NOVA IMS Information Management School

MEGI

Master Degree Program in

Statistics and Information Management

Computer Vision for Real-Time Boxing Video Analysis Automation

State-of-the-art computer vision on Video Analysis

Felipe Dias SIlverio

Master Thesis

presented as partial requirement for obtaining a Master's Degree in Statistics and Information Management

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Computer Vision for Real-Time Boxing Video Analysis Automation

State-of-the-art computer vision on Video Analysis

by

Felipe Dias Silverio

Master Thesis presented as partial requirement for obtaining the Master's degree in Statistics and Information Management, with a specialization in Data Analytics

Supervised by

Carina Albuquerque, PhD, NOVA Information Management School

June, 2025

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism, any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, 02/06/2025

Felipe Dias SIlverio

ABSTRACT

Computer vision (CV) is gaining relevance in sports analytics, offering automated tools for performance assessment and strategic insights. In high-speed, dynamic sports like boxing, implementing CV poses several challenges, including motion blur, body occlusion, and frame instability. This study evaluates the effectiveness of four deep learning architectures in detecting, tracking, and classifying boxing movements in real time. The selected models represent different architectural paradigms: convolutional neural networks (CNNs), a hybrid CNN-transformer model, and a fully transformer-based model. These architectures are implemented through YOLOv5, YOLOv11, YOLOv12, and RF-DETR, respectively. The research follows a two-stage evaluation process. First, all models are tested for object detection accuracy using a public boxing dataset and evaluated with standard metrics such as precision, recall, and mean average precision (mAP). YOLOv11 and YOLOv12 reached top performance (98,9% and 98,4% mAP, respectively), while YOLOv5 achieved the highest recall (98%), despite slightly lower precision. In the second stage, the same models are implemented in a real-time processing pipeline using full boxing match footage to assess tracking stability and inference speed, measured in frames per second (FPS). YOLOv5 maintained the best balance between accuracy and speed, reaching up to 17,63 FPS and the lowest ID switch count (7). Although the transformer-based models demonstrated strong object detection metrics, their computational demands significantly impacted real-time viability. A custom player clustering algorithm was integrated into the pipeline to address the recurring issue of ID loss, especially during occlusion or body overlap, and proved effective across all models to resolve player-ID tracking loss due to occlusion. This work contributes to the growing field of CV in sports by proposing an innovative method, clustering, to solve player occlusion during the fights, and by offering a comparative benchmark of leading model architectures under realistic, operational conditions, and recommends further research into lightweight models, mixed models of object detection and pose-based recognition systems.

KEYWORDS

Computer Vision; Convolutional Neural Networks; Boxing Analytics; Real-Time Detection; Fighting Sports

Sustainable Development Goals (SDG):

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AITF	Average Inference Time per Frame
AI	Artificial Intelligence
CNN	Convolutional Neural Network
CV	Computer Vision
DETR	Detection Transformer
DL	Deep Learning
EXIF	Exchangeable Image File Format
FPS	Frames Per Second
FT	Frames per Track ID
GAN	Generative Adversarial Network
GELAN	Generalized Efficient Layer Aggregation Network
HMM	Hidden Markov Model
ID	Identity
IoU	Intersection over Union
IS	ID Switches
LLM	Rectified Linear Unit
LSTM	Long Short-Term Memory
LLaMA	Largue Language Model Meta AI
MHA	Multi-Head Attention
ML	Machine Learning
MMA	Mixed Martial Arts
MoE	Mixture of Experts
NMS	Non-Maximum Suppression
PD/RP	Punches Detected / Real Punches

PGI	Programmable Gradient Information
PPC	Problem, Purpose, Contribution
ReLU	Rectified Linear Unit
RGBD	Red-Green-Blue-Depth
ROI	Region of Interest
RPN	Region Proposal Network
RT-DETR	Real-Time Detection Transformer
SDG	Sustainable Development Goals
SigLip	Single-label Image Pretraining
SORT	Simple Online and Realtime Tracking
SSD	Single Shot Multibox Detector
SVM	Support Vector Machine
TCN	Temporal Convolutional Network
UMAP	Uniform Manifold Approximation and Projection
UTI	Unique Tracker IDs
ViT	Vision Transformer
YOLO	You Only Look Once
mAP	Mean Average Precision
mAP@0.5	Mean Average Precision at IoU threshold 0.5

1. Introduction

Computer Vision technology has transformed multiple segments across various industry sectors, impacting our society in both the digital and physical worlds, and affecting fields such as security, robotics, healthcare, and self-driving vehicles (W. Wang & Siau, 2019). Its main kickstart occurred with the release of AlexNet, which revolutionized the industry with its Convolutional Neural Network (CNN) approach. It continues to flourish due to a partnership with the rise of big data, allowing current models to consume large amounts of data (Voulodimos et al., 2018). Currently, most computer vision models are based on CNN architecture or the newest transformer architecture.

The combat sports products market was worth 8,7 billion USD in 2023 (*Combat Sports Products Market Size and Trends 2029*, n.d.), with a growing tendency, and this accumulation of capital has attracted numerous researches into combat sports, and boxing, including computing vision either by making use of sensors or pure image processing (Pang et al., 2022). Advancements in Deep Learning (DL) technology, specifically in Computer Vision, have helped boost the development of new applications capable of conducting scoring video analysis on boxing matches to assist judges in their decision-making (Cardino et al., 2022) and even perform real-time video analysis (Quinn & Corcoran, 2022). An example of the efforts invested in the development of these applications is the new AI Judge, first announced by Forbes, which is set to be used in the upcoming fight between Tyson Fury and Oleksander Usyk in 2024 (Mazique, 2024).

Other researchers have identified common issues in boxing video analysis, including player occlusion during fights, which can hinder the tracking of each player, and the detection of punches due to occlusion and the diverse hand and body positions used during the punch (Kasiri et al., 2017). This thesis expands upon previous work that confirmed the viability of real-time boxing analysis (Quinn & Corcoran, 2022) by proposing a standard pipeline to compare different models in real-time while computing statistics and implementing a functionality to associate players and punches, and another to solve player occlusion during a match.

The research questions for this study are:

1. Which architectures are capable of performing boxing analysis at near-real-time speed (30 FPs)?
2. What solutions could be developed to solve one of the problems commonly faced by boxing analysis, such as movement occlusion, inference at high speed, and lack of a large public repository of images or trained models for boxing computer vision applications?

This thesis provides a review of relevant literature in the field, initially covering modern Artificial Intelligence applications, computer vision architectures, and their potential in sports and boxing. Afterward, the methodological approaches used to develop, train, and evaluate the models include dataset selection, preprocessing, evaluation, and discussion of the results. This work evaluation consists of two parts: (1) the development of an object detection model capable of detecting the players, boxing bags, punches, and guards in a boxing match; (2) the development of an application that performs object detection, tracking, and statistics output in real-time while handling problems such as player occlusion and association of movement recognition to players. The last chapter summarizes the key findings of the study and proposes future research directions.

2. Literature review

This chapter provides fundamental knowledge on the architecture of different algorithms to comprehend the intricacies of this research better, and how computer vision is associated with boxing video analysis. The identified research gaps based on scientific work are exposed at the end.

2.1. Modern Artificial Intelligence and Machine Learning

As stated by Wu et al. (2025) in their paper, modern artificial intelligence (AI) and machine learning (ML) technologies have been an emerging and transformative subject, which started as systems focused on processing and predicting information and evolved into ones that are capable of making plans, decisions, and acting. This evolution is divided into distinct phases, each marked by its technological breakthroughs and limitations. Nonetheless, the most crucial breakthrough happened in 2012 when a group of researchers from University of Toronto revealed that ImageNet-scale datasets together with high-performance graphic processing units (GPUs) could improve deep neural network's performance on image classification (Krizhevsky et al., 2017; Neural Information Processing Systems Foundation, 2013), a marvel built on top of all the studies developed decades before this revolution in the machine learning field.

Algorithms, such as convolutional and recurrent neural networks, continued to evolve despite facing problems related to data availability, and computational power remained the decisive factor in this field (Wu et al., 2025). It was then proved that concepts like scaling laws revealed that models, when trained with larger datasets could be able to achieve new capabilities (Kaplan et al., 2020), such as GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) achieved when they demonstrated ability to generate human-like text after its set of parameters and training data reached a determined threshold.

There are different phases or ages of AI: the AI 1.0 consists of systems focused on classifying and interpreting information on static data; the AI 2.0 began with the emergence of autonomous decision-making systems where despite classifying static data, it could also adapt its behavior to achieve goals in complex or evolving scenarios; the AI 3.0 revolutionized by merging digital and physical worlds through embodied systems that are capable of perceiving, planning and acting in real time under unstable conditions, and its applications are primarily found in autonomous vehicles, drones, and drones (Wu et al., 2025). The future of AI is now being led by the surge of foundation models, a set of neural architectures that can be used for multiple tasks (Bommasani et al., 2022).

The advent of foundation models enabled the proposal of transformer-based architecture models, exemplified by LLaMA (Touvron et al., 2023), which combines various frameworks, including Mixture-of-Experts (MoE), Multi-Head Attention (MHA), and a Knowledge distillation module. The MoE framework enhances the computational efficiency and

scalability through the dynamic activation of a subset of specialized neural pathways for each query (Shazeer et al., 2017); the addition of MHA allows to enhance the model's capability to perform diverse and complex tasks (Voita et al., 2019); and at last, the knowledge distillation allows a larger “teacher” model to transfer its expertise to a smaller student model (Park et al., 2019), which enables the Large Language Model (LLM) retain its reasoning and reduce its computational requirements, even under real time-time usability, so as the entire system becomes more flexible throughout multiple devices.

2.2. Computer Vision

Research on Computer Vision technology began in the early 1950s. However, one of the earliest, most important papers, Vision, was published in 1982 by David C. Marr, establishing computer vision as an independent study area (Marr, 2010). In the current century, computer vision has allowed for the creation of multiple applications for different fields through its integration with artificial intelligence and the rise of big data. This partnership enabled its use in physical reality, such as in self-driving vehicles, factory automation, and healthcare (W. Wang & Siau, 2019).

Computer vision technologies focus on the acquisition, manipulation, and interpretation of visual data from images and videos (Chen & Wang, 2016). They can be divided into two research fields: 2D vision and 3D vision. The first includes target recognition, tracking, content understanding, and other related tasks. The second contains 3D reconstruction, pose estimation, and others (Poggi & Moeslund, 2021; Smith, 2023).

A common problem faced by this technology was the lack of properly prepared data for computer vision tasks, as the process of building a database suitable for classification requires significant costs (Ma et al., 2024). The creation of the ImageNet large-scale visual recognition challenge kick-started a powerful base for the promotion of this technology, as ImageNet provided the largest image dataset of that time (Russakovsky et al., 2015), allowing for an algorithm such as AlexNet to revolutionize the field with a Convolutional Neural Network (CNN) architecture, which comprises three main neural layers: a convolutional layer, a pooling layer, and fully connected layers (Voulodimos et al., 2018).

