified brands such as Apple, enhancing our understanding of the product promotions.

OCR: Text Extraction from Video Content: The OCR service extracted key text, including product descriptions, hashtags, and other on-screen text, which are essential for capturing the influencer's promotional message.

Examples of Results

- **YouTube Reel 1:**
  - **Detected Objects:** person, remote.
  - **Top Class Predictions:** cup, lighter, coffee_mug, ballpoint.
  - **Detected Logos:** Apple.
  - **OCR Extracted Text:** @THISISTECH, ODA.
- **YouTube Reel 2:**

- **Detected Objects:** person, tie.
- **Top Class Predictions:** Windsor_tie, suit, bow_tie.
- **Whisper Transcript & Sentiment:** A positive sentiment was detected with the transcript "So I would take the screen from the S-22 Ultra, it's incredible."
- **OCR Extracted Text:** SCREEN FROM THE, A4.

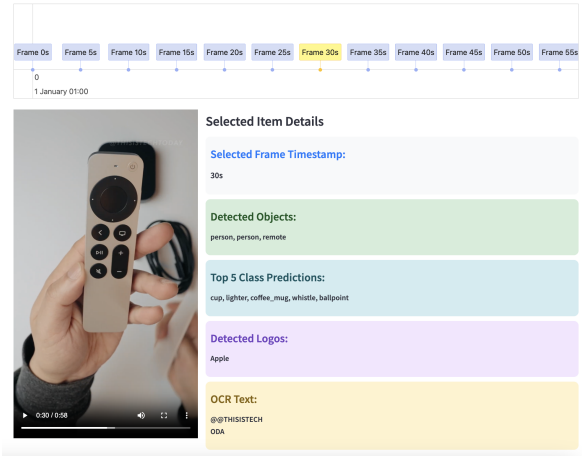


Figure 5.13: YouTube Reel 1

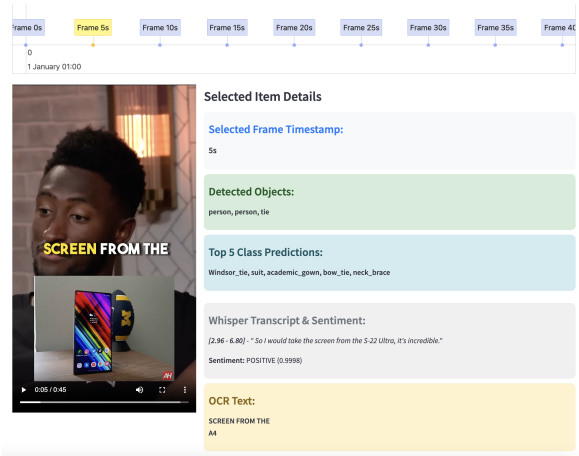


Figure 5.14: YouTube Reel 2

Figure 5.15: YouTube Reel Analysis: Detected objects, logos, and OCR results from two YouTube Reels.

These results demonstrate the effectiveness of using object detection, logo recognition, and OCR for analysing YouTube Reels. Figure 5.15 showcases the analysis of two YouTube Reels with detected content.

5.5.4 Critical Analysis

The integrated results from testing on real-world content demonstrate the system's effectiveness in providing a holistic view of influencer media across platforms such as Instagram, TikTok, and YouTube. The combination of object detection, logo identification, speech recognition, sentiment analysis, and OCR enables a multifaceted analysis, revealing valuable insights into product placements, influencer messaging, and user engagement.

One of the key strengths of the system is its ability to handle multiple types of data—visual, textual, and auditory—simultaneously. The integration of different AI models allows the system to perform complex content analysis with relatively high accuracy, as seen in the detailed results for each platform. The detection of objects and logos in influencer videos, coupled with speech transcriptions and sentiment analysis, provides a comprehensive understanding of the content's overall tone and intent. This can be particularly useful for brands or advertisers looking to monitor how their products are being presented in social media content.

However, there are certain limitations worth addressing. The object detection and logo identification tasks performed well under ideal conditions, but the accuracy declined in scenarios involving cluttered backgrounds, poor lighting, or low-resolution videos. For instance, in the TikTok and YouTube analyses, objects such as bottles and remote controls

were detected effectively, but other relevant items may have gone undetected due to environmental factors. The model could benefit from further fine-tuning on more diverse datasets to improve detection in more challenging conditions.

