Conflict detection system for multiple humans in an airport

Group Design Project (Team 2)

Adeola Aderogba *S329155*
Claire Delgove *S359263*
Marc-Olivier Pokam *S357187*
Ruolin Liu *S366569*
Karan Kheta *S358663*
Muhammed Mehmet Unal *S365355*

Word counts without tables, titles, figures, references and appendix: around 10,000

MSC Applied Artificial Intelligence
2021-2022

Contents

1. Introduction

Airports are getting smarter – not just on the passenger side, but also on the maintenance and operations side. This is because operations are getting more complex, there are new threats from both humans and autonomous machines, and human operators are often working alongside robots in potentially safety-critical environments. Future airports will become more congested with more travellers and, thus, these environments will become hotspots for potential conflicts to break out as well as targets for terror events. An intelligent system which renders security surveillance more effective in detecting conflicts as well as reducing costs in hiring additional personnel for monitoring brings many benefits to the airport in cost and operations but also to the passengers in terms of their safety. This report details the development of a conflict detection system to be used within airports. Two different approaches are demonstrated for detecting conflict, and the results of the final model are shown. A physical hardware system is integrated with the final to demonstrate whether it is capable. An ethical review is also shown in this paper to inform the requirements of the system and ensures the system operates ethically. A number of recommendations are outlined at the end informed by the many challenges that have been faced, particularly during the integration phase. These recommendations relate to how the system should be configured to be used in an actual airport setting.

All the codes with a few demos for each approach led in the project are provided on the following GitHub link:

2. Context

2.1 Business need

Airports are already highly secure environments, however, they are some of the busiest locations for human flow. They need to deliver powerful, extensive security solutions whilst also being cost-effective and sensitive. These systems should not only apply to passengers but also for the many airport staff providing services throughout the airport. Even coming out of the pandemic, IATA expects overall traveller numbers to reach 4 Bn in 2024 exceeding levels pre-pandemic and from there airport growth will only continue to climb. Boeing even anticipates long-term growth, where the Indian airline operators will require over 2000 new single-aisle aircraft over the next 20 years. Additional passengers and growth in the future will only put more pressure on existing security systems in the airport. This will require more intelligent and smarter solutions which are able to fit into existing airport budgets and infrastructure. Menzel et al mentions that the average airport master plan has a 20-30 year horizon so building new infrastructure is not a quick and easy fix to cope with increasing number of people and aircraft (Menzel & Hesterman, 2017). Menzel also mentions the operational losses that airports are under, 66% of the world's airports operate at a loss and 92% losing money, have fewer than one million passengers a year. This combined with space being a luxury resource in an airport means that existing spaces must be pushed further in terms of their capability, whilst also finding ways to reduce financial losses and not compromise on Airport Security.

As reported by the Federal Aviation Administration (FAA), in 2021 from 1st January through to 24th May, 2,500 reports of unruly behaviour by passengers were approximately received by the FAA. Those categorised as "unruly" reached 394, compared with well under 200 for each full year of 2019 and 2020. In 2020, the rate of incidents doubled, and that trend is continued into 2021. From the IATA's Cabin Operations Safety Technical Group, an informal survey revealed one member airline reported over 1,000 incidents of non-compliance in a single week. Another report calculated a 55% increase in unruly passenger incidents based on the numbers carried. Incidents have even resulted in diversions, including a flight from Paris to Delhi. These reports of unruly behaviour are quite minor and do disrupt airport operations causing flight delays and incurring scheduling costs however there have been cases which have been much worse. For instance, the airport in Istanbul which is the third busiest in Europe after London's Heathrow and Paris's Charles de Gaulle. Its total passenger traffic was some 61 million people in 2015. The explosions occurred in the airport's international arrivals terminal involving three suicide bombers. The investigation revealed that all three fired their weapons before detonating their explosive devices. Other attacks include Brussels Zaventem International Airport, Jinnah International Airport in Pakistan, and Moscow's Domodedovo International Airport in Russia.

Error! Reference source not found. shows a bow-tie diagram to visualize what can be done to prevent conflicts. In the centre is the conflict itself, on the far left is threats that lead to the conflict and on the far right is the consequences if the event occurred. For example, if there is physical provocation, a preventative measure will be security or staff intervention before leading to the

conflict. If a system is in place to detect this provocative behaviour such as pushing, we can prevent a conflict from spiralling out of control.

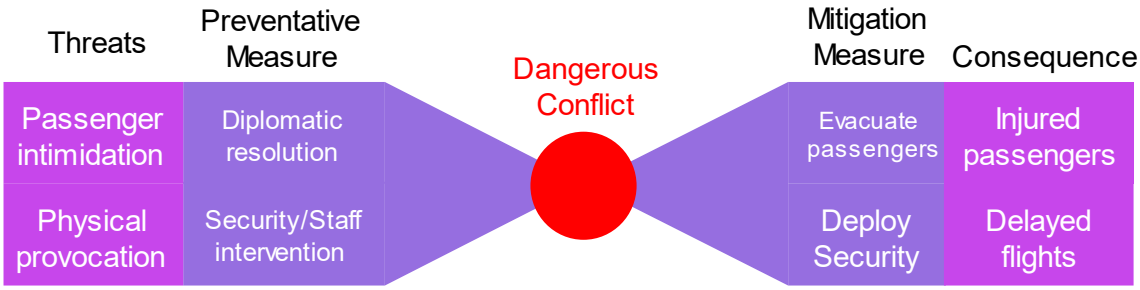


Figure 2.1: Bow-tie diagram for conflict

To quantify the benefit of a detection system, the cost for a flight delay per minute is $74.20 and with the average delay being 12.4 minutes, it costs approximately $920.08 per delay. For the impact on an airport closure, the drone scare that occurred at Gatwick airport that closed the runway for 33 hours in 2018 cost over £50m. There are also other benefits, the obvious one being improvements in passenger and staff safety, increased security awareness, alleviation in security staff monitoring and reducing risk of terror events.

The need for smarter and more intelligent security systems is apparent. A system which can spot for unruly behaviour ranging from small physical conflicts to actions which could lead to devastating terrorist events is very much needed to cope with the challenges that future airport may be faced with. These being higher passenger flows, increased congestion, and increased number of flights within already busy airports as highlighted above. The next section will detail what the conflict detection system will be required to have to operate in an airport.

2.2 Existing solutions

Currently surveillance is done with a human operator monitoring the many feeds of security cameras scattered across the airport however there has been much research in surveillance and there are many tools available for behaviour recognitions. Ben Mabrouk & Zagrouba (2018) reviews the different methods for detecting abnormal behaviour, done through behaviour representation and modelling. The paper also highlights the many challenges with abnormal behaviour detection and offers suggestions to overcome them. One of the challenges being describing the behaviour of a subject if the scene constantly changes. It also highlights that behaviour detection algorithms assume that the subject is Infront of the camera but in a real setting, the subject could be anywhere in the view of the camera. It suggests using multiple cameras to overcome this problem.With this, it also suggests that cloud computing will be needed to cope with the huge amount of data. Li et al., (2014) conducts a survey on state-of-the-art techniques for analysing crowded scenes. This paper looks at crowds rather than the individuals. Denman et al., (2015) talks about the many different intelligent surveillance techniques for pedestrian throughput, crowd size and dwell times can be used to benefit airport operations and networks. These papers have all discussed in some way detecting for anomalous behaviour however Pan et al., (2019) outlines in their paper a method to detect a fight using pedestrian key node positions to estimate a pose, it then also shows how the motion is calculated from the optical flow and if the subject exists next to the target to detect a fight. Much from this paper can be taken away about using optical flow to generalise motion for an individual.

Much research points out the vast benefits for an intelligent surveillance system such as those highlighted in Denman et al., (2015) to recognise threats early, for the protection of critical infrastructure and for support of protocols for the rapid escalation of response by capability and capacity. There is however very little research which points out to using pose detection in security surveillance.

2.3 Subject description

This section will detail the aim and objectives of the system as well as covering many of the ethical considerations that need to be considered during development of a conflict detection system.

2.3.1 Aim

The high-level aim for the project is as follows

To develop a conflict detection system utilising existing monitoring systems within airports to spot for conflicts within crowds.

Conflict in our systems means physical conflicts between people (passengers, staff, and crew). These poses include pushes, punches, and kicks. The system will also look out for weapon poses such as aiming a firearm. The system will interface with existing hardware within airports, so hardware during development is comparable to CCTV cameras used in airports.

2.3.2 Objectives

Development of the conflict detection system will take a staged approach as outlined in ***Error! Reference source not found.***. The initial objective states that the system just detects conflicts on one camera for simplicity, ethical impacts will be evaluated and implemented in our system. The third objective relates to the system dashboard which is the interface between the security operator and the system. The alarm system is the feedback given to the operator to be alerted when the system has detected conflict. In an actual airport setting, surveillance is done over the whole airport using multiple cameras, the fifth objective is about achieving that capability.