The CNN architecture relies on supervised learning which in the case of CV related problem depends on the availability of large and annotated datasets for training, and due to these costs associated with data collection and annotation, research is focusing in alternative methods for this issue such as data enhancement, generating adversarial networks (GANs), and crowdsourcing image annotation (F. Fang & Bao, 2022; Ma et al., 2024; Ørting et al., 2019).

2.3. Architectures

The boom in deep learning technology, especially in computer vision, following the release of AlexNet, enabled the creation of new algorithms within the deep learning realm. Object

detection methods based on deep learning have become mainstream in newer research, and they are mostly divided between CNN-based models and transformer-based models (Sun et al., 2024). This section will explore the intricacies of these two architectures.

2.3.1. Convolutional Networks

The general structure of a CNN and its based models can be roughly divided into two major parts: two-stage object detection (candidate region-based model) and one-stage object detection (classification/regression-based model). The first type uses a region of interest to generate candidate bounding boxes, extracting features within it and passing them to a pooling layer to identify the category of objects. The second, in contrast, merges both steps into one. The RestNet and VGG are among the most popular architectures in CNN networks, which consist of three types of layers: convolutional layers, pooling layers, and fully connected layers (Sun et al., 2024).

As can be observed in the Figure 1, there are two object detection architectures: (a) a two-stage detector like a Faster R-CNN, and (b) a single-stage detector like YOLO or SSD. In the first step, the input image is processed by a convolutional backbone to extract features, which are then utilized by a Region Proposal Network (RPN) to suggest candidate object regions. These proposals undergo Region of Interest (ROI) pooling before being classified and localized in a second stage. In contrast, the second approach eliminates the proposal step, directly predicting object classes and bounding boxes from the feature maps produced by the backbone. Faster R-CNN offers higher precision but is slower, while the single-stage approach is faster and more suitable for real-time applications (Sun et al., 2024).

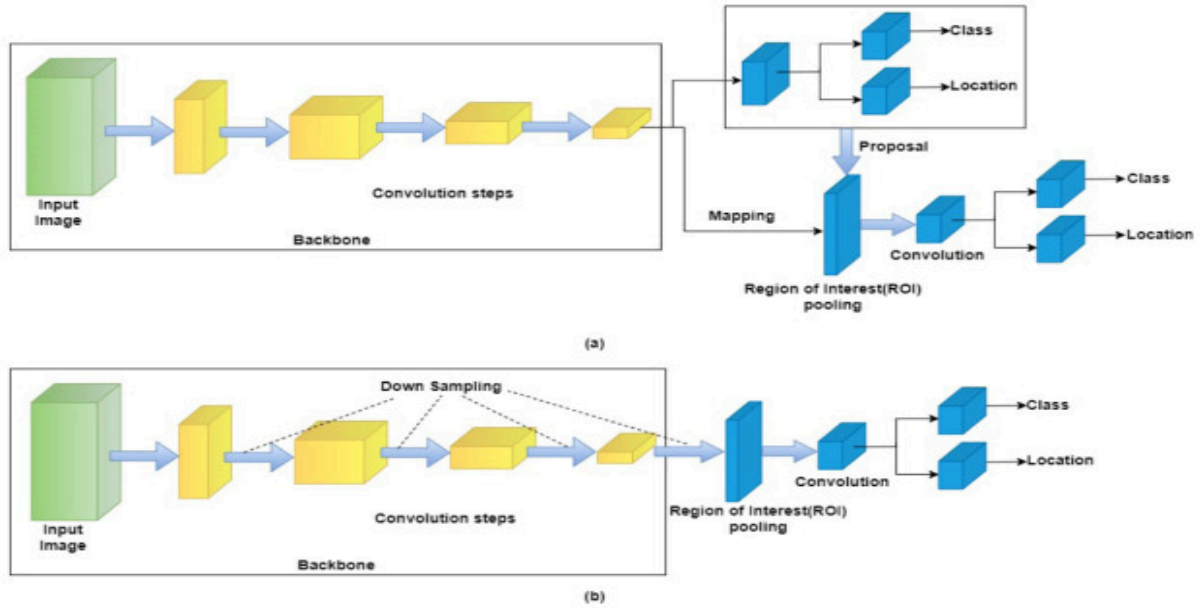

Fig. 1. Comparison between part (a), the basic structure of the two-stage detector, and part (b), the structure of the one-stage type (Sun et al., 2024).

In summary of the process explained in the paper, the structure of a general CNN network consists of a pipeline where at first receives the original image data, converts it into a feature map through the usage of a feature detector and a filter with its ReLu activation function (Agarap, 2019), passes the information for the pooling layer which reduces the dimension of the data to minimize meaningless information and maximize selected features importance (Yongbing Zhang et al., 2011). Afterwards, the compacted data is sent into a fully connected layer, allowing the output neurons to be transformed into binary values through a softmax classification layer (Chen et al., 2022).

2.3.2. TRANSFORMERS

CNN models excel in feature extraction, as they are designed to capture visual semantics and enable the representation of image hierarchical structure in the form of semantics. However, its performance is restricted by biases in variance and receptive fields that limit its ability to recognize long-range correlations and contextual features. Meanwhile, Visual Transformers (ViT) relies on minimal inductive biases, allowing it to capture attention links between input tokens and obtain global connections that model the structural dependencies between input features (Jin & Ma, 2023).

To understand how transformer architectures differ from CNN architectures, it is essential to comprehend the structure of a generic transformer. It was Vaswani et al. (2023) who proposed a self-attention-based model transformer. Its core idea relies on computing a weight matrix by combining weighted representation information from different positions of input sequences to obtain an output representation for each position. This matrix can be modified according to different features among input sequences, making it flexible for various tasks. This self-attention mechanism enables the model to focus more on the positions of sequences, thereby improving the training process (Sun et al., 2024).

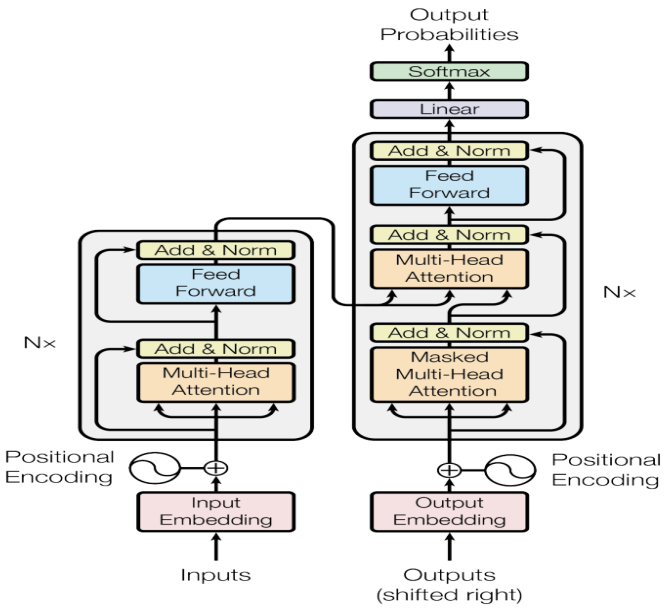

Fig. 2. The transformer model architecture (Vaswani et al., 2023)

The architecture consists of encoder-decoder stacks, an attention function, and a multi-head attention mechanism. Each of the layers within the encoder-decoder contains a feed-forward network with its respective ReLU activation functions. The input and output tokens are translated using embeddings. Specifically, for the output tokens, a softmax function is used to convert their output into next-token probabilities (Vaswani et al., 2023).

The general functioning of the architecture consists of: (1) an encoder: each layer in it contains two sub-layers, where the first is a multi-head self-attention mechanism, and the second is a feed-forward network; (2) a decoder: also contains two sub-layers similar to encoders and adds a third layer dedicated to multi-head attention. The attention mechanism involves transforming queries, keys, and values into vectors through embeddings and additional positional encoding to incorporate sequentiality, and then submitting them to a compatibility function. The weights of this function are then assigned to each value. These functions can be divided into: (1) a scaled dot-product, which uses either an additive or multiplicative where the first outperforms the second in larger values of dimension; (2) a multi-head, it executes the attention function in parallel instead of a single function, resulting in different output values that are concatenated and then project as final values, this mechanism enables the model to analyze different subspaces at different positions. After these attention sub-layers, the output is submitted to a feed-forward network with two linear transformations (Vaswani et al., 2023).

The transformer-based methods can be categorized into two approaches: (1) a Detection Transformer (DETR) series model, which reduces the process of removing redundant boxes in CNN by simplifying the object detection into an encoder-decoder framework. They're mainly

used for detecting tasks; (2) The ViT series model; (2) The ViT series model divides images into patches, and each patch contains a piece of the image area. The pixel values are then stacked into vectors, which serve as input for the model. They're usually used for image classification tasks (Sun et al., 2024).

2.3.3. YOLO Models

The YOLO (You Only Look Once) framework appeared in its first version in 2015-2016 and has been evolving since then. It was originally developed by Joseph Redmon on top of a custom network called Darknet, as a single-stage detector that handles object identification and classification in a single pass. It stands out among other algorithms as a family of versatile algorithms that can be easily deployed, and the most famous implementations were the v5 and v8 by Ultralytics (Nelson, 2025).

The YOLO family of models has the advantage of real-time and simpler structure, and they can be applied in a variety of real-life detection scenarios such as 3D object detection and diseases (Li et al., 2022; Simon et al., 2019). The simplicity of its structure comes from a single convolutional network that predicts the location of bounding boxes and categories together, thus improving the detection speed and achieving higher FPS than two-stage detectors (Sun et al., 2024).

Despite the availability of CNN-based models within the YOLO family, there is also availability using a transformer architecture such as YOLOS-DETR (Y. Fang et al., n.d.; Neural Information Processing Systems Foundation, 2013), which uses an encoder-decoder module of transformers (Sun et al., 2024).

Other versions were released recently, such as YOLOv9, YOLOv10, and YOLOv11 in 2024, and they were built on top of other Ultralytics' models. YOLOv9 introduced programmable gradient information (PGI) and generalized efficient layer aggregation network (GELAN), where the former improves the training process by refining how gradients propagate through the network. The last enhances computational efficiency by reducing redundancy in feature aggregation and providing a scalable architecture for varying sizes of datasets (C.-Y. Wang et al., 2024). YOLOv10 is an upgraded version of YOLOv5, focusing on real-time tasks by introducing an end-to-end head that replaces non-maximum suppression (NMS), thus reducing inference latency (A. Wang et al., 2024). The newest YOLOv11 introduced by Ultralytics brings a new variety of tasks, such as segmentation, classification, and estimation, and surpasses all previous models in terms of accuracy and precision while also offering support for different platforms and computational resources (Glenn Jocher et al., 2023).