Another limitation is the reliance on Whisper.AI for speech recognition, which, while effective, can sometimes struggle with lower-quality audio or overlapping speech. The sentiment analysis component is highly dependent on the quality of the transcription, and errors in speech-to-text conversion could lead to inaccurate sentiment predictions. Additionally, the sentiment analysis is currently focused on text, which may miss important non-verbal cues conveyed through body language or facial expressions. Expanding the system to include facial emotion detection could significantly enhance the sentiment analysis, providing a more nuanced understanding of the influencer's emotions and intentions.

Moreover, while the system excels in extracting information from social media content, the sheer variety of influencer content across platforms requires constant model updates and retraining. Influencers use a wide array of visual styles, tones, and content structures, making it difficult for a one-size-fits-all model to perform optimally across all contexts. Continuous learning mechanisms and more specific fine-tuning per platform or content type would likely yield better results.

Finally, the system's scalability could also be a concern when processing large volumes of content. While the cloud-based infrastructure supports horizontal scaling, real-time analysis of high-volume content may still face latency issues, particularly with long-form video content or high-resolution imagery.

In conclusion, while the system demonstrates a powerful proof of concept, there are several areas where improvements could be made to enhance its accuracy, robustness, and scalability. By addressing the limitations in object detection, expanding sentiment analysis to include facial emotion detection, and continuously updating the models to account for new influencer trends and platform-specific content, the system could become an even more valuable tool for brands, regulators, and marketers aiming to monitor and analyse digital content more effectively.

Chapter 6

Conclusions

6.1 Summary of Findings

This dissertation successfully demonstrated the design and implementation of a multimedia data extraction and analysis tool focused on video and image processing. The system integrates cutting-edge AI technologies such as YOLOv8 for object, logo detection, image classification, Whisper.AI for speech recognition, EasyOCR for text extraction, and XLM-RoBERTa for sentiment analysis. By employing these models, the tool was able to accurately detect logos, transcribe speech, extract text from images, and analyse sentiment from influencer-generated content.

The tool's cloud-based architecture, ensures scalability and the ability to handle large datasets. A user-friendly application interface built with Streamlit allows users to upload content, configure services, and review results in real-time, demonstrating the practical applicability of this system for analysing influencer content in various industries.

Tests on real-world multimedia content showcased the system's capacity to detect key elements within influencer content, providing valuable insights for brand compliance, content moderation, and market analysis. Object detection, sentiment analysis, and text extraction worked effectively to provide a comprehensive view of how products are promoted and discussed in multimedia.

6.2 Future Directions

While the current implementation of the system achieves a significant level of functionality, several areas for improvement and expansion have been identified. The future development of this tool could focus on the following aspects.

One of the key challenges identified during testing was the processing time for high-resolution videos and large datasets. Implementing GPU acceleration would be a major enhancement to the system's performance, particularly for resource-intensive tasks like object detection, speech recognition, and OCR. Models such as Whisper.AI and EasyOCR could benefit from GPU usage, drastically reducing inference times and enabling the system to handle real-time or near-real-time content processing. Deploying GPU instances on cloud platforms like AWS would allow the system to scale efficiently and provide faster results.

The existing models in the system work well for general use cases, but fine-tuning them on more specific datasets could improve their accuracy for niche applications. For example, training a custom YOLO model for detecting specific product logos or training sentiment models for influencer-specific language could yield better results. By creating custom datasets, either through manual labelling or leveraging platforms like Roboflow, the system can evolve to meet the needs of specific industries or regulatory environments.

In the current implementation, the upload API handles both file uploads and video frame extraction, which may lead to performance bottlenecks, especially in a production environment. Splitting these two tasks into separate microservices would improve the system's scalability and make it more efficient. The upload API could focus solely on handling file storage, while a separate service could manage frame extraction, enabling asynchronous processing and allowing the system to scale better under heavy loads.

From a more purist microservices perspective, each service should ideally have its own database, ensuring a clear separation of concerns and reducing dependencies between services. This approach would further enhance the system's scalability and resilience, allowing individual services to scale independently based on their unique workloads.

In addition, distributed storage for large files, such as videos, would be beneficial in handling vast amounts of multimedia data. Implementing a distributed storage solution across multiple regions or cloud platforms would ensure both high availability and faster access to the data, especially for large-scale video processing tasks. This would prevent bottlenecks and improve the overall efficiency of the system in handling massive datasets.