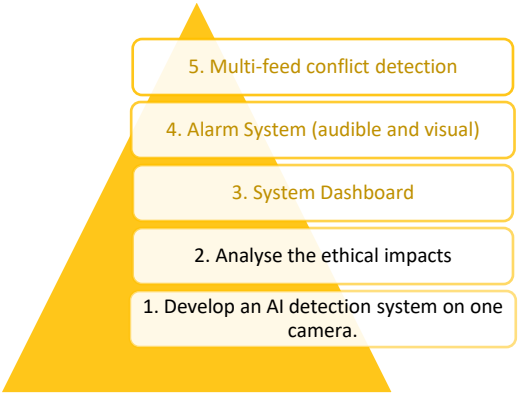


Figure 2.2: System Objectives

2.3.3 Environment considered

The airport environment is highly varied containing areas for retail, food, passageways, security points, lounges, and such so the conflict detection system should be able to handle any kind of scenario. These settings are usually well lit however the system still should be able to perform under low-light conditions for extreme cases. Some of these areas will also be far busier than others such as the security checkpoint will have high passenger flows compared to the likes of airport lounges, the system will be required to cope with the high number of potential subjects.

2.3.4 Ethical considerations

Importance of ethical considerations

Artificial Intelligence ethics is a “set of values, principles, and techniques that employ widely accepted standards of right and wrong to guide moral conduct in the development and use of AI technologies” ("Understanding artificial intelligence ethics and safety", 2019).

The aim of our system is to detect any physical conflict in the airport environment. In order to make this process as morally cooperative as possible, we need to consider all ethical factors both for and against the system. “Conflict” can be considered as subjective, and our system aims to have the highest degree of accuracy. Analysing all the possible factors that could compromise the success of our detection system is largely based on ethics. Factors such as racial bias, trust issues, cultural differences (i.e., different symbolic gestures), to name a few, are assessed to retain the integrity of our system and to make sure there are no human rights’ violations.

Ethical considerations in conflict detection system

The following points concern the ethical issues and benefits that arise from the implementation and usage of a conflict detection system. Different areas of concern are discussed and will be used to inform the design and operation of the system.

There are many issues around privacy. Many will be particularly concerned with such a system which is able to identify and hold information about them, there are many risks with this which the person can be concerned about. The first concern relates to the collection and processing of data, thus there are rules on this is handled according to the General Data Protection Regulation (GDPR) act.

Article 22 of the GDPR states that individuals have the right not to be subject to a decision that has a legal or similar effect upon them and, that is based solely on automated decision-making (without human intervention). There are some exemptions to this right; where the use of personal data is necessary to enter a contract, if the processing is authorised by law or if explicit consent is given by the data subject. Even though the detection system is used to spot for unruly behaviour and if used to identify people then that information can be used with the authorities to help for analysis post-event. However, to be in compliance with the act, then this detection system shall not personally identify the individual. It will purely be used to alert airport security of conflict occurring in that area of the airport, whether the system records footage will be specified by the operator, but the system will have that capability for post-event analysis.

Security issues

Concerns around the security of the system. Without adequate cyber safeguards in place, the system can be prone to cyber-attacks and thus be exploited. These vulnerabilities will most definitely impact the integrity of the system and whether the system can be trusted to detect for conflicts becomes questionable. Ideally the system is able to monitor different areas of the airport assisting the security operator in keeping the airport safe however if such a system provides false alarms or cyber-attacks induce nil detection, then this only makes it worse for the security operators as there will be wasted time investigating false positives or missing conflicts entirely. The design of the security framework should not be hidden, an open-source approach so that it can be audited but also the system code should be robust and reliable enough to minimise false positives.

Operational issues

A system which is able to autonomously detect conflicts renders redundant human operator. There is risk that there will be a loss of human decision making and questions on accountability. As this is purely a conflict detection system, the system will be required to only monitor for unruly behaviour and provides indication if there is a presence of a conflict.

Behaviour of people can be misinterpreted by the detection system. Children who are play fighting can be mis-classified by the detection system as a potential conflict. People from different cultures may have exaggerated gestures compared to others and people with certain disabilities could be misinterpreted by the detection system. These other cases need to be taken into consideration for the development of the system, ensuring the training set is diverse enough to spot for these other behaviours or through comparisons be able to determine if the subject is a child. Subjects with disabilities will be the extreme cases as it is expected they make a very small proportion of the tracked subjects. To deal with these ambiguous and discriminative classifications, system alerts should be verified by the operator before action is taken by the security force.

Safer environment

Conflict is inherently an act of violence which can often go unresolved/undetected before it escalates. Having a 24/7 autonomous conflict detection system is a step forward to improving the general sense of safety and security in society. Such a system can enhance the capability and perception of security personnel in keeping the airport safe.

Test for discrimination

To eliminate the opportunity for racial bias or size discrimination, using a conflict detector which just uses the posture of a human to detect for conflict achieves this. This uses the keypoints of a human body such as the noses, arms, eyes, etc to obtain the keypoints which results in a method which is indifferent to a person's race, size, demographic or background.

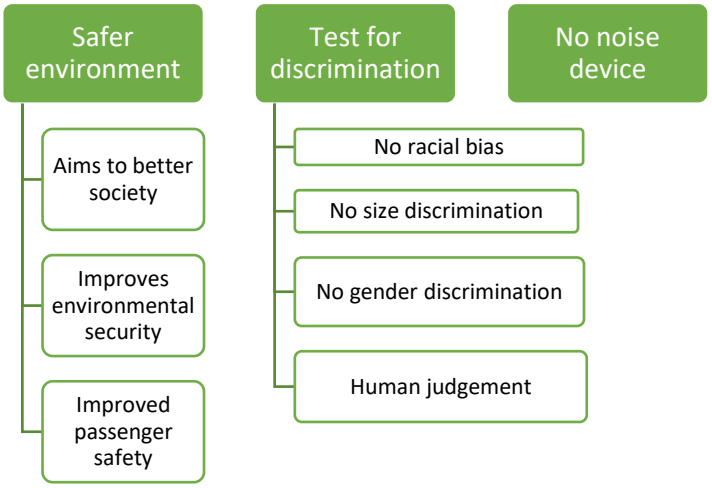


Figure 2.3: Ethical benefits

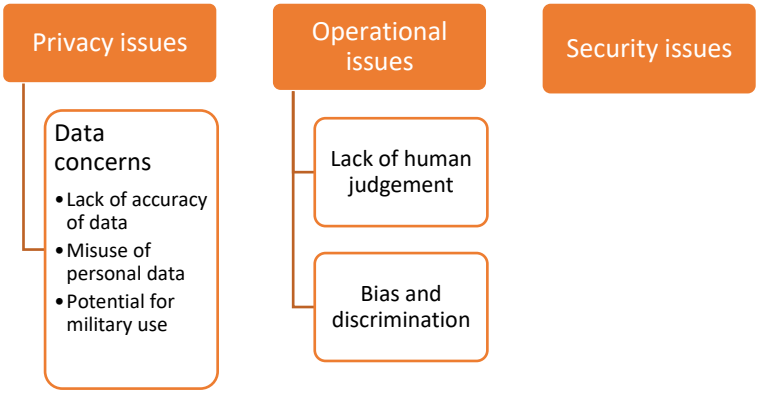


Figure 2.4: Ethical concerns

2.3.5 Requirements

Stakeholders are individuals or institutions that are directly or indirectly affected by a project, either positively or negatively. Stakeholder needs are transformed into verifiable requirements that define "what" the system does, not "how" it does. Stakeholder needs are examined, analysed, and requirements are identified, validated, and documented with these needs in mind. All requests agreed with the customer and all constraints set are considered as requirements. The requirements for achieving the project objective are considered, and the project is advanced within the requirement frameworks. Project expectations and constraints are determined.

The types of stakeholders applicable to our project are client, actors, and owners, respectively. Clients are people who directly benefit from the system. Actors are people who carry out activities for the system to work. Owners are people who finance and manage the system throughout its lifespan. The purpose and usage area of the project should be taken into consideration when determining stakeholders. Since our project will detect conflicts at the airport, potential people who are at the airport are accepted as stakeholders. The stakeholders for our project have been determined as follows. Security (client), passenger, visitors, airline staff, store staff, maintenance/developer, cleaner (these stakeholders act as actors), and finally the project owner (owner).

The Use Case is a technique used in software and system engineering to cover the functional requirements of systems. They define the interactions of stakeholders with the system.
Security expects conflicts to be detected in a short time and tries to take precautions as soon as possible. The project owner wants the system to make a lifetime profit and operate regularly. Developers maintain the system and make the necessary updates on regular periods. Finally, other stakeholders such as passengers, staff, and visitors want to feel safe and be in a safe environment. *Figure 2.5* shows the conflict detection use case diagram.