As explored in this chapter, the YOLO family is renowned for its CNN-based models, particularly in real-time applications. Nonetheless, recent advancements in the family led to the creation of a real-time detection model based on transformers (Zhao et al., 2024), based on DETR methodology, and makes use of a ViT-based design to reduce computational costs

and together with the usage of different decoder layers without the need for retraining, facilitates the use of this model for real-time scenarios.

2.4. Boxing Market and A.I. Applications

The combat sports products market, valued at 8.7 billion USD in 2023, is expected to grow by 6.5% until 2029, driven by trends in fitness and wellness, technological innovations, and the globalization of combat sports events (*Combat Sports Products Market Size and Trends 2029*, n.d.).

A study conducted in 2022 managed to analyze and organize major technological advancements in computer vision related to Sports, and it revealed how the research is biased towards ball-related sports (such as soccer, tennis, etc.) by around 79% leaving a reasonable gap in other fields. It also states that most of the studies analyzed covered Detection, Tracking, Classification, and Movement Recognition, whereas their applications are done primarily by established machine learning algorithms and architectures such as YOLO-Models, AlexNet, CNN, and others (Naik et al., 2022).

Despite applications directly related to fighting sports, computer vision also demonstrated its usability in education. An experiment conducted by a group of researchers that made use of 3D Pose, CNN, and LSTM to construct an application to be used in martial arts classes, and it was capable of improving students' participation, self-inquiry, and problem-solving ability. According to the paper, the results were confirmed by a paired t-test with two classrooms where one received the AI application (Ye et al., 2022).

2.4.1 Computer Vision in Sports

As is stated by the study review of Computer Vision in Sports (Naik et al., 2022), the building of a model for real-time use in sports comes with major difficulties such as accuracy, computation speed, and model size. These problems vary in terms of optimization when multiple problems are to be attacked at once, such as multi-person tracking, where occlusion, rapid movements, and tracking loss are common. Thus, there is still an open research area for solutions aimed at solving these challenges.

A study published in 2022 revealed the state of AI applications in combat sports and stated: the methods of action recognition can be divided into two: (1) image feature-based method and (2) human joint-based method. In sequence, Pose Estimation is fundamental as it is the basis of understanding the positioning of the human body, where a series of action evaluation metrics can be applied, such as cosine distance, direct comparison, and signal classification. The study also warns about video alignment and how most studies were focused on speed and accuracy rather than providing professional guidance, which reveals an opportunity for research (Pang et al., 2022).

The Table 1 summarizes a series of works that applies computer vision techniques to sport analysis stating different difficulties and usage of varying methodologies in their approaches, such as: the development of an object detection and tracking of balls at soccer through YOLOv3 faced difficulties to perform under high speed conditions (Naik & Hashmi, 2022); the potential as an educational assistant-tool to enhance fighters skills (Ye et al., 2022); the usage of newer models and architectures, like YOLOv5, can lead to build better performance in object detection and tracking (Quinn & Corcoran, 2022); a common problem faced by CV methods in boxing analysis is occlusion, specially for punch recognition, and SVM has been found to be an innovative approach to handle this problem (Kasiri et al., 2017).

Table 1: Studies Compilation of CV applications in Sports

Reference	**Problem Statement**	**Methodology**	**Performance Criteria**	**Remarks**	**Sport**
(Naik & Hashmi, 2022)	Detect and track the ball at high velocity	YOLOv3 and SORT	detection 23.7 FPS, tracking 11.3 FPS, 93.7% accuracy		Soccer
(Ye et al., 2022)	Improve martial arts' impact on students' multidisciplinary skills	Resnet34, LSTM, CNN, TCN, Paired T Test	p-value <0.05 for active participation attitude, independent inquiry ability	Despite a lack of details on the model, it reveals a positive impact in a real scenario	martial art
(Quinn & Corcoran, 2022)	To save time and reduce human error	YOLOv5m	95.5% accuracy at a confidence threshold of 50%.	Confirms the viability of CV applications in combat sports by automating CV models for scoring and analysis	combat sport
(Kasiri et al., 2017)	Occlusion in the recognition of boxers' punches	SVM	97.3% accuracy	Innovative approach using overhead imagery	boxing

2.4.2 Latest Advancements in AI Applications for Boxing

In 2022, the analysis of computer vision applications in martial arts developed by Pang et al. (2022) compiled a variety of studies up to that moment. The algorithms with the highest frequency within these applications studied were composed of Support Vector Machine (SVM), OpenPose, Hidden Markov Model (HMM), and other specific object tracking algorithms. These technologies were employed in action recognition and pose estimation. As for action recognition, it has been identified in two main usages: (1) image feature-based, which consists of feature recognition applied either for pose estimation, fight distinctions, body contour, etc.; (2) human-joint based, which consists of categorization based on joint positioning. Meanwhile, for pose estimation, the studies relied on sensor-based technology and camera-based approaches, both of which are based on kinetic RGBD technology.

Advancements in the field have been driven by the employment of the newest technologies for video analysis and scoring proposals, as proposed by C. Cardino et al. (2022), which proposed an application for scoring boxing matches based on boxer detection, dataset curation, multi-model scoring, and system testing with the intent of being a monitoring tool to be used by judges in real matches. It uses a YOLOv5-based model and a customized CNN model.

Another study proposes a YOLOv5-based model for real-time boxing video analysis, utilizing a diverse dataset comprising Olympic boxing and mixed martial arts (MMA) video samples. This model successfully performed object detection, tracking, pose estimation, and computed statistics. It leaves room for improvement by suggesting the use of YOLOv7 to enhance accuracy and performance, as well as building a large annotated video dataset for boxing (Quinn & Corcoran, 2022).

2.5. Research Gap and Justifications

The literature shows that most computer vision research in sports is focused on ball-related disciplines (Naik et al., 2022). The use of CV in combat sports is emerging and still limited to isolated tasks, such as punch detection or pose estimation, as revealed by the evaluated works in the previous subsections.

Although studies in the field, such as the ones developed by C. Cardino et al. (2022) and Quinn & Corcoran (2022) revealed success in scoring and movement recognition in boxing and MMA, there are still constraints related to processing speed, dataset size, diversity and lack of adaptability to real-world combat dynamics.

Given the previous limitation, this research proposes the comparison of multiple CV architectures on boxing analysis. This comparison happens at two stages: (1) the training of an object detection model; (2) the model performance as a main engine for an application that asks the model to detect players and punches while performing secondary tasks such as player-punch association and fight statistics like the number of punches. The metrics of

comparison are: real-time capability, which is determined by the FPS of each model, tracking loss, and accuracy of punches counted per player. It is expected that this experiment will yield new proposals to solve the problems currently faced by computer vision applications in fight analysis, as previously exposed throughout the literature review.

3. Methodology

In this chapter, it is presented the methodologies used in the development of the pipeline and details regarding the dataset.

3.1. Application Framework

This application consists of building a framework that is capable of object detection, object classification, tracking, and statistics output. The different models with varying architecture will be responsible for detecting players and certain actions (such as a punch), where this last detection will be associated with the closest player's box annotation to attribute the action to the one performing the action.

The dataset used for the development is available on the Roboflow platform under the name Combat Sports (CombatSports, 2025), and it consists of 6 classes (labels). After applying preprocessing and data augmentation methods, as shown in Figure 3, it was used as input for the models. The different models are compared in terms of accuracy when detecting each class available during the training of the object detection model; afterwards, the models are used as an engine of an application that performs object detection of players and punches, player tracking, and player clustering. The objective of this last phase is to detect which model is best suited for a real-time scenario.

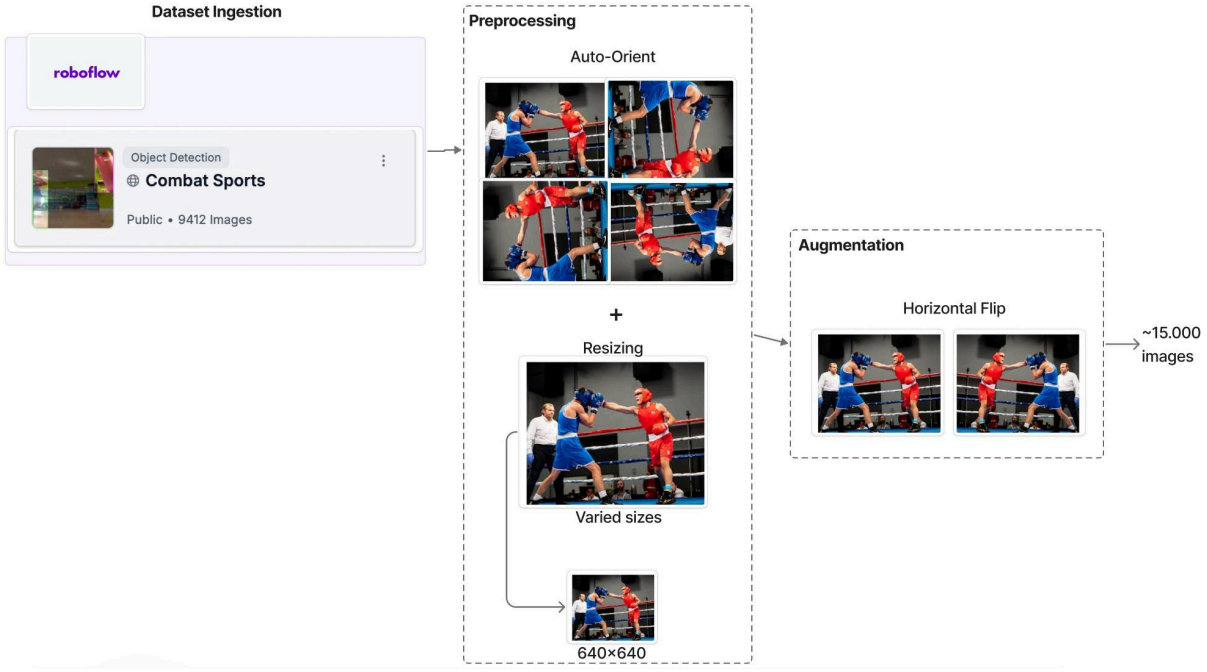

Fig. 3. Design of Data Preparation

Once it has been through the preprocessing and data augmentation steps, it is submitted to the selected model for training; once it has been trained, the model is subjected to perform its detections on real-time speed videos.

In this last stage, the model is subjected to two different pipelines, one dissociated from the other. As can be seen in Figure 4 and Figure 5, both pipelines share a similar methodology: after receiving the input video, a loop process initiates in order to iterate through every frame. The iterations of player detection, punch detection, and player-punch association are similar between the two pipelines. The objective of having two pipelines is to evaluate how the impact of extra calculations, such as the addition of clustering in the pipeline and reassignment of player ID, will affect the FPs of the application; thus, the simplest pipeline can be observed in Figure 4, and the most complex in Figure 5.