Currently, sentiment analysis is applied to text segments, such as transcriptions from Whisper.AI. However, the results from text-based sentiment analysis can fall slightly short of capturing the full emotional context. Expanding this functionality to include facial emotion detection would provide significantly better insights into influencer content. By analysing both text and facial expressions, the system could offer a more comprehensive understanding of the emotions conveyed in a video, greatly enhancing the sentiment analysis process. This would be particularly valuable in gauging the authenticity of emotions behind spoken words, especially in situations like product endorsements or testimonials.

Another enhancement would be the addition of scene detection and person recognition capabilities. Identifying distinct scenes within videos, along with recognising whether the same person appears throughout the content, would add further depth to the system's analysis. For example, the tool could track when different influencers or guests are featured in a video, helping brands understand who is being promoted or mentioned. Scene detection could also help segment long videos, allowing for more targeted analysis of specific sections.

To further increase the tool's utility, the introduction of search and filter capabilities based on keywords, objects, or detected logos would allow users to quickly find relevant information. For example, a user could search for all videos mentioning a specific brand or containing certain product logos. Implementing a search layer over the MongoDB database would enable users to filter results based on predefined criteria, making it easier to analyse large volumes of multimedia content.

The system could be made even more powerful by allowing custom rules based on specific organisational requirements. For example, different regulatory bodies or brands may have unique guidelines for detecting content violations, such as prohibited language or

unauthorised use of logos. By creating a flexible rules engine, the tool could be adapted to enforce these specific rules automatically, flagging potential violations or generating customised reports based on the needs of each organisation.

Chapter 7

Bibliography

[1] Statista, "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025." `https://www.statista.com/statistics/871513/worldwide-data-created/` last accessed 11 Oct. 2024, 2020.

[2] Statista, "Forecast of mobile data traffic worldwide from 2017 to 2022." `https://www.statista.com/statistics/880677/mobile-data-traffic-worldwide/` last accessed 11 Oct. 2024, 2020.

[3] Statista, "Influencer marketing market size worldwide from 2016 to 2024." `https://www.statista.com/statistics/1092819/global-influencer-market-size/` last accessed 11 Oct. 2024, 2020.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[5] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.

[7] SuperAnnotate, "Openai whisper: Automatic speech recognition system," 2023. Accessed: 2023-10-01.

[8] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning*, pp. 28492–28518, PMLR, 2023.

[9] J. AI, "Easyocr: Ready-to-use optical character recognition with 80+ supported languages," 2020.

[10] T.-C. Phan, A.-C. Phan, H.-P. Cao, and T.-N. Trieu, "Content-based video big data retrieval with extensive features and deep learning," *Applied Sciences*, vol. 12, no. 13, p. 6753, 2022.

[11] J. Bragança and F. Silva, "Unveiling the secrets: In-depth analysis of youtube video data and metadata extraction," *To be published*, 2024. To be published soon.

[12] I. Stavrakantonakis, A.-E. Gagiu, H. Kasper, I. Toma, and A. Thalhammer, "An approach for evaluation of social media monitoring tools," *Common Value Management*, vol. 52, no. 1, pp. 52–64, 2012.

[13] E. Perakakis, G. Mastorakis, and I. Kopanakis, "Social media monitoring: An innovative intelligent approach," *Designs*, vol. 3, no. 2, p. 24, 2019.

[14] J. P. S. Rodrigues, A. C. Munaro, and E. C. Paraiso, "Identifying sponsored content in youtube using information extraction," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3075–3080, IEEE, 2021.

[15] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023.

[16] S. Kemmis and R. McTaggart, *The Action Research Planner*. Victoria, Australia: Deakin University Press, 1988.

[17] E. T. Stringer, *Action Research*. Thousand Oaks, CA: Sage Publications, 1999.

[18] J. McNiff, *Action Research: Principles and Practice*. New York, NY: Routledge, 2013.

[19] J. Elliott, *Action Research for Educational Change*. Milton Keynes, UK: Open University Press, 1991.

[20] M. Kaur, A. K. Shukla, and S. Kaur, "An introduction to machine learning in a nutshell," in *2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART)*, pp. 17–22, IEEE, 2021.

[21] G. Stockman and L. G. Shapiro, *Computer vision*. Prentice Hall PTR, 2001.