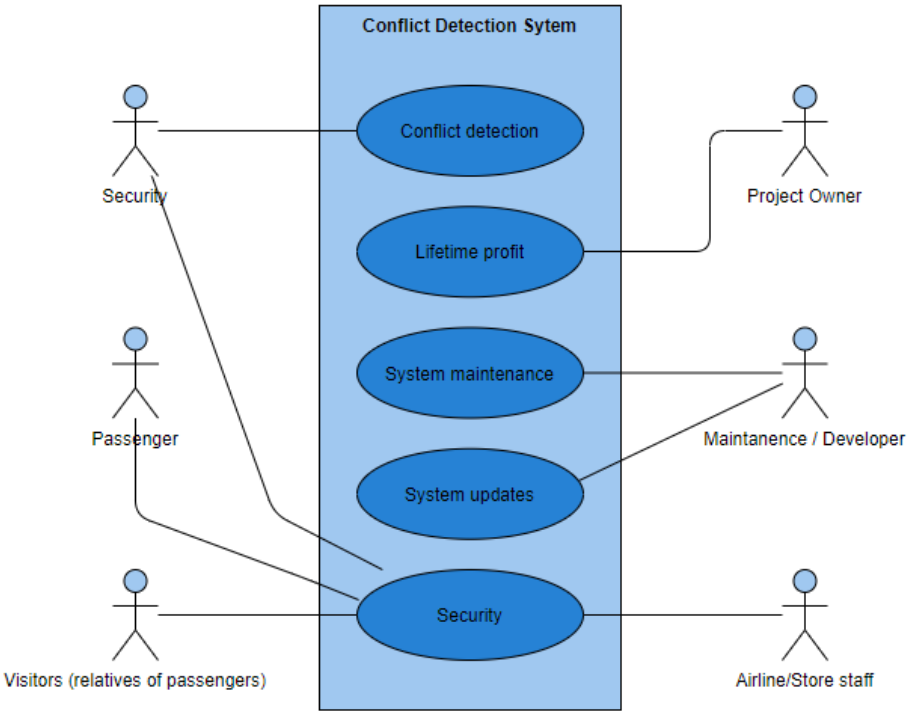


Figure 2.5: Conflict detection use cases

Project requirements are divided into different areas such as user requirements, system requirements (operational requirements, functional requirements, constructional/physical requirements), and non-functional requirements.

User requirements define the business requirements demanded from the system. Usually, these specifications are written before the system is built, as early in the verification process as possible. System owners and end-users determine these requirements.

System requirements are the details and instructions that determine how the system performs and behaves. Here, it is aimed that the system will achieve the desired success and work properly with the appropriate model. Requirements are established to ensure that the software is easy to use and in working order, and to ensure that the software and hardware work together properly to perform the intended task. System requirements are divided into three operational, functional, and constructional requirements.

Finally non-functional requirements refer to the user experience rather than the defining function of the system. These requirements assist improve the system's efficiency and displaying the system's quality.

Table 2.1 shows the project stakeholders, the project requirements, and which requirements affect which stakeholders.

		Requirements Priority	Requirements	Stakeholders							
				Security (Client)	Passenger (Actor)	Visitors (Actor)	Airline staff (Actor)	Store Staff (Actor)	Maintenance Developer (Actors)	Cleaner (Actor)	Project Owner (Owner)
	User Requirements	1	The System shall detect for conflicts within the airport environment	X	X	X					X
		1	Operators shall be notified within 5 seconds when conflict is detected and present the confidence level	X							
		2	Considering Article 4 GDPR Definitions and Article 9 GDPR Processing of special categories of personal data, the protection of data such as physical, physiological, location, cultural, and social identity of individuals shall be ensured		X	X	X	X			
		2	System source-code shall be updated remotely	X					X		X
System Requirements	Operational Requirements	2	System accuracy shall be at least 90%	X					X		
		1	The system shall detect for conflict concerning pushing, punching, kicking, and even firing	X					X		X

		1	The system shall detect the conflicts near and far from the camera.	X					X		
		2	The system shall be capable of detecting 10 people	X					X		X
		2	Depending on the hardware, the system shall record at least 20 minutes of video	X					X		
	Functional Requirements	2	The system shall be able to take at least one video feed						X		
		3	The camera to be used shall have a minimum viewing angle of 135 degrees						X		
	Constructional/	1	The system shall be protected against external attacks (flammable hardware, cutting the wire, data disconnection)						X		
	Non-Functional Requirements	1	Cost shall be at most £1000								X
		1	The conflict detection system shall work continuously for 24 hours a day	X					X		
		2	The system shall be able to detect conflicts in low light conditions	X					X		

Table 2.1: System requirements

1 ("must have")
3 ("less important")

2.4 Risk analysis

Risk assessment

The risks in our conflict detection system can be assessed after dividing the process into three categories:

1. **Building the conflict detection system**

This is the design stage of the process that involves developing the system, using machine learning and embedding the system with the hardware to create a functional AI detection device. The risks at this stage will be a combination of both physical and software.

2. **Implementing the conflict detection system**

Implementing the conflict detection system is the stage at which the system is being transported and installed in the airport environment. This process could pose many risks as there is a window of opportunity where the system is more vulnerable to security breaches and cyber-attacks as it is not fully secured in place.

3. **Using the conflict detection system**

The system being in use is potentially the riskiest part of the process as this is when it will be carrying out its primary function. This is where the system will be interacting with the users and so there are a lot more threats to the security of the system.

Risk matrix

This risk matrix (*Table 2.2*) is a guide to give ratings to each risk (*Table 2.3*), to have a better idea of degree of mitigation that we would have to put in place to combat each risk.

RISK PROBABILITY		RISK SEVERITY				
		Catastrophic A	Hazardous B	Major C	Minor D	Negligible E
Frequent 5	*Happens several times per year at location*	5A	5B	5C	5D	5E
Occasional 4	*Happens several times per year in company*	4A	4B	4C	4D	4E
Remote 3	*Incident has occurred in company*	3A	3B	3C	3D	3E
Improbable 2	*Heard of incident in industry*	2A	2B	2C	2D	2E
Extremely improbable 1	*Never heard of in industry*	1A	1B	1C	1D	1E

Table 2.2: Risk matrix

Facility / Activity	Identify the hazards	Who/what may be harmed	Risk Rating	Existing control measures	Recommendations/ further action required
Building Pose Detection System	Flammable GPU. Fire risk	Jetson Nano could contain a flammable GPU. This has the possibility of overheating. The personnel in charge of building the system (System Engineer, Embedded System Engineer) may be exposed to flammable hardware – potentially leading to a fire outbreak	2A	There is a cooling fan which will be used in the event of overheating. Personal protective equipment (PPE) should be worn at all times. All systems building and architecture should be done in a carefully controlled environment with minimal source of heat/fire	Fire extinguishers, fire alarms and firefighting personnel onsite in the unlikely event of a fire outbreak
	Exposure to live wire in the building of system	The HDMI cable may be exposed to a live wire, which could lead to malfunction of the system and possibly cause an electrocution	2A	Do not use any frayed cords and always assume a wire is live and do not store any machinery or equipment near them	All wires should be earthed (insulated) to protect any live wires
	Exposed circuit on the development board	System engineer could be exposed to an open circuit, leading to electrocution	5C	System should have a switch which shuts down system as soon as circuit is exposed	Circuit breakers should be included in the system as well as ground fault circuit interrupters.
	Trip hazard from data connection cable	System engineer could trip on a very long cord during the building	4B	Necessary barricades and warning signs should be in place. Avoid the use/need for	Health workers and first aid should be on site in case a trip occurs

		of the system		unnecessarily long cable. Connection cables should be carefully hidden	
Implementing System in Airport	Faulty/broken wiring or equipment	There could be a faulty wire in the system which can cause shock/electrocution	4B	Do not use any frayed cords and always assume a wire is live and do not store any machinery or equipment near them	Regular checks and maintenance should be done to make sure all faulty wiring/equipment are replaced
	Trip hazard from cable wires	Embedded systems engineer could trip on a very long cord during the building of the system	4B	Necessary barricades and warning signs should be in place. Avoid the use/need for unnecessarily long cable. Connection cables should be carefully hidden	Health workers and first aid should be on site in case a trip occurs
	False detection during implementation	The system could potentially mistake the personnel implementing the system as one of the four actions (punching, pushing, kicking, shooting)	5D	Fine-tune the pose detection system by using a wide range of datasets to ensure highest degree of accuracy	Have extra set of personnel/surveillance to ensure conflict is happening before security intervenes
	Worn electrical cords	Wires could be worn out over time due to frequent use	4C	Regular checks and maintenance should be done to make sure all worn out wires are replaced	Replace wires twice a year whether they are worn out or not
Using Pose Detection System	False detection/false alarm	The system could potentially mistake a couple of innocent passers-by as one of the four actions	5D	Fine-tune the pose detection system by using a wide range of datasets to ensure highest degree of accuracy	Have extra set of personnel/surveillance to ensure conflict is happening before security intervenes

		(punching, pushing, kicking, shooting)			
	The memory on the GPU could be lost/too small	All the data stored in the GPU could be lost/overwritten due to overworking the system or small memory	3E	Having regular maintenance checks to ensure the hardware is running smoothly will prevent this from occurring	Having a replacement/back up GPU on standby to be immediately put in if this event occurs
	Cyber attack	The system could be attacked by hackers who could leak very private data	5A	Software should be installed in the system which can prevent malware/virus and any other bug that could lead to a data leak	Only highly trained and qualified personnel should be able to access the system
	Biased algorithm	The pose detection system could be biased to a certain class (weight, race, height etc.)	5C	The pose detection system is carefully designed to identify keypoints on the body, e.g., nose, mouth, left ear, right arm etc. and so there is no opportunity for bias	Fine-tune the pose detection system by using a wide range of datasets to ensure highest degree of accuracy
	System failure	The system could shut down unexpectedly during the process due to overloading of the hardware/software	5C	Having regular maintenance checks to ensure all parameters look like they are supposed to and checking for errors will prevent this from happening	Having replacement hardware on standby to be immediately put in if this event occurs

Table 2.3: Risk analysis

3. Project management

3.1 The team

The team is comprised of a Project Manager, System Engineer, Embedded System Engineer, Machine Learning Engineer, Data Scientist and R&D/ Quality Assurance manager. This is organised in a flat team structure lead by the project manager. Details of the team are shown in *Table* .