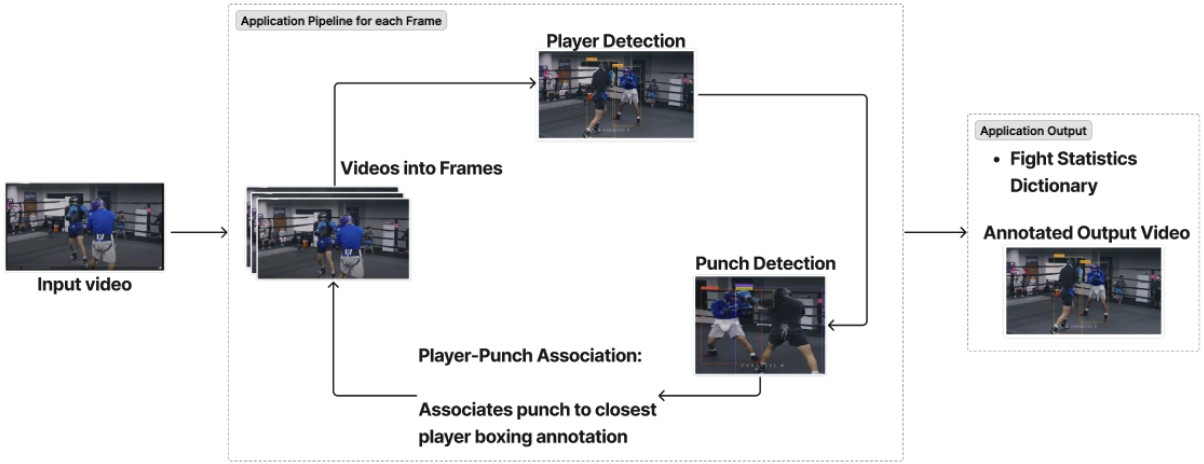

Fig. 4. Simple Application Pipeline

The second pipeline in Figure 5 adds two new steps to the loop iteration: clusterization and player ID reassignment. These steps were added to enhance player ID tracking, as the player ID can be lost due to players' occlusion during the fight. The clustering addition assigns the player ID to a higher-level ID, the cluster ID. Thus, even if the player receives a new ID once the occlusion occurs, it will be overwritten by the cluster ID in the next frame.

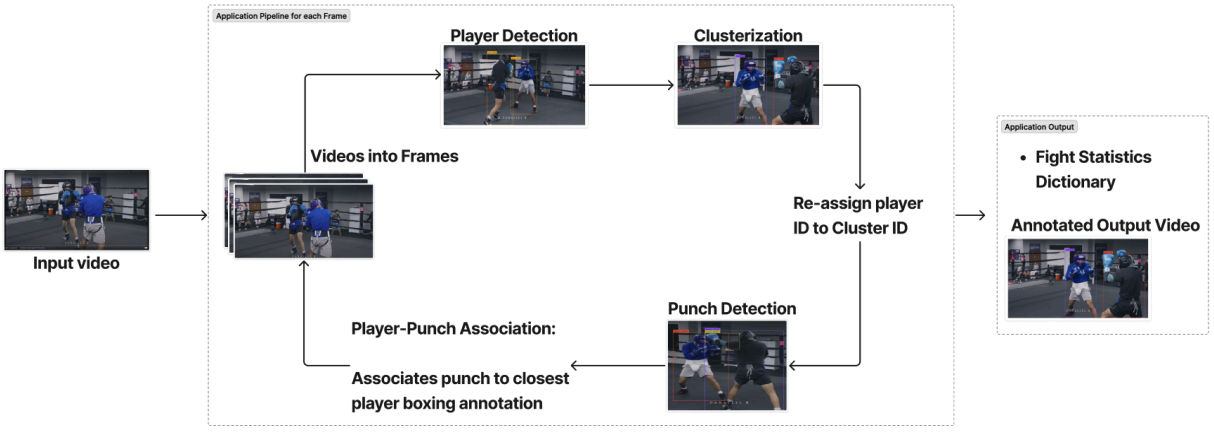

Fig. 5. Complex Application Pipeline

The training pipeline of the clustering model is not depicted in Figure 5 because it occurs outside the main workflow and is illustrated in Figure 6. The clustering training pipeline is

based on the TeamClassifier model, a package used to identify different teams in sports analysis through visual clustering, which requires the usage of player detection on multiple frames, so that the identified players can be cut out from the frame, and then be used as input for the training model (Sports: Computer Vision and Sports, n.d.). Each cropped image is passed through SigLip (Zhai et al., 2023), a vision transformer model that outputs a 768-dimensional embedding per image; these embeddings capture the visual semantics of the image, such as color. The Uniform Manifold Approximation and Projection (UMAP) is applied on the embedding to reduce it to Nx3 vectors (McInnes et al., 2020). These vectors are then clustered into two groups by a K-Means algorithm, and receive their respective label.

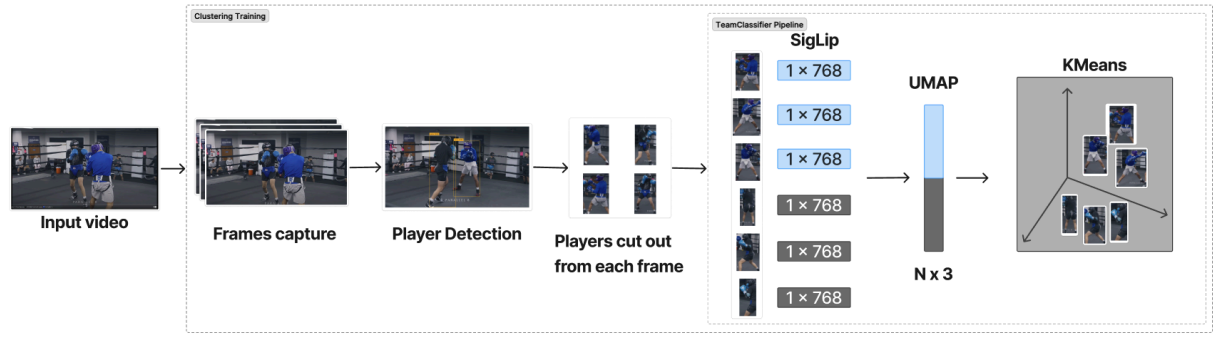

Fig. 6. Clustering Pipeline

3.1.1 Environment Configurations

The first phase of development, which involves training the model for object detection, utilized the Roboflow Platform environment for training the Transformer-Based architecture (RF-DETR) and the Roboflow base model, both of which were trained using the Fast algorithm. The other detectors were trained using YOLOv5, YOLOv11, and YOLOv12 lib using the Kaggle notebooks environment, consisting of a machine with 10GB of disk, 29GB of RAM, and 2x GPU T4 of 15GB memory each.

The platforms like Roboflow have proprietary pipelines (with proprietary optimizations) where they may not offer all hyperparameters, limiting reproducibility. Although a premium plan has been used for this research, allowing for the download of the weights of each parameter used for the model training, it is unknown if they're a result of a common pipeline that could be built on opensource environments, thus the reproducibility of the training pipeline of Roboflow models is uncertain.

The second phase of development, consisting of developing the application with object detection, clustering, object tracking, and statistics computing, was performed on a Google Colab environment on a machine with 235GB of disk, 51GB of RAM, and a GPU T4 with 15GB of RAM.

The comparison of the model's performance in the first phase of this project must take into consideration the limitations of the two environments used in this phase of the research.

3.1.2 Models

The experiment is designed to compare different models, with different architectural proposals, in the same tasks within the application. The application pipeline will ask the models to perform object detection on fighters and their punches, while other computations are also being performed, such as punch counting, clustering, and box annotation associations. The model can be better interpreted through the following analysis:

- YOLOv5, a single-stage, fully convolutional object detector designed for real-time performance and ease of deployment, was developed by Ultralytics. It has a traditional CNN-based detection paradigm, which relies on anchor-based detection, a system where fixed bounding box templates guide detection (Khanam & Hussain, 2024a).
- YOLOv11, a hybrid architecture that combines a CNN backbone with a Transformer-inspired dynamic head. The architecture makes use of convolutional layers for feature extraction, while the dynamic head adaptively weights different regions and scales of the image (Khanam & Hussain, 2024b).
- YOLOv12, the architecture integrates optimized self-attention modules into the CNN backbone to enhance spatial and channel feature modeling, improving performance in cluttered or low-contrast scenarios (Alif & Hussain, 2025).
- RT-DETR, proposes a transformer-based architecture. The main application of this new model is to perform real-time detection while maintaining high accuracy, and provides better global context understanding, which could potentially bring better results in occlusion problems (Zhao et al., 2024).

The reason for their selection relies on testing newer architectures to perform the tasks proposed, as recommended in the work of Quinn & Corcoran (2022). Due to computational limitations and to facilitate faster training and a better workflow in the dataset, from dataset ingestion to preprocessing, data augmentation, and model training, the Roboflow platform has been chosen to carry out this first part of the development.

In the second part of development, these models are applied to new video samples to perform player detection, punch detection, player tracking, and statistics computation with the expectation of reaching real-time performance, which can be defined as numbers bigger than 30 FPS. A clustering algorithm is applied to the image in order to detect distinct players, and attribute the respective player's boxing annotation to each player's respective cluster. The objective of this approach is to reduce the players' ID tracking loss. The player-punch association mechanism works by attributing the punch to the closest player boxing annotation as soon as it is detected.

3.2. Dataset

The image dataset consists of multiple images of different combat sports, either boxing, MMA, or person versus boxing bag. The dataset is important for object detection of the fighters and recognition of their different actions. It is composed of images from different sources within the combat sports realm and of varying quality. It has been chosen not only for being the biggest publicly available dataset on combat sports, but also for its class versatility.

It originally consisted of 9,412 images with six different classes, and after preprocessing and data augmentation, the dataset increased to between 15,353 and 15,474 images, like boxing-bag, high-guard, kick-knee, low-guard, person, and punch (CombatSports, 2025). The final dataset used as input for the models reached varying sizes, consisting of three sets: training (91%), validation (7%), and test (2%). This dataset split choice resembles the dataset split originally set for this dataset, Combat Sports, on Roboflow, in which an object detection model YOLOv11 achieved approximately 97% mAP, 93% precision, and 95% recall across all classes (*Combat Sports Model 2*, 2025).

To perform real-time detection, a boxing match video publicly available (Parallel 8 Productions, 2025) is selected, as it features constant single-camera video footage of the entire match, allowing for the tracking of both players throughout the video. This scenario is close to ideal, as it avoids the use of multiple-angle footage, which is common in other videos.

3.3. Preprocessing

As is demonstrated in Figure 3, the very first step within preprocessing is the usage of auto-orient for every image. Every image contains metadata that reveals the orientation in which the image should be displayed, and this information is stored in the Exchangeable Image File Format (EXIF) orientation field. Afterward, the image goes through a resizing process where all images are standardized on a 640x640 format.

Once the preprocessing is done, the dataset goes through a data augmentation process, where the aim is to increase the size of the dataset by applying certain transformations to the images in order to give more context for the model, and in this step, a horizontal flip.