[22] R. Gorwa, R. Binns, and C. Katzenbach, "Algorithmic content moderation: Technical and political challenges in the automation of platform governance," *Big Data & Society*, vol. 7, no. 1, p. 2053951719897945, 2020.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[24] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[26] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics." `https://github.com/ultralytics/ultralytics`, 2024. Accessed: 2024-01-22.

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

[28] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia computer science*, vol. 199, pp. 1066–1073, 2022.

[29] M. Hussain, "Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, p. 677, 2023.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[33] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[35] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, MA: MIT Press, 1997.

[36] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6645–6649, IEEE, 2013.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pp. 6000–6010, 2017.

[38] B. Liu, *Sentiment analysis and opinion mining*, vol. 5 of *Synthesis lectures on human language technologies*. Morgan & Claypool Publishers, 2012.

[39] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[40] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pp. 216–225, 2014.

[41] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[42] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (A. Moschitti, B. Pang, and W. Daelemans, eds.), (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, Oct. 2014.

[43] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (K. Erk and N. A. Smith, eds.), (Berlin, Germany), pp. 207–212, Association for Computational Linguistics, Aug. 2016.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[45] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[46] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1631–1642, 2013.

[47] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, 2011.

[48] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[49] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[50] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency, "Tensor fusion network for multimodal sentiment analysis," *arXiv preprint arXiv:1707.07250*, 2017.

[51] V. Govindaraju and S. Xiu, "Ocr: An overview," *Handbook of Document Image Processing and Recognition*, pp. 379–388, 2009.

[52] G. Nagy, "Twenty years of document image analysis in pami," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 38–62, 2000.

[53] R. Smith, "An overview of the tesseract ocr engine," *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, pp. 629–633, 2007.

[54] M. S. Kasem, M. Mahmoud, and H.-S. Kang, "Advancements and challenges in arabic optical character recognition: A comprehensive survey," *arXiv preprint arXiv:2312.11812*, 2023.

[55] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pp. 369–376, ACM, 2006.

[56] D. Suh, S. Hong, and J. Kim, "Multilingual text recognition using ocr systems," *International Journal of Computer Applications*, vol. 178, no. 45, pp. 1–10, 2019.

[57] H. Singh and L. Kaur, "Document image analysis for scanned documents: A review," *International Journal of Computer Applications*, vol. 10, no. 5, pp. 14–16, 2010.

[58] S. Newman, *Building microservices*. O'Reilly Media, Inc., 2021.

[59] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," *Present and ulterior software engineering*, pp. 195–216, 2017.

[60] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture," *Ieee Software*, vol. 33, no. 3, pp. 42–52, 2016.

[61] F. Eyvazov, T. E. Ali, F. I. Ali, and A. D. Zoltan, "Beyond containers: Orchestrating microservices with minikube, kubernetes, docker, and compose for seamless deployment and scalability," in *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pp. 1–6, IEEE, 2024.

[62] C. Pahl, "Containerization and the paas cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[63] S. Rangaswamy, S. Ghosh, S. Jha, and S. Ramalingam, "Metadata extraction and classification of youtube videos using sentiment analysis," in *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, pp. 1–2, IEEE, 2016.

[64] S. B. Nabijonovich, K. Ahror, *et al.*, "Unveiling the future of data extraction using python and ai for video-based information recognition," *American Journal of Technology and Applied Sciences*, vol. 17, pp. 26–32, 2023.

[65] Infludata, "Infludata: Social media influencer analytics," 2023. Accessed: 2023-10-01.

[66] Influencer Monitor, "Influencer monitor: Influencer marketing compliance," 2023. Accessed: 2023-10-01.

[67] Storyclash, "Storyclash: Influencer monitoring & performance analysis," 2023. Accessed: 2023-10-01.

[68] Google Cloud, "Google cloud video intelligence api," 2023. Accessed: 2023-10-01.

[69] IBM Watson, "Ibm watson natural language understanding," 2023. Accessed: 2023-10-01.

[70] Microsoft Azure, "Azure ai video indexer," 2023. Accessed: 2023-10-01.

[71] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, 2021.

[72] A. Vina, "Ultralytics yolo11 has arrived! redefine what's possible in ai," 2024. Accessed: 2024-10-08.

[73] B. Dwyer, J. Nelson, T. Hansen, *et al.*, "Roboflow (version 1.0)," 2024. Computer vision software.

[74] Ultralytics, "Ultralytics hub: A unified ai platform for yolo models," 2023. Accessed: 2024-10-08.