Role	Name	Responsibilities
Team Leader Project Manager	*Karan Kheta*	• Provide team direction and drive • Compile the business need and conduct research • In charge of the communication with the teaching team. • Organise meetings and logistics of testing. • Conduct ethical review • Hardware procurement • Maintain good team moral • Keep an eye on the timeline • Responsible for the clarity of the report and presentations
System Engineer	*Muhammed Mehmet Unal*	• Able to act and think with a system engineering perspective • Work on the dataset • In charge of the risk analysis, requirements performance, use cases, system design… • Design the testing plan • Quality assurance • System compliance
Embedded System Engineer	*Ruolin Liu*	• Lead on building the physical system • Lead on the entire integration of the system
Machine Learning Engineer	*Claire Delgove*	• Lead the code for the model • Link the model and the physical system in the integration
Data Scientist	*Marc-Olivier Pokam*	• Lead the code for the model • Responsible for the data management, data processing and analysis • Responsible of the keypoints : Neural Network approach
Research & Development/ Quality Assurance Manager	*Adeola Aderogba*	• In charge of finding new ideas or improvements • Create risk assessment and risk matrix • Check for quality at each development stage • Responsible for ethics considerations • Create Gantt chart • Work on dataset image labelling

Table 3.1: Roles and responsibilities

3.2 Strategy

Management of the project was driven highly from the design review stages. This is how the Gantt chart[1] was formulated and to help break down the individual tasks required to progress through parts of the project. The Gantt chart reflects every stage in the process of creating our system – from the initial design phase to the final device we created. Weekly progress meetings with the team were held using Microsoft Teams to keep track of the project development and make aware of any issues/concerns that may have risen throughout the project development. Upon completion of the preliminary design stage, the team was broken up into three groups. One group to focus on the model of the system, another group on the hardware and the third group on capturing data and many of the system engineering practices. The group was organised in a way which allowed a subsequent team member to be diverted to any part of the group which required extra support, these de-risks any possibilities of delays and minimise team member stresses.

3.3 Collaborative tools

For effective team collaboration, a number of tools are outlined in *Table 3.2* highlighting how each tool was used for the development and management of the conflict detection system. These were either used internally (within the team) or used externally (for demonstrations and public sharing).

Tool used	Function	Intended to
Teams	- Sharing documents - Planning meetings	Internal
WhatsApp	- Fixing the last updates regarding a meeting - Sharing quick updates, e.g., news regarding achievements or problems - Contacting someone who is missing at a meeting, Informing of a late	Internal
Google Drive	- Sharing dataset sources - Sharing the final dataset - Sharing pieces of codes - Sharing large files	Internal + External
GitHub	- Consolidating the final codes and required files (Model + Dataset)	Internal + External

Table 1.2: Tools for collaboration

[1] Appendix 11.1 – Gantt Chart

4. Design process

4.1 System architecture

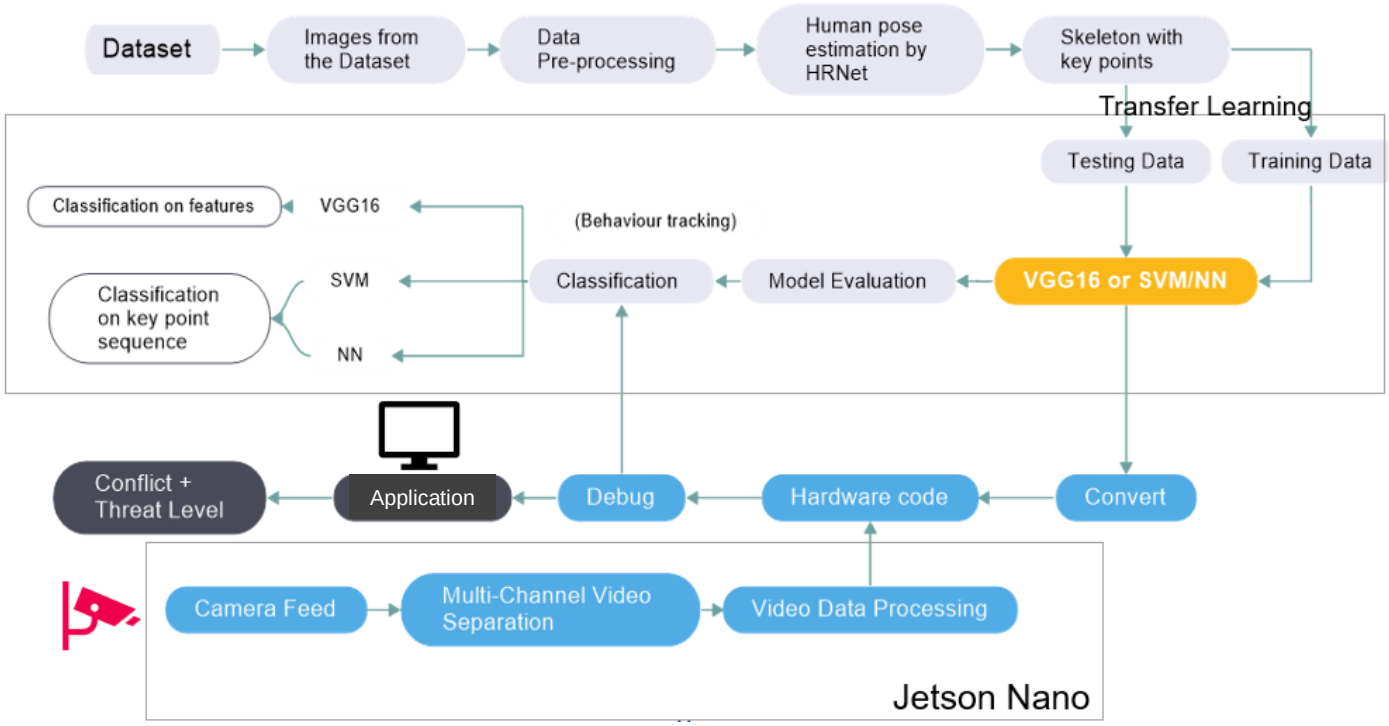


Figure 4.1: System architecture

The architecture shown on *Figure 4.1* illustrates the first acquisition process of the data, from the dataset to the step where the data can be sent to the model for the training after the segmentation (keypoints plotting). After the training achieved, the same kind of data used for the training can be used for the testing interference which leads to the model evaluation for each model (explained later on) and the selection of the most performant one to classify fight poses (*VGG16 features approach* or *Keypoints Approach* with *SVM* or a neural network). After the model is ready, it can be integrated with the hardware to get the predictions on it.
The entire hardware system flowchart is shown on the button. The video is first captured by the camera and transmitted to the development board, where it is transformed into suitable video data after the video processing module. It is then sent to the converted model for processing and waiting for the surveillance screen to be displayed. At the completion of the debug, a distinction can be made between fight and normal behaviours. Fight behaviours include: punching, pushing, kicking and shooting.

4.2 Dataset creation

4.2.1 Sources of data

In the project, we found various datasets from different sources for human posture analysis. The data sets we use should include the following actions for the class 'fight': pushing, kicking, punching, and shooting. At the beginning of the project, we examined different datasets such as DCSASS, CAVIAR, G3D, Real Life Violence Situations Dataset. Later, we decided to use our own dataset (with images taken from YouTube videos and the internet), UT-Interaction Semantic Description of Human Activity (SDHA 2010), MPII Human Pose Dataset datasets as they are more comprehensive and suitable for the project purpose.
The important points in the dataset are that the whole body is present, and the images are clean. There should be no ambiguous images in the data set, and if any, they should be eliminated.

4.2.2 Dataset conception

We created our dataset using YouTube videos and images we found on the internet (*Figure 4.2*). The videos we use include fight movies, boxing matches, and street fights. There are actions taking place indoors and outdoors in our dataset, and this dataset includes punching, kicking, pushing, and shooting actions. We labelled our dataset as fight and normal actions.

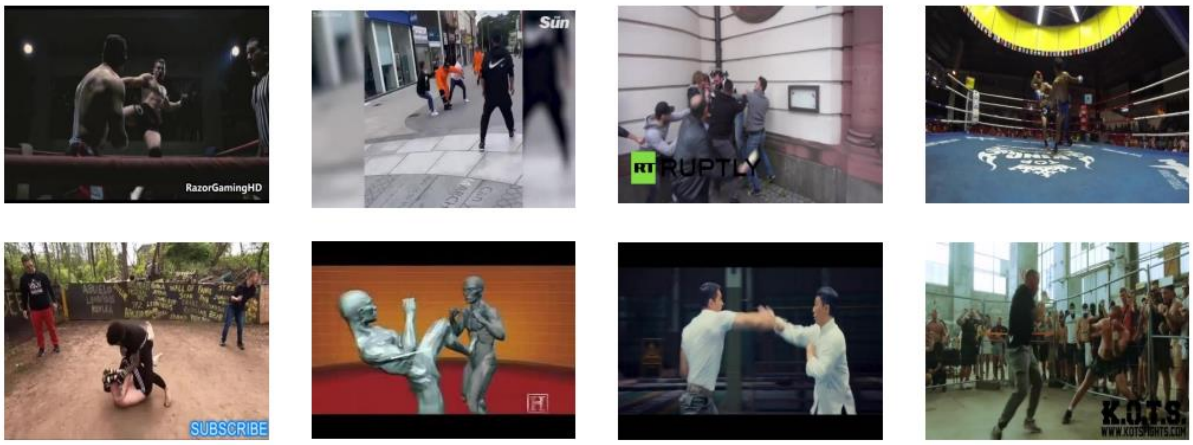


Figure 4.2 Sample images from our own dataset

- **UT-Interaction Semantic Description of Human Activity (SDHA 2010)**

The UT-Interaction dataset contains videos of continuous executions of 6 classes of human-human interactions: shake-hands, point, hug, push, kick, and punch (*Figure 4.3*). Ground truth labels for these interactions are provided, including time intervals and bounding boxes. There is a total of 20 video sequences whose lengths are around 1 minute. Each video contains at least one execution per interaction, providing us 8 executions of human activities per video on average.