3.4. Evaluation Metrics

At the first phase of evaluation, object detection evaluation, it is evaluated the capability of the model to detect each class of the Combat Sports dataset through metrics like F1 Score for the test's accuracy, mean average precision (mAP@0.5) to evaluate the object detection performance across all classes, recall, and precision. In the second phase, the performance of the models is evaluated considering the application's mechanisms and near-real-time stress. This study proposes the following metrics: an average inference time per frame (AITF), an

effective FPS, ID switches, frames per track ID, and the ratio of the number of punches detected to real punches in the video.

3.4.1 F1 Score

The F1 Score is a harmonic mean of Precision and Recall, providing a balanced metric that considers both false positives and false negatives. This measure is particularly useful when the dataset is imbalanced, as it avoids the bias of metrics that only consider one type of error. It is defined as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (1)$$

In this project, the F1 Score is calculated for each class of the dataset (e.g., person, punch, boxing bag, etc.) and averaged to assess the model's ability to simultaneously maximize both identification correctness and completeness (Sokolova & Lapalme, 2009).

3.4.2 Mean Average Precision

Mean Average Precision (mAP) calculates the average precision for each class at an Intersection over the Union (IoU) threshold of 0,5, and then averages these values across all classes. It is defined as:

$$mAP@0.5 = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad (2)$$

Where AP_i is the average precision for class i, and N is the total number of classes. This metric evaluates both the precision and the spatial accuracy of the bounding box predictions. A higher mAP score implies that the model is accurately locating and classifying objects in the images (Everingham et al., 2010).

3.4.3 Recall

Recall measures the model's ability to detect all relevant instances of a class. It is defined as the ratio of true positive predictions to the total number of actual positive samples:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (3)$$

A high recall indicates that the model successfully identifies most of the relevant objects in the dataset (Powers, 2020).

3.4.4 Precision

Precision is defined as the proportion of true positive predictions out of all predicted positive instances:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (4)$$

High precision means that most objects detected by the model are indeed correct, which is essential in avoiding overcounting actions like punches or mislabeling players (Saito & Rehmsmeier, 2015).

3.4.5 Average Inference Time Per Frame

The Average Inference Time Per Frame (AITF) measures how much time the model takes to process each frame on average during inference. It is calculated as:

$$AITF = \frac{Total\ Inference\ Time}{Number\ of\ Frames} \quad (5)$$

This metric helps determine if the system is capable of real-time operation by assessing time consumption per frame.

3.4.6 Effective FPS

Effective Frames Per Second (FPS) measures how many frames the system is able to process per second during inference. Unlike theoretical FPS, this is derived from actual performance measurements and depends on model complexity and system resources. It is defined as:

$$FPS = \frac{Number\ of\ Frames}{Total\ Inference\ Time} \quad (6)$$

A higher FPS value indicates that the model can handle high-throughput video processing more effectively, which is essential for real-time boxing analysis.

3.4.7 Unique Tracker IDs

The Unique Tracker IDs (UTI) metric counts how many different object identities were generated by the tracking system during a session. Ideally, each tracked player should maintain a consistent ID throughout the video. A high UTI count may suggest that the model is misidentifying or resetting players' identities too frequently.

3.4.8 ID Switchers

ID Switchers (IS) refers to the number of times a tracker assigns a different ID to the same object over time. This typically occurs due to occlusion, overlapping, or missed detections. A

low number of ID switches is desirable, as frequent switching can disrupt tracking consistency and downstream statistics.

3.4.9 Frames Per Track ID

Frames Per Track ID (FT) quantifies the average number of consecutive frames a single tracker ID persists. It reflects the stability of identity assignment over time. A higher FT indicates that a model can maintain consistent tracking, which is crucial in combat sports scenarios with fast movements.

$$FT = \frac{Total\ Frames}{Unique\ Tracker\ IDs} \quad (7)$$

3.4.10 Punches Detection

This metric evaluates the ratio between the number of punches detected by the model and the actual number of punches annotated manually in the video. It is a direct measurement of the action detection system's recall and accuracy. It is expressed as:

$$PD/RP = \frac{Detected\ Punches}{Real\ Punches} \quad (8)$$

4. Results and Discussion

In this chapter, the results of the experiments of the first part of the object detection model training are presented, and its usage in the pipeline for real-time applicability in the second stage.

4.1. Object Detection Training

An object detection approach was chosen in order to detect the movement of players and their actions in a boxing match. As it was explained in the previous chapter, a couple of architectures were selected for a comparison study regarding their performance in object detection for six classes: boxing-bag, high-guard, kick-knee, low-guard, person, and punch. The architectures used for the training of each model were: a CNN Single-Stage detector, which uses an anchor-based detection, a Hybrid CNN + Transformer with a dynamic head, a Transformer-based (DETR), and a CNN backbone with optimized self-attention modules. Each of these architectures is represented through the following models: YOLOv5, YOLOv11, RT-DETR, YOLOv12. To establish a common comparison between all models, the standard and proprietary model Roboflow will be used since it is the simplest model available on the platform (Gallagher, 2023).

The comparison of object detection models, presented in Table 2, shows that all models deliver high accuracy, but with differences in balance and consistency. Overall, all models perform exceptionally well in the task designated, although with minor differences between each other, differences that are not meaningfully significant. If it were to select the model with minimum complexity and maximum performance, the YOLOv5, with its architecture of a CNN Single-Stage detector, although surpassed by its competitors, is highly capable of performing the tasks at this phase of development. To emphasize the specific differences among models, YOLOv11 and YOLOv12 stand out with identical top performance: both reach 98,9% and 98,4% mAP@50, respectively, and achieve the highest F1 Scores of 97,24. YOLOv5 shows the highest recall (98%) but a slightly lower precision (95,5%) and F1 score (96,67), suggesting it detects nearly all relevant objects but at the cost of slightly more false positives. Roboflow performs similarly to YOLOv11 and YOLOv12 with a strong F1 Score (97,14), showing that it is also reliable across classes. Meanwhile, RF-DETR maintains decent performance, but its lower F1 Score reflects weaker consistency compared to the YOLO-based models.

Table 2: Metrics Comparison for Model's Object Detection Task

	map@50	**Precision**	**Recall**	**F1 Score**
YOLOv11	98,9%	96,8%	97,7%	97,24
RF-DETR	94,8%	92,5%	87%	87,66

Roboflow	98,5%	96,9%	97,4%	97,14
YOLOv12	98,4%	96,8%	97,7%	97,24
YOLOv5	98,3%	95,5%	98%	96,67

The considerations for selecting the ideal model are still to be determined by their performance and computer power requirements to perform their respective functionalities in real-time boxing video analysis.

4.2. Real-Time Detection Application Development

Each of the models presented previously with their distinct architectures are put to trial in two pipelines where the first pipeline consists of: (1) player detection, (2) player tracking, (3) punch detection, (4) punch attribution, (5) statistics output; the second pipeline is composed of: (1) player detection, (2) player clustering, (3) player tracking, (4) punch detection, (5) punch attribution, (6) statistics output.

The punch and player detection tasks are performed using the same model, and the player tracking is made possible through the ByteTracking library. The player clustering task is introduced in the second pipeline in order to experiment with whether it is possible to solve the player's ID tracking loss due to occlusion or movement outside of the video frame; the way that clustering solves this problem is by attributing the player's ID to a higher class, the cluster. In this way, each player can be represented from a higher class to a lower class: player cluster-ID, player tracking ID. Even though the tracking ID is changed, it will still be associated with the same cluster-ID, thus avoiding the detection of multiple non-existing fighters.

As can be seen, this is an example of when player occlusion can happen during a boxing match, where one of the players is in front of the other, and in such a scenario, when the model detects the occluded player again, it would receive a new tracking ID. In this strategy, once the occluded player is detected once again, it is rapidly clustered and its new tracking ID is associated again with the old cluster. Although it can be seen in Figure 7, it is not perfect since the result of the very first frame after player occlusion is wrongly clustered, but in the next frame, it is corrected and inserted in the right cluster.

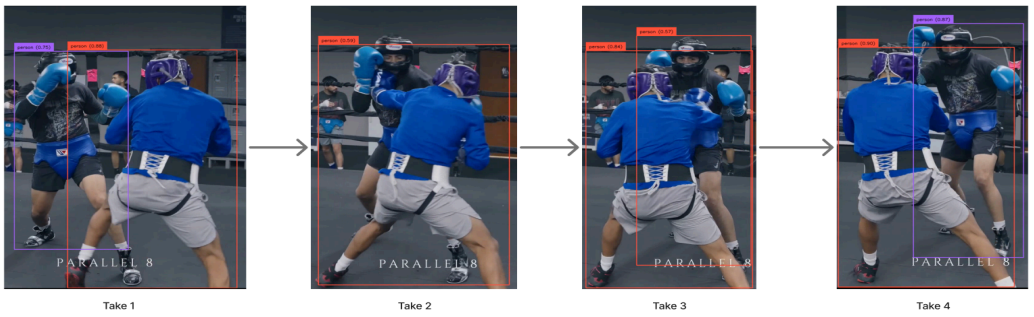

Fig. 7. Frame-by-Frame follow-up of clustering acting in tracking loss.

The table 3 makes a comparison of each model selected to represent its architecture through a few selected metrics, as explained in the previous chapter. Considering the objective of achieving real-time performance in boxing video analysis, the table 3 reveals the measures for model efficiency through inference speed (Avg ITF), effective FPS, and ID stability.

YOLOv5 achieved an average inference time between 0,019 and 0,022s per frame and effective FPS varying between 7,1 and 17,63. It achieves near-optimal real-time speeds (close to or over 30 FPS) while maintaining very low total ID switches (TW id: 7) and stable identity tracking (UT id: 5). YOLOv11 also shows promising real-time potential. In the non-clustered pipeline, it achieves 11,36 FPS and a lower inference time (0,067s), with manageable ID switching (TW id: 12), suggesting it can operate in near real-time with relatively stable tracking. Its clustered performance, however, drops to 6,14 FPS, not so far from YOLOv5.

YOLOv12 and RF-DETR, on the other hand, fall short of real-time applicability. YOLOv12 reaches only between 3,26 and 4,56 FPS, with higher ID switching (TW id: 64 and 20.06) despite clustering. RF-DETR, being the slowest (0,61s per frame, ~1,4–1,5 FPS), is unsuitable for real-time use regardless of pipeline; while it performs well in maintaining UT IDs (8), the delay in inference makes it impractical for live analysis. Roboflow, used as a baseline, performs poorly in both real-time performance and ID management, especially in the non-clustered setup where UT ID explodes to 145 and ID switches reach 379, indicating frequent identity loss and re-assignment due to occlusion.