[75] J. Camacho-collados, K. Rezaee, T. Riahi, A. Ushio, D. Loureiro, D. Antypas, J. Boisson, L. Espinosa Anke, F. Liu, and E. Martínez Cámara, "TweetNLP: Cutting-edge natural language processing for social media," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (W. Che and E. Shutova, eds.), (Abu Dhabi, UAE), pp. 38–49, Association for Computational Linguistics, Dec. 2022.

[76] NLP Town, "bert-base-multilingual-uncased-sentiment (revision edd66ab)," 2023.

Appendix A

Sequence Diagram

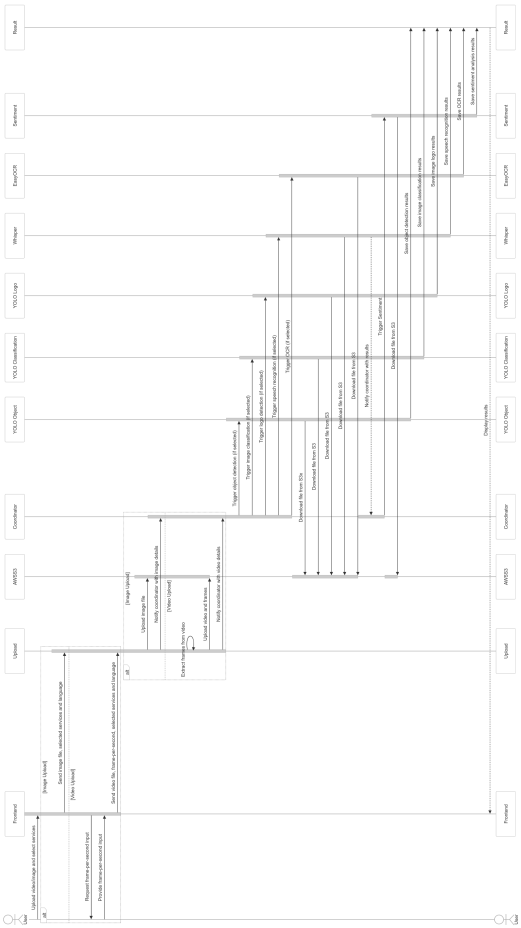


Figure A.1: Sequence diagram showing the process of uploading and processing a file.