Figure 4.3: SDHA 2010 dataset sample images

❖ **ISR-UoL 3D Social Activity Dataset**

This is a social interaction dataset between two subjects. This dataset consists of RGB and depth images, and tracked skeleton data (i.e. joints 3D coordinates and rotations) acquired by an RGB-D sensor. It includes 8 social activities: {handshake, greeting hug, help walk, help stand-up, fight, push, conversation, call attention} (*Figure 4.4*). Each activity was recorded in a period around 40 to 60 seconds of repetitions within the same session at a frame rate of 30 frames per second.

Figure 4.4: ISR-UoL dataset sample images

❖ **MPII Human Pose Dataset**

MPII Human Pose dataset (*Figure 4.5*) is a state-of-the-art benchmark for the evaluation of articulated human pose estimation. The dataset includes around 25K images containing over 40K people with annotated body joints. The images were systematically collected using an established taxonomy of everyday human activities. Overall, the dataset covers 410 human activities, and each image is provided with an activity label. In this dataset, we will use data that includes punch, kick, push, the pistol shot actions.

Figure 4.5: MPII Human Pose dataset sample images

4.2.3 Final dataset

At the very beginning, the first approach was to realize a four-class classification with the following actions: pushing, punching, kicking, and shooting. However, the first results of the models (explained later) showed that it was sometimes difficult to make a proper difference between these actions since they are quite close to each other's, particularly between punching and pushing. Therefore, it has been decided later to realize a two-class classification with '*Fight*' and '*Normal*' classes. The Fight class gathers four fighting poses previously selected.

The final dataset is composed of several widely used datasets and custom images, to assure the diversity of situations our algorithm encounters. In that sense, images from a controlled environment, taken from the ISR-UoL dataset, will form half of our data. The second half is from our custom dataset, G3D and MPII Human Pose Dataset, forming images with diverse environments with varying factors that impact our model: luminosity, background, actors' clothing.

To exploit the ISR LCAS dataset, we had to work on the labelling first. Indeed, the labelling available with the original dataset would only classify the 40 seconds to 1minute long video (1700 frames for 1 action) even if the action is repeated several times with pauses, resulting in one label for images

that do not necessarily share the same poses (during said video, the actors are performing several times the action, with inactive time between each). Labelling the data more precisely allows us to only consider relevant images, avoiding confusion with ambiguous ones (*Table 4.1*).

session	action	start frame	end frame	actor	act
2	5	7	15	1	0
2	5	139	150	1	1
2	5	619	626	1	1
2	5	772	807	1	1
2	5	887	956	1	0
2	5	1436	1500	1	0

Table 4.1: ISR-UoL enhanced labelling sample

With that we hope to cover a wide range of cases, to have a robust model that can perform in various conditions. The SHDA dataset will serve as a validation dataset thanks to the good diversity in the behaviours: multiples people in the sequences, involved or not with the action.

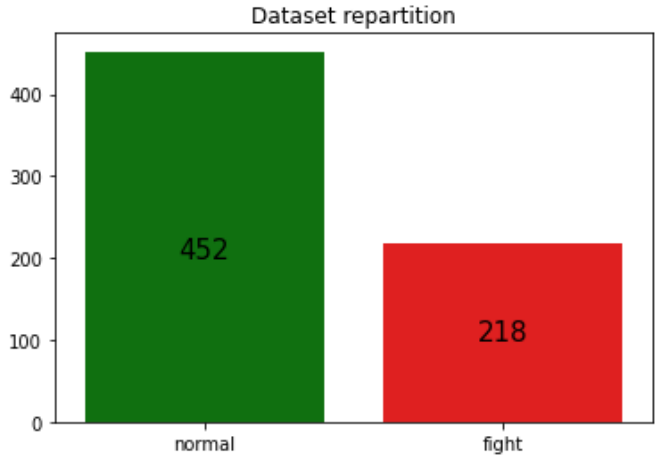


Figure 4.6: Final dataset repartition

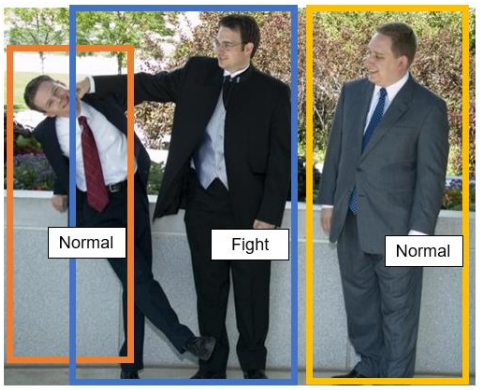


Figure 4.7: Image cropping example

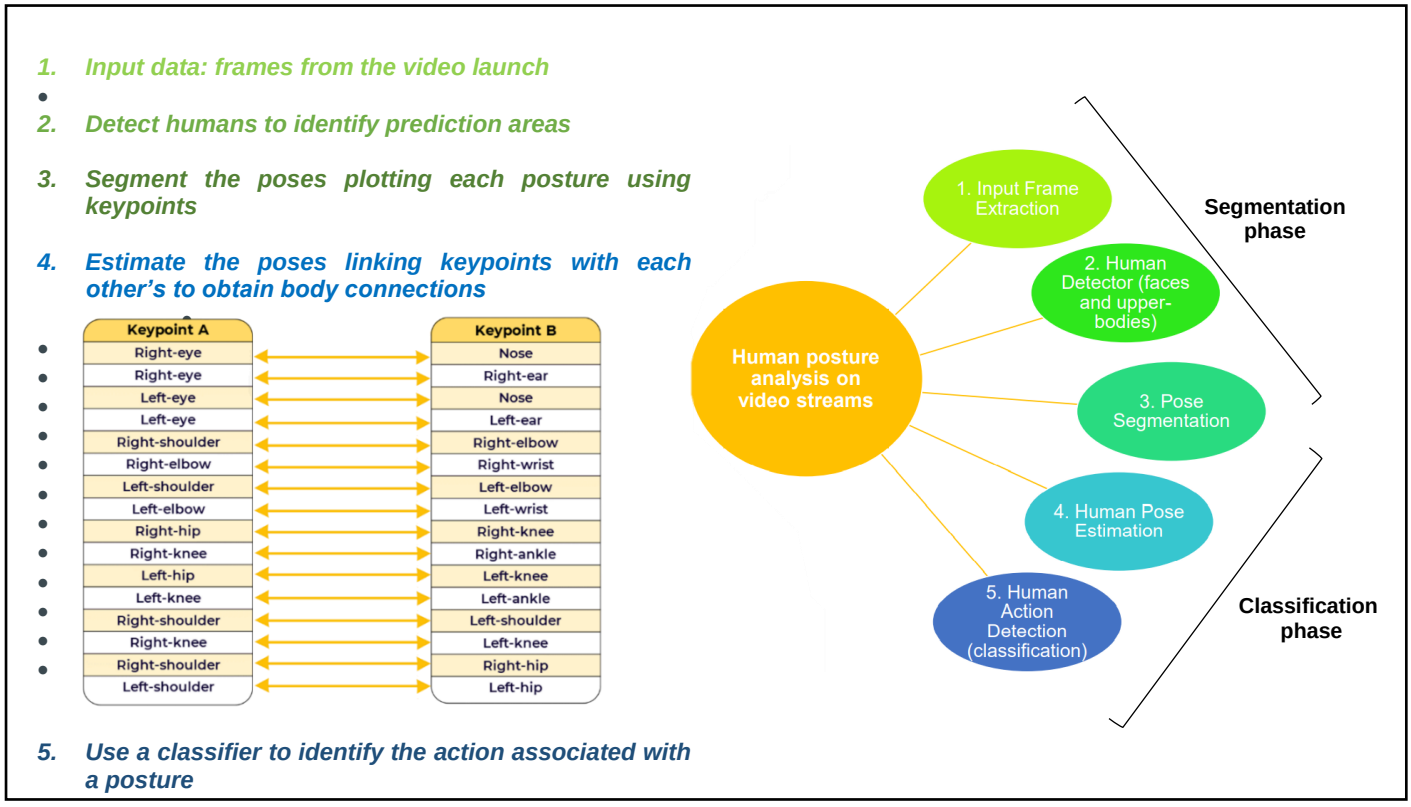
For each future data pre-processing, the dataset's images (*Figure 4.6)* are cropped considering human bounding boxes (*Figure 4.7*). After that, each bounding box is classed in the folder which corresponds to its label (*Fight* or *Normal*).