Table 3: Real-time Metrics Performance per Model

Model	Pipeline	Total Frames	Avg ITF	Eff. FPS	UT id	TW id	punches_p1	punches_p2
YOLOv11	clustered	1156	0,0682	6,14	3	24	25	13
	non-clustered	1152	0,067	11,36	3	12	27	6
RF-DETR	clustered	1156	0,6117	1,41	8	28	86	54
	non-clustered	1152	0,6132	1,57	8	14	56	80

Roboflow	clustered	1156	0,0741	5,79	5	40	10	3
	non-clustered	1156	0,0082	31,11	145	379	4	None
YOLOv12	clustered	1152	0,1968	3,26	4	64	2	24
	non-clustered	1152	0,1954	4,56	65,46	20,06	4	32
YOLOv5	clustered	1152	0,0192	7,1	5	7	81	140
	non-clustered	1152	0,0226	17,63	5	7	81	140

In conclusion, YOLOv5 is the optimal choice for a real-time system due to its combination of speed, stability, and simplicity. YOLOv11 is a viable second-best, particularly when moderate latency is acceptable. Clustering clearly reveals itself as essential since all models detected higher than two tracking IDs; thus, this strategy is a must to avoid losing the player performance.

The Table 4 reveals the consumption of computer power, and it shows that across all models, the implementation of the clustering strategy also affected the trade-off between RAM and GPU usage. The clustered approach has decreased the usage of GPU memory while increasing the other, and the non-clustered approach has the opposite behavior. Despite these trade-offs, there is a clear high usage of GPU memory, which could bring up the hypothesis that the performance of the models within the real-time detection application is being handicapped due to algorithm performance.

Table 4: Computation Power Usage per Model on Real-Time Task

Model	Pipeline	Total Frames	GPU usage	Memory Usage
YOLOv11	clustered	1156	49,39%	10,4%
	non-clustered	1152	55,73%	6,59%
RF-DETR	clustered	1156	50,26%	9,38%
	non-clustered	1152	53,39%	6,59%
Roboflow	clustered	1156	49,6%	8,63%
	non-clustered	1156	59,24%	6,08%
YOLOv12	clustered	1152	65,46%	20,06%
	non-clustered	1152	69,69%	29,49%
YOLOv5	clustered	1152	84,27%	31,41%

	non-clustered	1152	89,38%	30,97%

The base algorithm of Roboflow revealed an extreme deviation of tracking ids due to a problem of generalization where it failed to properly identify the players in the match due to their non-standard boxing clothing, as can be seen in Figure 8, and the model interpreted the players' legs as the players' torso, due to each players having their torso covered by a shirt in the fight.

Fig. 8. The Left part is the output of Roboflow's model, and the right is the output of the YOLOv11 model.

Regarding the punch detection, which faces a different scope of difficulties such as occlusion, rapid movement, not well-placed camera angle, multiple sequential punches at close range, and others, either related to the environment of the boxing ring or camera positioning. Figure 9 shows how different scenarios lead the model to fail at detecting the punch. As stated, punch detection faces a bigger challenge than player detection, and this can be clearly seen in the difference between the actual number of punches overall in the video (18) and the number of punches per player (player blue: 10, player black: 8).

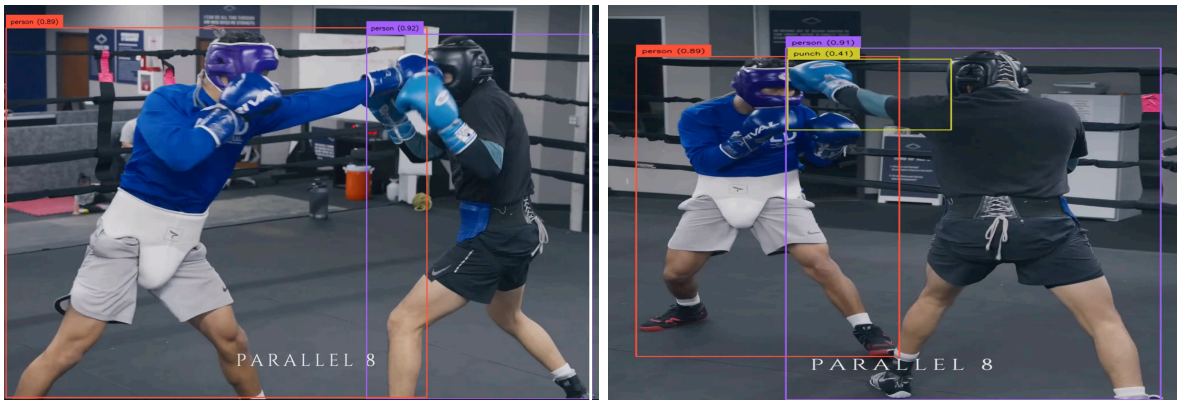

Fig. 9. Examples of punch detection challenges

The challenges faced by punch detection using an object detection approach seem difficult to handle due to the intricacies of box annotation limitations and the possibilities of punch

association with the player. For example, in the right image of Figure 9, once the punch has been thrown, most of its box annotation is within the punch-receiver-end box annotation; thus, the punch would be associated with it. A logic could be applied to associate the punching count to the player whose box annotation has less overlap with the punch's box annotation. Another possible approach would be to completely dissociate punch detection from an object detection strategy and instead apply a skeleton-based action recognition. In this approach, once the hand joints overlap with any other body joint of the opposite player, it is counted as a punch.

4.2. Joint Analysis

The experiment aimed to evaluate multiple object detection models with two core objectives: achieving near-real-time inference performance (≥30 FPS) and introducing an innovative pipeline to address key challenges in boxing analysis, specifically player occlusion and punch detection. To achieve this result, the first phase of the experiment focused on the training of the models. All models demonstrated strong training performance, with minimal variance in accuracy-based metrics. YOLOv11 and YOLOv12 achieved the highest training accuracy, with mAP@50 scores of 98,9% and 98,4%, respectively, and a shared F1 Score of 97,24, indicating strong class balance. YOLOv5 slightly trailed in precision but compensated with the highest recall (98%), suggesting that it identified nearly all target instances, although with a slightly elevated false positive rate (F1: 96,67). Statistically, differences between the YOLO models in training performance were not significant, and all models are considered effective for the detection tasks.

In the second phase of the experiments, the models are submitted to a pipeline that puts all models under stress to perform player and punch detection in real-time or near-real-time.
Particularly under performance stress, a clear divide emerged in the models' ability to meet the real-time inference goal of 30 FPS. Only YOLOv5 approached this threshold, with frame processing times between 0,019 and 0,022s, and actual FPS ranging from 7,1 to 17,63, depending on system load. YOLOv11, while competitive in accuracy, reached 11.36 FPS in the non-clustered pipeline and dropped to 6,14 FPS with clustering, falling short of real-time benchmarks. YOLOv12 and RF-DETR significantly underperformed in this regard, achieving only between 1,4 and 4,56 FPS, making them impractical for live deployment. These models also exhibited higher ID switching, particularly YOLOv12 (TW ID: 64) and Roboflow's baseline model (TW ID: 379), indicating frequent tracking instability.

In contrast, YOLOv5 maintained low identity switches (TW ID: 7) and consistent tracking (UT ID: 5), establishing itself as the only model with viable real-time detection and stable identity management. Regarding the punch detection, none of the models achieved a satisfactory performance close to the real number of detections expected (18 in total). However, between the two main performing models in terms of FPs, the YOLOv11 achieved a significantly lower punch counting than YOLOv5.

The introduction of the clustering strategy was a critical component in the tracking pipeline and proved to be a beneficial innovation. Across all models, clustering effectively reduced the number of fragmented tracking IDs, consolidating detections into more coherent identities, especially important in boxing scenarios where occlusion and motion blur are common. Although clustering introduced a moderate trade-off in speed, increasing inference latency slightly and demanding more RAM, it led to a noticeable improvement in tracking accuracy and model robustness. For example, clustering helped mitigate visual misinterpretation issues caused by non-standard clothing (e.g., leg regions being misidentified as torsos), particularly in Roboflow's baseline. Overall, clustering was a worthwhile addition, especially when paired with high-performing models like YOLOv5 or YOLOv11. It contributes meaningfully to the broader goal of enhancing boxing analysis through intelligent visual tracking.

5. Conclusions and Future Research

This study evaluates and compares different computer vision architectures for real-time boxing video analysis. Four architectures were selected: the CNN Single-Stage detector, which uses an anchor-based detection; Hybrid CNN + Transformer with a dynamic head; Transformer-Based (DETR); and a CNN backbone with optimized self-attention modules, respectively represented by the models YOLOv5, YOLOv11, RF-DETR, and YOLOv12. These models were trained and assessed across two main phases: object detection on a curated dataset and real-time performance during boxing analysis. The intention is to determine which architecture provides the best balance between detection accuracy and real-time performance. As outlined in the introduction, the focus is on applying CV models to automate boxing analysis, thereby supporting ongoing advancements in AI-assisted sports evaluation.

5.1. Limitations

The models submitted to the application faced several challenges typical of boxing matches, which often present scenarios with unbalanced lighting, multiple camera angles, a variety of player clothing, movement occlusion, and the requirement of rapid inference by the model to detect movements. These challenges posed limitations and difficulties for the performance of the models presented in this work when executing the tasks of player detection, punch detection, movement tracking, and player-punch association. Despite these problems, the training of these models can pose a challenge to the machine responsible for their training, since even a paid environment such as Google Colab has its resources heavily utilized. Other than the coding environment, although the training can be done through a platform like Roboflow, it is a paid resource that doesn't offer complete customizability of the model, and neither does it offer the internal functioning of the model architectures publicly. The usage of proprietary platforms such as Roboflow imposes other restrictions, such as the uncertainty that the hyperparameters returned by the platform are indeed all the ones utilized during the training. It is also uncertain how the training pipeline is functioning. Due to these intricacies, the reproducibility of the models trained in a proprietary platform is not guaranteed in other environments.

Although the platform Roboflow can be handy in order to train advanced models on a variety of different datasets available publicly on their platform, those datasets are not of good quality in terms of generalization. There is still a lack of a proper large dataset for boxing analysis that can provide a variety of scenarios such as Olympic boxing, MMA matches, amateur and professional lighting scenes, different camera angles, and image quality. These problems related to dataset quality are carried over to the trained models used to perform boxing analysis on videos, which requires higher generalization capacity under near-real-time speed.

During the development of the application pipeline that applies the player and punch detection, the choice to utilize an object detection approach to perform punch detection faced the problem of movement occlusion and low response by the model to properly detect and recognize the player-punch association. The player-punch association logic is that the player box annotation with the biggest overlap with the punch box annotation will have the punch associated with it. However, due to the model's low response in the near-real-time scenario, the punch can often be associated with the one being punched.

5.2. Conclusions

This section aims to answer the questions that started this work.

Q1: Which architectures are capable of performing boxing analysis at near-real-time speed (30 FPs)?

This research resulted in YOLOv5 being the most efficient model for real-time applications, presenting the best combination of processing speed and detection accuracy, although it didn't manage to reach near-real-time speed. Among the other models, YOLOv11 and YOLOv12, presented competitive accuracy, although their speed performance was even more hindered by inference delays.