Appendix B

Training Yolo Model

```
[ ] !yolo task=detect mode=train model=yolov8x.pt data="/content/Brand-Logo-recognition---Yolov8-1/data.yaml" epochs=2000 imgsz=640 device=0

    255/2000      13.1G     0.5487     0.4292     0.9842         32        640: 100% 22/22 [00:26<00:00,  1.19s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.29it/s]
                   all        100        137      0.657      0.482      0.495      0.217

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    256/2000      13.3G     0.5209     0.4201     0.9599         39        640: 100% 22/22 [00:26<00:00,  1.20s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.32it/s]
                   all        100        137      0.687      0.442      0.491      0.225

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    257/2000      13.1G     0.5195     0.4148     0.9632         34        640: 100% 22/22 [00:26<00:00,  1.20s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:02<00:00,  1.35it/s]
                   all        100        137      0.619      0.481      0.496      0.228

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    258/2000      13.3G     0.5259     0.4315     0.9825         34        640: 100% 22/22 [00:26<00:00,  1.19s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.27it/s]
                   all        100        137      0.657      0.466      0.517      0.238

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    259/2000      13.1G     0.5431     0.4425     0.9625         35        640: 100% 22/22 [00:26<00:00,  1.19s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.29it/s]
                   all        100        137      0.618      0.435      0.471      0.217

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    260/2000      13.3G     0.5336     0.4586     0.9644         35        640: 100% 22/22 [00:26<00:00,  1.19s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.30it/s]
                   all        100        137       0.58      0.476       0.46      0.213

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    261/2000      13.1G     0.5405     0.4405     0.9722         46        640: 100% 22/22 [00:26<00:00,  1.19s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.29it/s]
                   all        100        137      0.559      0.498      0.471      0.223

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    262/2000      13.3G     0.5165     0.4074     0.9683         33        640: 100% 22/22 [00:26<00:00,  1.19s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.29it/s]
                   all        100        137      0.573      0.479      0.472      0.221
EarlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 162, best model saved as best.pt.
To update EarlyStopping(patience=100) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopping.

262 epochs completed in 2.331 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 123.7MB
Optimizer stripped from runs/detect/train/weights/best.pt, 123.7MB

Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.7 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 286 layers, 61,600,383 parameters, 0 gradients, 226.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:04<00:00,  1.08s/it]
                   all        100        137      0.641      0.548      0.547      0.254
                  Coke         26         31      0.497      0.613      0.511      0.176
              Heineken         17         23      0.612      0.348      0.387      0.192
                    Mc         23         35      0.778      0.543      0.601      0.184
                 Pepsi         22         27      0.536      0.428      0.413      0.226
             Starbucks         17         21      0.781       0.81      0.825      0.492
Speed: 0.4ms preprocess, 27.3ms inference, 0.0ms loss, 3.8ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Figure B.1: Training the YOLOv8 model with logo detection dataset.

Appendix C

Upload Interface

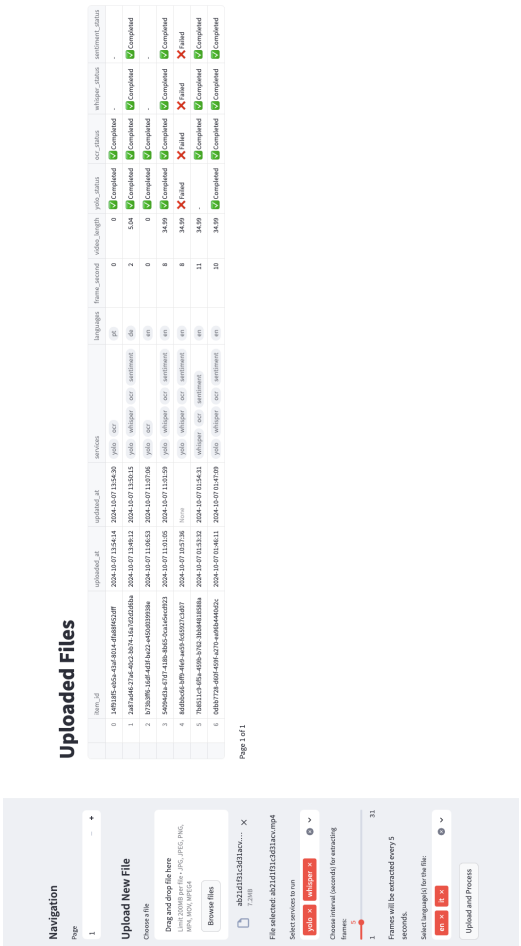


Figure C.1: Interface for uploading new files with service options and file metadata displayed in the table view.

Appendix D

Uploaded Files Interface

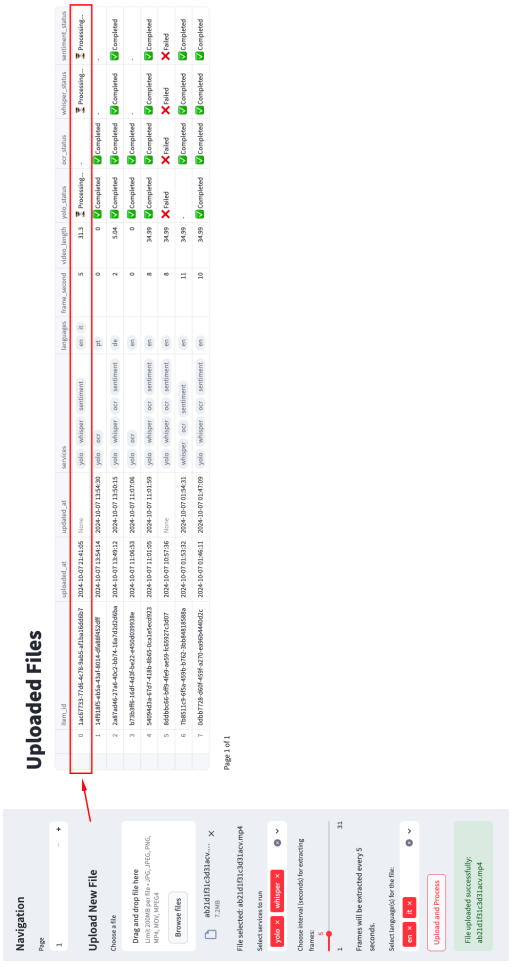


Figure D.1: Uploaded file entry showing ongoing processing with the status of selected services.