4.3 Model conception

4.3.1 Model design

Strategy of AI

The modelling part is dedicated to conceiving the main algorithm for the human pose analysis. It is composed by a human pose detector and a classifier to predict the different behaviours according to the postures. The whole algorithm will be linked with the hardware later to obtain a physical system able to identify potential dangers within a small crowd in an airport. The main working steps for the model are illustrated on *Figure 4.8*. These steps will be undertaken under two main phases: first the segmentation part, and then the classification.



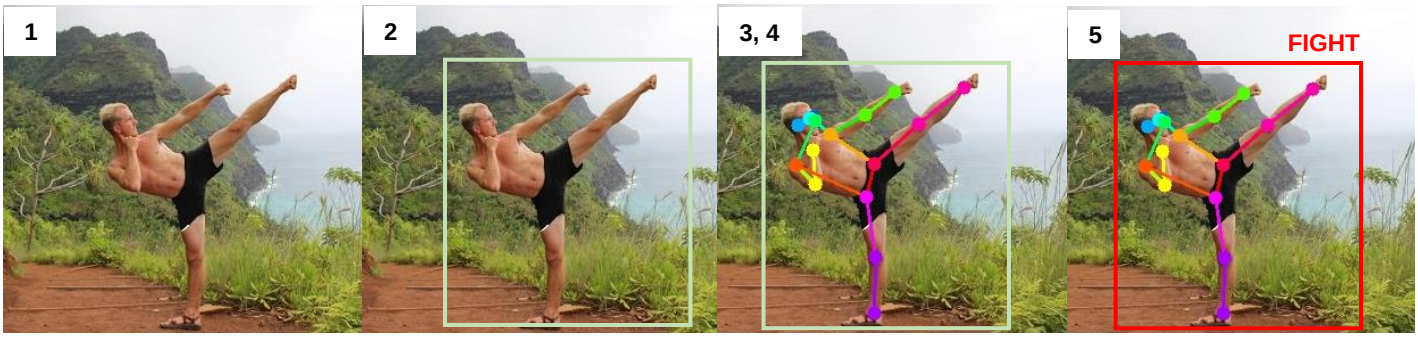


Figure 4.8: Model strategy

Potential solutions

Before beginning any algorithm, it is necessary to analyse different potential methods which could be employed. This is why several ideas regarding each phase of the previous frame have been gathered on *Table 4.2*.

	METHOD	APPROACH
2. Human Detector	• Face detector with upper body detector - *HOG feature pyramid* - *Haarcascade* Cascade Filtering *(Viola and Jones)* - *Multi-Level Gaussian Representation* (2D Gaussian heatmaps)	• Bottom-up * or Top-down ** approach for the upper-body detection (face, torso, …)
3. Pose Segmentation **4. Human Pose Estimation**	• *OpenPose* library • *MaskRCNN* • *DeepCut* library • *AlphaPose* library • Silhouette-based detector • Kalman Filter • YOLO • *HRNet* Neural Network • *Using the Pictorial Structure (CRF)* presentation • *Using the Cardboard* presentation	• 2D pose estimation for the training • 3D estimation on video streams
5. Human Action Detection	• Pose classification with *Hough Orientation Calculator (HOC) (*extracting the orientation of each limb from the CRF image) • ANFIS architecture (Neural Network) • SVM classifier • ResNet + FastAI (Neural Network) • RNN (Neural Network) • VGG16 (pre-trained) • Inception V3 (pre-trained) • Cascade of CNN (3D) as Pose regressor (AlexNet) • Dynamic Bayesian Network • LSTM (Long short-term memory) Neural Network	• Fully convolutional model for 3D human pose estimation on video streams • Apply transfer learning for the CNN (using a pre-trained model) • Semi-supervising or no supervised training • Comparison of poses with suspicious action dataset • Using images or video sequences

Table 4.2: Potential model solutions

*Bottom-up *: First, detect a body one-by-one. Then, estimate the pose for the body detected.*
*Top-down **: First, detect all the bodies. Then, allocate the limbs belonging to distinct persons.*

Before selecting any of these solutions in the system design, a consistent time has been spent to analyse which could be implement efficiently according to the project criteria.

Main model frame

After doing some research regarding the potential pipeline for a conflict detection algorithm, the main frame for the model part has been drawn (*Figure 4.9*). It consists of obtaining a segmentation dataset (dataset with the keypoints plotted) using pictures of fighting people, that will be used to train a classifier (orange path). After that, the inference part (green path) will be undertaken using a real-time video sequence with a camera, to predict the violent behaviours (*Pushing, Punching, Kicking*, or *Shooting*) on the sequence.

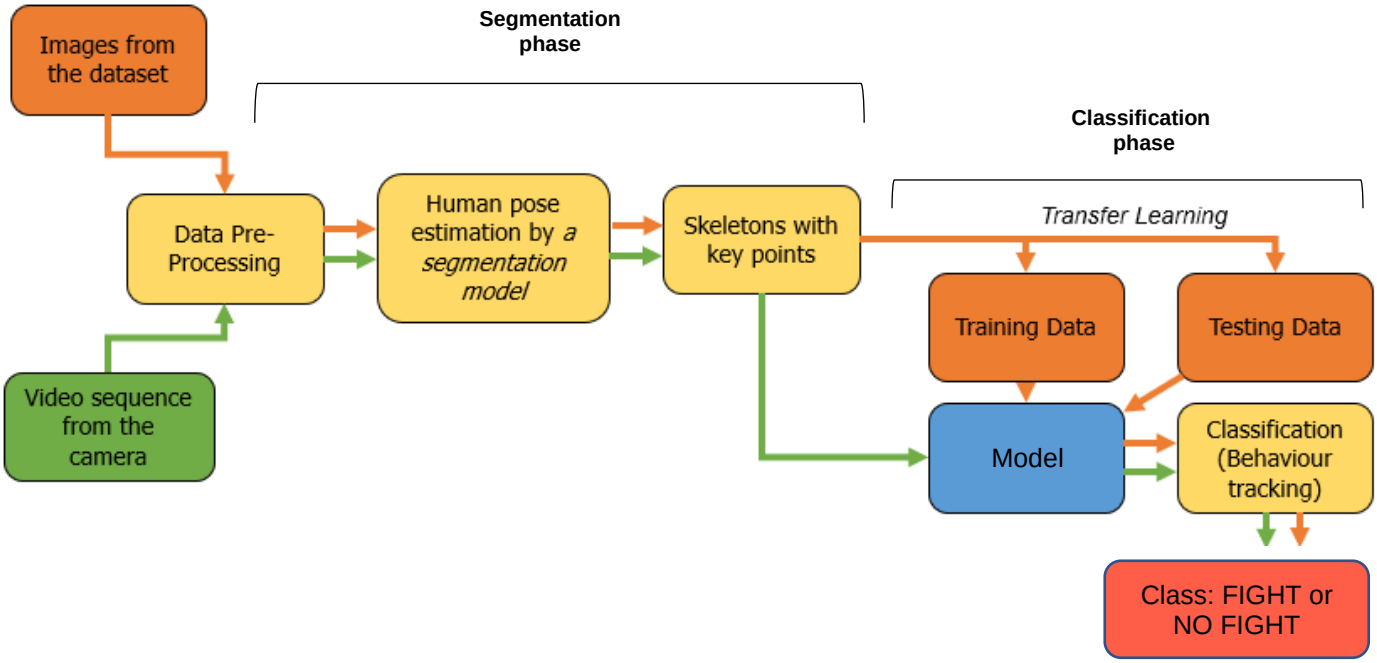


Figure 4.9: Potential model solutions

4.3.2 Segmentation phase

After that the first approach decided was to use the multi-human pose detector library *OpenPose* to recognize and estimate peoples poses. Indeed, this real-time pose detector is well-known for being quite performant in terms of speed and precision. Using a bottom approach, it can detect 135 keypoints on a human body. However, the positions have changed following the discovery of an even more adequate and performant top-down approach: a **High-Resolution Net Neural Network** (*HRNet*). On *Table 4.3*, it is visible that this neural network can achieve in a way better results on the average precision (AP) than *OpenPose* for different values of *Object Keypoint Similarity* (equivalent of Intersection over Union (IoU) in object detection) which is the distance between the predicted keypoints and the ground truth points. *HRNet* outperforms with 82,0 general standard average precision over the other models keeping an adequate computation complexity.

Table 2. Comparisons on the COCO test-dev set. #Params and FLOPs are calculated for the pose estimation network, and those for human detection and keypoint grouping are not included.