When comparing the results to previous studies, this research aligns with past findings from (Quinn & Corcoran, 2022) and (C. Cardino et al., 2022), which supported the use of CNN-based architectures with self-attention modules or transformers, like YOLO-based models, in combat sports. It builds upon previous cited studies by introducing an application that uses state-of-the-art models as an engine to perform player detection and punch detection while addressing challenges faced by the model during a boxing match.

Due to the scenario of model comparison being done in an application that submits the models to multiple problems and secondary computations, such as the ones performed to solve player occlusion and player-punch association, these factors affected the performance of these models to realize their inference in optimal speed.

Q2: What solutions could be developed to solve one of the problems commonly faced by boxing analysis, such as movement occlusion, inference at high speed, and lack of a large public repository of images or trained models for boxing computer vision applications?

Despite the model's performance under real-time stress, it has been found that the applied clustering strategy in the players throughout the boxing match effectively improved the tracking of the players along the video, even in scenarios where player occlusion occurred. In this scenario, the occluded player is usually considered a new entity once it is detected again and receives a new ID. The clustering strategy approached this problem by associating the players' ID with their respective cluster ID thereby overwriting the player ID with the cluster ID.

Other approaches have not been as successful as the clustering strategy, such as player-punch association, which, due to the difficulty of performing in real-time, often associates the punch with the wrong player because the association is being done by the player box annotation with the biggest overlap with the punch box annotation. This one could be improved if the model could be made to function at higher FPs frequencies, although that wouldn't solve the problem of punch occlusion.

5.3. Future Work

This research provides a comparison of different computer vision architectures in a scenario with multiple problems that can diminish their performance in boxing analysis and attempts to find solutions to common problems faced by these models when performing their analysis. This work recommends the use of a clustering strategy to address the issue of player occlusion, as it has proven successful in enhancing player tracking during combat. Although a strategy to solve the player-punch association problem has been attempted, and despite code optimization, which could potentially improve this logic, in a scenario where the movements are occluded, this strategy would bring up the association problem again. Future work could develop newer research on replacing this work approach on player-punch association with a skeleton-based method and optimizing the application coding pipeline in order to improve the performance of all the models.

Bibliographical References

Agarap, A. F. (2019). *Deep Learning using Rectified Linear Units (ReLU)* (No. arXiv:1803.08375). arXiv. https://doi.org/10.48550/arXiv.1803.08375

Alif, M. A. R., & Hussain, M. (2025). *YOLOv12: A Breakdown of the Key Architectural Features* (No. arXiv:2502.14740). arXiv. https://doi.org/10.48550/arXiv.2502.14740

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., Arx, S. von, Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., … Liang, P. (2022). *On the Opportunities and Risks of Foundation Models* (No. arXiv:2108.07258). arXiv. https://doi.org/10.48550/arXiv.2108.07258

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language Models are Few-Shot Learners* (No. arXiv:2005.14165). arXiv. https://doi.org/10.48550/arXiv.2005.14165

C. Cardino, P., Chua, J. L., & Llaga, J. R. (2022). Advance scorecard for boxing: Combat sport analysis with deep learning. *Advance Scorecard for Boxing: Combat Sport Analysis with Deep Learning*. https://animorepository.dlsu.edu.ph/etdb_ece/11

Chen, L., Chu, X., Zhang, X., & Sun, J. (2022). Simple Baselines for Image Restoration. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, & T. Hassner (Eds.), *Computer Vision – ECCV 2022* (Vol. 13667, pp. 17–33). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-20071-7_2

Combat Sports Model 2. (2025, January 28). https://universe.roboflow.com/combatsports-lvdva/combat-sports-52pyh/model/2

Combat Sports Products Market Size and Trends 2029. (n.d.). TechSci Research. Retrieved July 9, 2025, from https://www.techsciresearch.com/report/combat-sports-products-market/21386.html

CombatSports. (2025, January 1). *Combat Sports Dataset* [Open Source Dataset]. https://universe.roboflow.com/combatsports-lvdva/combat-sports-52pyh

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, *88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Fang, F., & Bao, S. (2022). *FragmGAN: Generative Adversarial Nets for Fragmentary Data*

Imputation and Prediction (No. arXiv:2203.04692). arXiv. https://doi.org/10.48550/arXiv.2203.04692

Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., Niu, J., & Liu, W. (n.d.). *You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection*.

Gallagher, J. (2023, November 7). *Announcing Roboflow Train 3.0*. https://blog.roboflow.com/roboflow-train-3-0/

Glenn Jocher, Jing, Q., & Chaurasia, A. (2023). *Ultralytics YOLO* (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

Jin, Y., & Ma, R. (2023). Applications of transformers in computer vision. *Applied and Computational Engineering*, *16*(1), 234–241. https://doi.org/10.54254/2755-2721/16/20230898

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling Laws for Neural Language Models* (No. arXiv:2001.08361). arXiv. https://doi.org/10.48550/arXiv.2001.08361

Kasiri, S., Fookes, C., Sridharan, S., & Morgan, S. (2017). Fine-grained action recognition of boxing punches from depth imagery. *Computer Vision and Image Understanding*, *159*, 143–153. https://doi.org/10.1016/j.cviu.2017.04.007

Khanam, R., & Hussain, M. (2024a). *What is YOLOv5: A deep look into the internal features of the popular object detector* (No. arXiv:2407.20892). arXiv. https://doi.org/10.48550/arXiv.2407.20892

Khanam, R., & Hussain, M. (2024b). *YOLOv11: An Overview of the Key Architectural Enhancements* (No. arXiv:2410.17725). arXiv. https://doi.org/10.48550/arXiv.2410.17725

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Li, Y., Li, A., Li, X., & Liang, D. (2022). Detection and Identification of Peach Leaf Diseases based on YOLO v5 Improved Model. *2022 The 5th International Conference on Control and Computer Vision*, 79–84. https://doi.org/10.1145/3561613.3561626

Ma, B., Li, F., & Ren, Y. (2024). Review of Research Progress in Computer Vision. *Advances in Engineering Technology Research*, *11*(1), 558. https://doi.org/10.56028/aetr.11.1.558.2024

Marr, D. (2010). *Vision: A computational investigation into the human representation and processing of visual information*. MIT Press.

Mazique, B. (2024, December 18). *Tyson Fury Vs. Oleksandr Usyk 2 Will Experiment With An AI Judge (UPDATE)*. Forbes. https://www.forbes.com/sites/brianmazique/2024/12/18/tyson-fury-vs-oleksandr-usyk-2-will-experiment-with-an-ai-judge/

McInnes, L., Healy, J., & Melville, J. (2020). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction* (No. arXiv:1802.03426). arXiv. https://doi.org/10.48550/arXiv.1802.03426

Naik, B. T., Hashmi, M. F., & Bokde, N. D. (2022). A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions. *Applied Sciences*, *12*(9), 4429. https://doi.org/10.3390/app12094429

Naik, B. T., & Hashmi, Md. F. (2022). YOLOv3-SORT: Detection and tracking player/ball in soccer sport. *Journal of Electronic Imaging*, *32*(01). https://doi.org/10.1117/1.JEI.32.1.011003

Nelson, J. (2025, January 9). *What is YOLO? The Ultimate Guide [2025]*. Roboflow. https://blog.roboflow.com/guide-to-yolo-models/

Neural Information Processing Systems Foundation (Ed.). (2013). *Advances in neural information processing systems 25: 26th Annual Conference on Neural Information Processing Systems 2012 ; December 3 - 6, 2012, Lake Tahoe, Nevada, USA*. Curran.

Ørting, S., Doyle, A., Hilten, A. van, Hirth, M., Inel, O., Madan, C. R., Mavridis, P., Spiers, H., & Cheplygina, V. (2019). *A Survey of Crowdsourcing in Medical Image Analysis* (No. arXiv:1902.09159). arXiv. https://doi.org/10.48550/arXiv.1902.09159

Pang, Y., Wang, Q., Zhang, C., Wang, M., & Wang, Y. (2022). Analysis of Computer Vision Applied in Martial Arts. *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 191–196. https://doi.org/10.1109/ICCECE54139.2022.9712803

Park, W., Kim, D., Lu, Y., & Cho, M. (2019). *Relational Knowledge Distillation* (No. arXiv:1904.05068). arXiv. https://doi.org/10.48550/arXiv.1904.05068

Poggi, M., & Moeslund, T. B. (2021). Computer Vision for 3D Perception and Applications. *Sensors*, *21*(12), 3944. https://doi.org/10.3390/s21123944

Powers, D. M. W. (2020). *Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation* (No. arXiv:2010.16061). arXiv. https://doi.org/10.48550/arXiv.2010.16061

Quinn, E., & Corcoran, N. (2022). Automation of Computer Vision Applications for Real-time Combat Sports Video Analysis. *International Conference on AI Research*, *4*(1),

162–171. https://doi.org/10.34190/icair.4.1.930

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). *ImageNet Large Scale Visual Recognition Challenge* (No. arXiv:1409.0575). arXiv. https://doi.org/10.48550/arXiv.1409.0575

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer* (No. arXiv:1701.06538). arXiv. https://doi.org/10.48550/arXiv.1701.06538

Simon, M., Amende, K., Kraus, A., Honer, J., Sämann, T., Kaulbersch, H., Milz, S., & Gross, H. M. (2019). *Complexer-YOLO: Real-Time 3D Object Detection and Tracking on Semantic Point Clouds* (No. arXiv:1904.07537). arXiv. https://doi.org/10.48550/arXiv.1904.07537

Smith, W. (2023). Guest Editorial: Special Issue on Computer Vision from 2D to 3D. *International Journal of Computer Vision*, *131*(2), 405–405. https://doi.org/10.1007/s11263-022-01724-3

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Sports: Computer vision and sports. (n.d.). Retrieved July 14, 2025, from https://github.com/roboflow/sports?tab=MIT-1-ov-file

Sun, Y., Sun, Z., & Chen, W. (2024). The evolution of object detection methods. *Engineering Applications of Artificial Intelligence*, *133*, 108458. https://doi.org/10.1016/j.engappai.2024.108458

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). *LLaMA: Open and Efficient Foundation Language Models* (No. arXiv:2302.13971). arXiv. https://doi.org/10.48550/arXiv.2302.13971

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need* (No. arXiv:1706.03762). arXiv.

https://doi.org/10.48550/arXiv.1706.03762

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). *Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned* (No. arXiv:1905.09418). arXiv. https://doi.org/10.48550/arXiv.1905.09418

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, *2018*, 1–13. https://doi.org/10.1155/2018/7068349

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). *YOLOv10: Real-Time End-to-End Object Detection* (No. arXiv:2405.14458). arXiv. https://doi.org/10.48550/arXiv.2405.14458

Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information* (No. arXiv:2402.13616). arXiv. https://doi.org/10.48550/arXiv.2402.13616

Wang, W., & Siau, K. (2019). Artificial Intelligence, Machine Learning, Automation, Robotics, Future of Work and Future of Humanity: A Review and Research Agenda. *Journal of Database Management*, *30*(1), 61–79. https://doi.org/10.4018/JDM.2019010104