Method	Backbone	Input size	#Params	GFLOPs	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	AR
Bottom-up: keypoint detection and grouping										
OpenPose [6]	–	–	–	–	61.8	84.9	67.5	57.1	68.2	66.5
Associative Embedding [39]	–	–	–	–	65.5	86.8	72.3	60.6	72.6	70.2
PersonLab [46]	–	–	–	–	68.7	89.0	75.4	64.1	75.5	75.4
MultiPoseNet [33]	–	–	–	–	69.6	86.3	76.6	65.0	76.3	73.5
Top-down: human detection and single-person keypoint detection										
Mask-RCNN [21]	ResNet-50-FPN	–	–	–	63.1	87.3	68.7	57.8	71.4	–
G-RMI [47]	ResNet-101	353 × 257	42.6M	57.0	64.9	85.5	71.3	62.3	70.0	69.7
Integral Pose Regression [60]	ResNet-101	256 × 256	45.0M	11.0	67.8	88.2	74.8	63.9	74.0	–
G-RMI + extra data [47]	ResNet-101	353 × 257	42.6M	57.0	68.5	87.1	75.5	65.8	73.3	73.3
CPN [11]	ResNet-Inception	384 × 288	–	–	72.1	91.4	80.0	68.7	77.2	78.5
RMPE [17]	PyraNet [77]	320 × 256	28.1M	26.7	72.3	89.2	79.1	68.0	78.6	–
CFN [25]	–	–	–	–	72.6	86.1	69.7	78.3	64.1	–
CPN (ensemble) [11]	ResNet-Inception	384 × 288	–	–	73.0	91.7	80.9	69.5	78.1	79.0
SimpleBaseline [72]	ResNet-152	384 × 288	68.6M	35.6	73.7	91.9	81.1	70.3	80.0	79.0
HRNet-W32	HRNet-W32	384 × 288	28.5M	16.0	74.9	92.5	82.8	71.3	80.9	80.1
HRNet-W48	HRNet-W48	384 × 288	63.6M	32.9	**75.5**	**92.5**	**83.3**	**71.9**	**81.5**	**80.5**
HRNet-W48 + extra data	HRNet-W48	384 × 288	63.6M	32.9	**77.0**	**92.7**	**84.5**	**73.4**	**83.1**	**82.0**

https://www.researchgate.net/figure/Comparisons-on-the-COCO-test-dev-set-Params-and-FLOPs-are-calculated-for-the-pose_tbl1_331343676

Table 4.3: Comparison of network performance tested on the COCO dataset

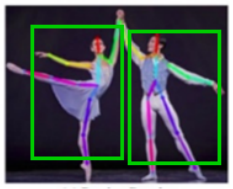
HRNet

HRNet is a **High-Resolution Net Neural Network** for multi-human pose estimation. It is pre-trained with COCO dataset which consists in 200,000 images and 250,000 person instances labelled with 17 keypoints (210 epochs: 50-60 hours). The approach employed is a top-down approach which firstly detect people on an image using a *Faster RCNN*, secondly put a bounding box area around each person, and finally estimate the keypoints configurations with the bounding boxes (*Figure 4.10*).

Human Detection

- *Pytorch* Model **Faster RCNN Resnet 50** as Human Detector to delimit the area of study for the neural network
- Prediction of people on the image
- Drawing the corresponding bounding boxes on the image

Pose Estimation



- Getting the pose estimation prediction
- Drawing the keypoints and the skeleton on the image

Figure 4.10: HRNet pipeline

Concerning the architecture of the network (*Figure 4.11*), this begins with a high-resolution sub-network at the first stage. After that, it adds high-to-low resolution subnetworks one by one and connects all the layers in parallel to obtain multi-scale fusion. Each of the high-to-low resolution representations receives information from other parallel representations over and over, leading to rich high-resolution representations.

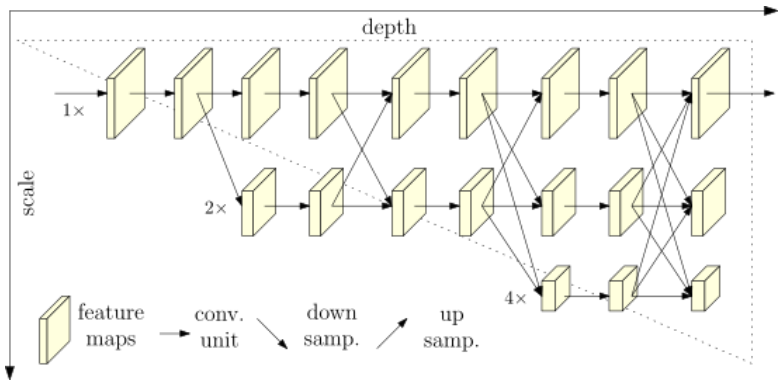


Figure 4.11: HRNet archiecture

The pose estimation is based on a few deconvolutional layers added over the last convolution stage on a backbone network, *ResNet* to estimate heatmaps from deep and low-resolution feature maps (*Figure 4.12*).

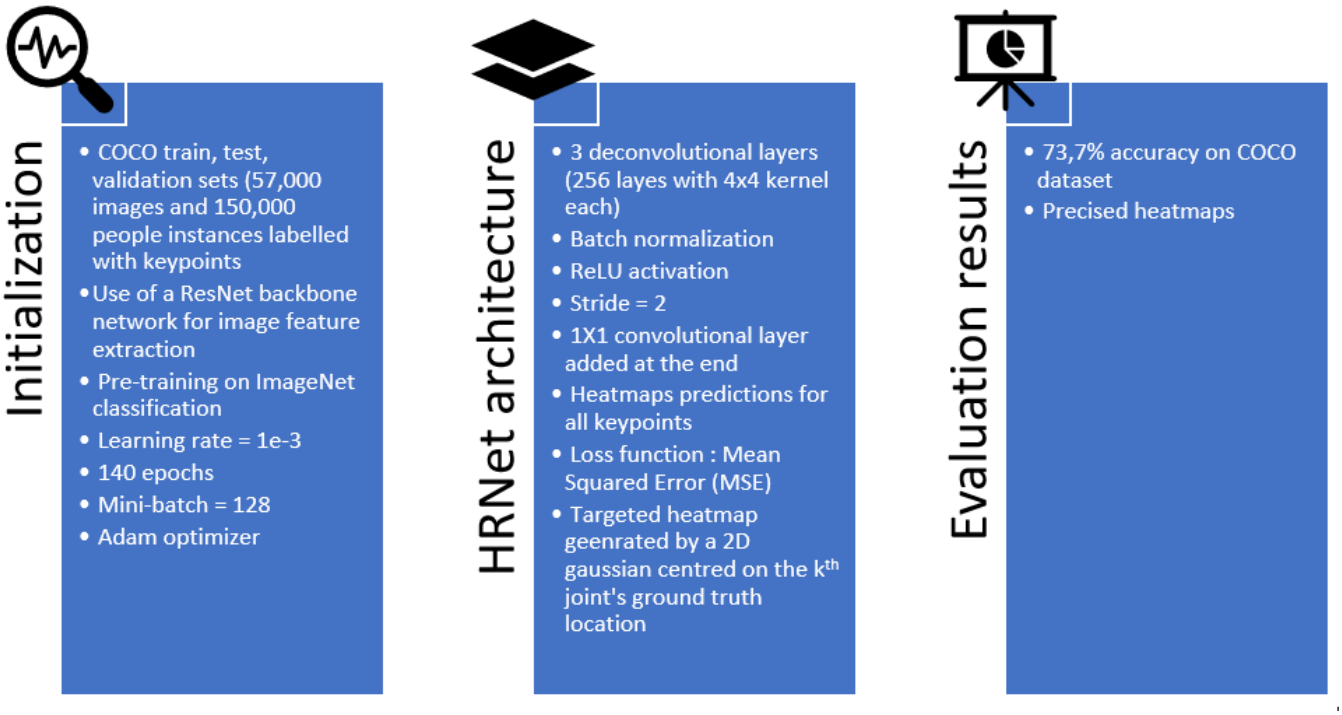


Figure 4.12: HRNet steps in more details

Thus, *HRNet* can obtain more precise heatmaps compared to *OpenPose,* maintaining high resolution representation of the input data with an efficient computation complexity. Therefore, *HRNet* has been chosen because it can provide more accurate results. However, the most performant form of *HRNet* has not been selected (*HRNet W-48*). A lighter version has been chosen instead (*HRNet W-32*) in order to avoid computation time troubles.
HRNet enables to obtain a segmented dataset to train the pre-trained classifier, such as plotting the keypoints on bodies on a video launch. The detector will thus delimit the area of study for the future action classification on each person.

Performance

The results obtained for the segmentation part are very accurate (*Figure 4.13*). *HRNet* can detect efficiently multiple people on a picture or on a video stream, thanks to its human detector at the top of its architecture. It obtains high precision segmentations on bad luminosity. Additionally, it is very accurate regarding the different bodies' angles, correctly associating the different body parts to each person, and considering the occlusions between people. Regarding the depth, the model can detect people which are quite far away from the objective. The only limit encountered is that the model needs to have a view of the full body to be fully performant and to plot the feet keypoints properly.