Wu, J., You, H., & Du, J. (2025). *AI Generations: From AI 1.0 to AI 4.0* (No. arXiv:2502.11312). arXiv. https://doi.org/10.48550/arXiv.2502.11312

Ye, W., Li, S., Liu, S., & Zhou, Y. (2022). Application of Artificial Intelligence Technology in Martial Arts Education Governance. *Discrete Dynamics in Nature and Society*, *2022*(1), 5606280. https://doi.org/10.1155/2022/5606280

Yongbing Zhang, Debin Zhao, Jian Zhang, Ruiqin Xiong, & Wen Gao. (2011). Interpolation-Dependent Image Downsampling. *IEEE Transactions on Image Processing*, *20*(11), 3291–3296. https://doi.org/10.1109/TIP.2011.2158226

Zhai, X., Mustafa, B., Kolesnikov, A., & Beyer, L. (2023). *Sigmoid Loss for Language Image Pre-Training* (No. arXiv:2303.15343). arXiv. https://doi.org/10.48550/arXiv.2303.15343

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., & Chen, J. (2024). *DETRs Beat YOLOs on Real-time Object Detection* (No. arXiv:2304.08069). arXiv. https://doi.org/10.48550/arXiv.2304.08069

Appendix A - Ethics Committee Report

RE: NOVA IMS | Ethics Committee - NEED REVIEW

De Ethics Committee <ethicscommittee@novaims.unl.pt>
Data Sex, 13/06/2025 13:44
Para Felipe Dias Silverio <20230890@novaims.unl.pt>; Carina Isabel Andrade Albuquerque <calbuquerque@novaims.unl.pt>
Cc Ethics Committee <ethicscommittee@novaims.unl.pt>

Dear professor Carina Albuquerque,
Dear Felipe Silvério,

Thank you for filling out the Research Ethics Checklist. After reviewing your request, you can proceed with the study as we do not foresee any major ethical concerns with the project.

Project No.: **DSCI2025-6-24426**

Project Title: **Computer Vision for Real-Time Boxing Video Analysis Automation**

Principal Researcher: **Felipe Dias Silverio**

according to the regulations of the Ethics Committee of NOVA IMS and MagIC Research Center this project was considered to meet the requirements of the NOVA IMS Internal Review Board, being considered **APPROVED** on 13/06/2025.

It is the Principal Researcher's responsibility to ensure that all researchers and stakeholders associated with this project are aware of the conditions of approval and which documents have been approved.

The Principal Researcher is required to notify the Ethics Committee, via amendment or progress report, of

- Any significant change to the project and the reason for that change;
- Any unforeseen events or unexpected developments that merit notification;
- The inability of the Principal Researcher to continue in that role or any other change in research personnel involved in the project.

Lisbon, 13/06/2025
NOVA IMS Ethics Committee
ethicscommittee@novaims.unl.pt

This email serves as formal proof of ethical approval. If required for inclusion in a thesis, dissertation, or any other academic documentation, a PDF version of this message may be created and attached accordingly.

Cristina Oliveira
Gestora executiva do centro de investigação MagIC | *Executive manager of the Information Management Research Center (MagIC)*
Find out more about our research at https://magic.novaims.unl.pt/en/

Appendix B - Boxing Analysis Automation Pipeline Code

Player Clustering Algorithm

```
from sports.common.team import TeamClassifier

frame_generator    =    sv.get_video_frames_generator(SOURCE_VIDEO_PATH,
stride=30)

crops = []
for frame in tqdm(frame_generator, desc='Collecting crops'):
   results = model.infer(frame)
   preds = results[0].predictions
    person_preds = [p for p in preds if p.class_name == "person" and
p.confidence > 0.4]

   for p in person_preds:
       x1 = int(p.x - p.width / 2)
       y1 = int(p.y - p.height / 2)
       x2 = int(p.x + p.width / 2)
       y2 = int(p.y + p.height / 2)
       crop = frame[y1:y2, x1:x2]
       if crop.size > 0:  # ensure valid crop
           crops.append(crop)

if not crops:
    raise ValueError("No player crops collected for clustering. Check
model output or class ID filter.")

team_classifier = TeamClassifier(device='cuda')
team_classifier.fit(crops)
```

Object detection of player and punches, tracking and clustering: Pipeline for real-time data

---

```
import supervision as sv
from inference import get_model
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time
import psutil

def box_iou(box1: np.ndarray, boxes2: np.ndarray) -> np.ndarray:
  """Compute IoU between a single box and an array of boxes."""
  x1 = np.maximum(box1[:, 0], boxes2[:, 0])
  y1 = np.maximum(box1[:, 1], boxes2[:, 1])
  x2 = np.minimum(box1[:, 2], boxes2[:, 2])
  y2 = np.minimum(box1[:, 3], boxes2[:, 3])

  inter_area = np.clip(x2 - x1, 0, None) * np.clip(y2 - y1, 0, None)
  box1_area = (box1[:, 2] - box1[:, 0]) * (box1[:, 3] - box1[:, 1])
  boxes2_area = (boxes2[:, 2] - boxes2[:, 0]) * (boxes2[:, 3] - boxes2[:, 1])

  union_area = box1_area[:, None] + boxes2_area - inter_area
  return inter_area / union_area

# Load model
model = get_model()

# Input video path
SOURCE_VIDEO_PATH = 'match1_test.mp4'
OUTPUT_VIDEO_PATH = 'annotated_boxing_ai.mp4'

# Setup video input
cap = cv2.VideoCapture(SOURCE_VIDEO_PATH)
ret, frame = cap.read()  # Read only the first frame
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = cap.get(cv2.CAP_PROP_FPS)
frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

# Setup video writer
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out_video = cv2.VideoWriter(OUTPUT_VIDEO_PATH, fourcc, fps, (width, height))

frame_generator = sv.get_video_frames_generator(SOURCE_VIDEO_PATH, stride=30)
```

```
tracker = sv.ByteTrack()
tracker.reset()
# Setup annotators
box_annotator = sv.BoxAnnotator(thickness=2)
label_annotator = sv.LabelAnnotator(text_color=sv.Color.from_hex('#000000'))

# Performance tracking
start_time = time.time()
total_inference_time = 0
processed_frames = 0
cpu_usages = []
memory_usages = []
seen_ids = set()
id_track_durations = {}
previous_ids = set()
id_switch_count = 0

# Process each frame
cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
for _ in tqdm(range(frame_count), desc="Processing video"):
  ret, frame = cap.read()
  if not ret:
      break

  cpu_usages.append(psutil.cpu_percent(interval=None))
  memory_usages.append(psutil.virtual_memory().percent)

  # PLAYER DETECTION
  frame_start = time.time()
  results = model.infer(frame)
  frame_end = time.time()

  inference_duration = frame_end - frame_start
  total_inference_time += inference_duration
  processed_frames += 1

  preds = results[0].predictions
  person_preds = [p for p in preds if p.class_name == "person" and p.confidence > 0.4]

  if len(person_preds) == 0:
      out_video.write(frame)
      continue

  xyxy = np.array([
      [p.x - p.width / 2, p.y - p.height / 2, p.x + p.width / 2, p.y + p.height / 2]
      for p in person_preds
  ], dtype=np.float32)
```

```
  class_id = np.array([p.class_id for p in person_preds])
  class_names = np.array([p.class_name for p in person_preds])
  confidence = np.array([p.confidence for p in person_preds])
  detections = sv.Detections(xyxy=xyxy, class_id=class_id, confidence=confidence)
  detections = tracker.update_with_detections(detections)

  current_ids = set(detections.tracker_id)
  seen_ids.update(current_ids)
  for tid in current_ids:
      id_track_durations[tid] = id_track_durations.get(tid, 0) + 1
  if previous_ids:
      id_switch_count += len(previous_ids.symmetric_difference(current_ids))
  previous_ids = current_ids


  players_crops = [sv.crop_image(frame, xyxy) for xyxy in detections.xyxy]
  detections.class_id = team_classifier.predict(players_crops)

  labels = [f"{cls_id} ({conf:.2f})" for cls_id, conf in zip(class_names, detections.confidence)]

  annotated_frame = frame.copy()
  annotated_frame = box_annotator.annotate(annotated_frame, detections)
  annotated_frame = label_annotator.annotate(annotated_frame, detections, labels=labels)

  # PUNCH DETECTION
  punch_preds = [p for p in preds if p.class_name == "punch" and p.confidence > 0.2]

  if len(punch_preds) == 0:
      out_video.write(annotated_frame)
      continue

  xyxy = np.array([
      [p.x - p.width / 2, p.y - p.height / 2, p.x + p.width / 2, p.y + p.height / 2]
      for p in punch_preds
  ], dtype=np.float32)
  punch_class_id = np.array([p.class_id for p in punch_preds])
  punch_class_names = np.array([p.class_name for p in punch_preds])
  punch_confidence = np.array([p.confidence for p in punch_preds])
  punch_detections = sv.Detections(xyxy=xyxy, class_id=punch_class_id,
confidence=punch_confidence)

  for i, punch_box in enumerate(punch_detections.xyxy):
      ious = box_iou(punch_box[None, :], detections.xyxy)
      best_match_idx = np.argmax(ious[0])
      if detections.tracker_id is not None:
          best_id = int(detections.tracker_id[best_match_idx])
          punch_counts[best_id] = punch_counts.get(best_id, 0) + 1
```

```
  labels = [f"{cls_id} ({conf:.2f})" for cls_id, conf in zip(punch_class_names,
punch_detections.confidence)]

  annotated_frame = box_annotator.annotate(annotated_frame, punch_detections)
  annotated_frame = label_annotator.annotate(annotated_frame, punch_detections,
labels=labels)


  out_video.write(annotated_frame)

cap.release()
out_video.release()
total_time = time.time() - start_time

print("Video processing complete and saved to", OUTPUT_VIDEO_PATH)
# Summary print
print(f"\n===== Performance Summary =====")
print(f"Saved annotated video to {OUTPUT_VIDEO_PATH}")
print(f"Total processing time: {total_time:.2f} seconds")
print(f"Total frames processed: {processed_frames}")
print(f"Average inference time per frame: {total_inference_time / processed_frames:.4f}
seconds")
print(f"Effective FPS: {processed_frames / total_time:.2f}")
print(f"Average CPU usage: {np.mean(cpu_usages):.2f}%")
print(f"Average Memory usage: {np.mean(memory_usages):.2f}%")

print(f"\n===== Tracking Evaluation =====")
print(f"Total unique tracker IDs: {len(seen_ids)}")
print(f"Total ID switches: {id_switch_count}")
print(f"Track duration per ID (frames):")
for tid, duration in id_track_durations.items():
  print(f"  ID {tid}: {duration} frames")

print(f"\n===== Punch Counts per Player =====")
for tid, count in punch_counts.items():
  print(f"  ID {tid}: {count} punches")
```


NOVA
IMS
Information
Management
School