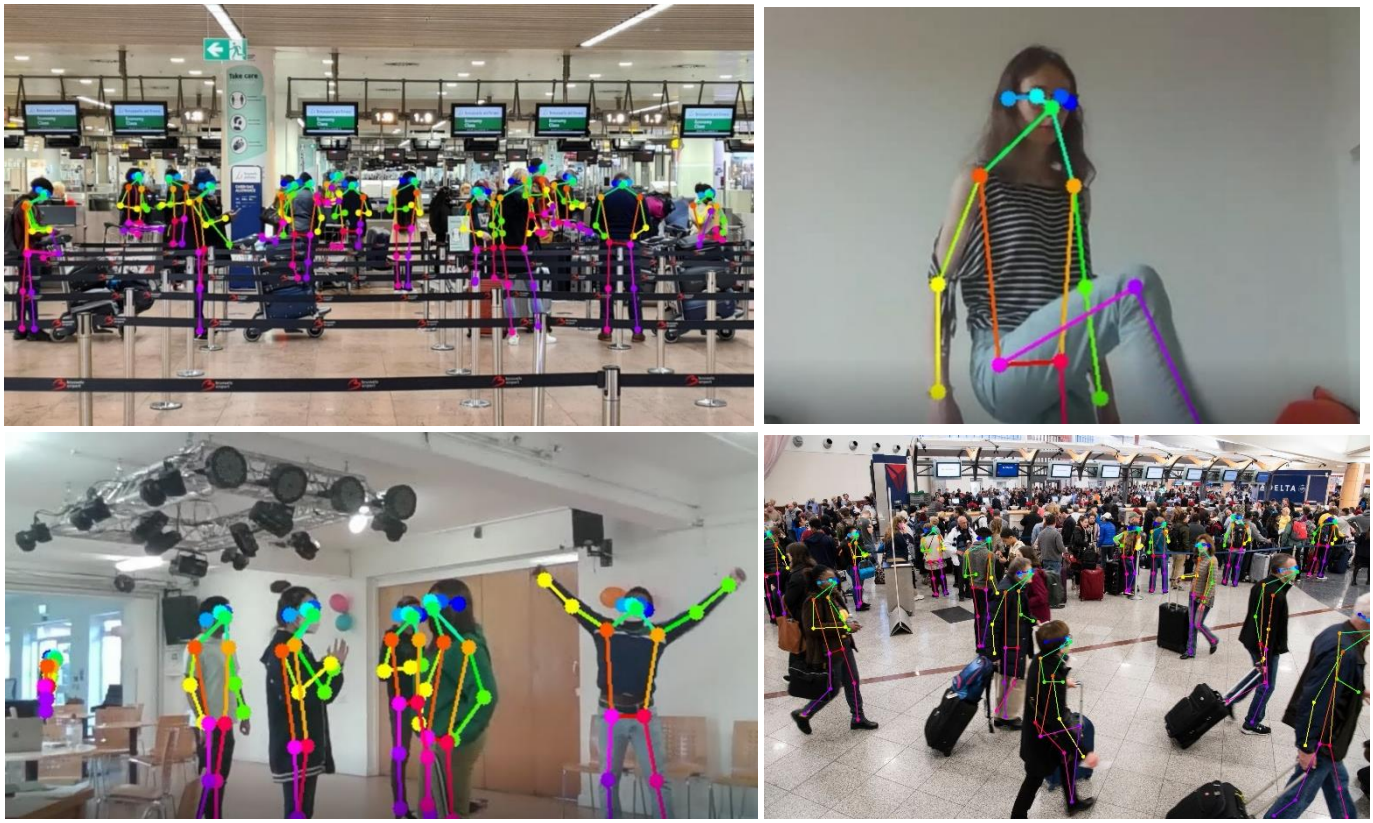


Figure 4.13: HRNet results

Limitations

Even if the model is very performant, sometimes when there are a lot of people gathered at the background to the point where it is a bit ambiguous to distinguish each person one by one, the model can't predict the poses for the people concerned (*Figure 4.13*).

4.3.3 Classification phase

To elaborate an efficient classifier, two different approaches have been led: the **features approach** using proper segmented images into the training of the VGG16 model, and the **keypoints approach** using keypoints stored into a CSV file after the segmentation to train a classifier (SVM or Neural Network) (*Figure 4.14*). Both approaches are led in supervised learning and using only pictures for the training.

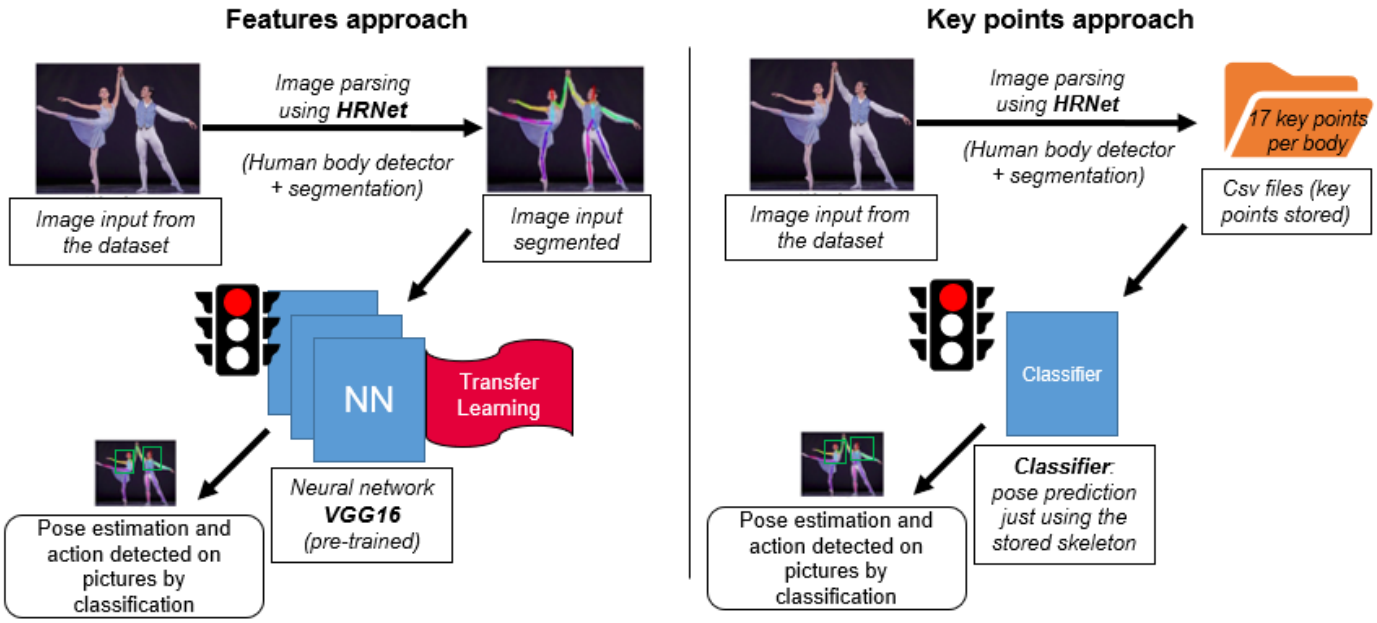


Figure 4.14: Approaches illustration

4.3.3.1 Features Approach: VGG16 Model

The first approach consists in using pictures' features for the training. As it is described on *Figure 4.15*, after obtaining the segmented images from the dataset, these ones are sent into a pre-trained neural network, in order to obtain an appropriate human behaviour classifier by transfer learning.

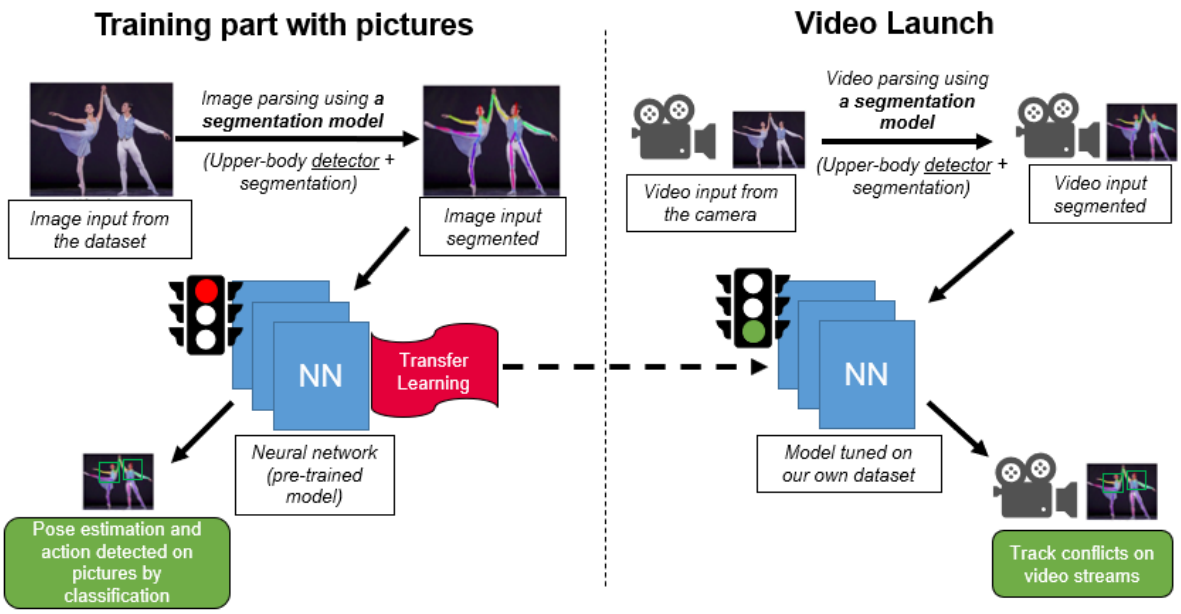


Figure 4.15: Pipeline applied for the features approach

Pre-processing

The pre-processing is the first step required in the process to adapt the data properly to use it as an input in the training. The input data needs to be an image from the dataset which is segmented containing only one person on it. Thus, the images with several people on it are delimitated with the *HRNet* human detector tracking and cropped in several pieces (*Figure 4.16*).

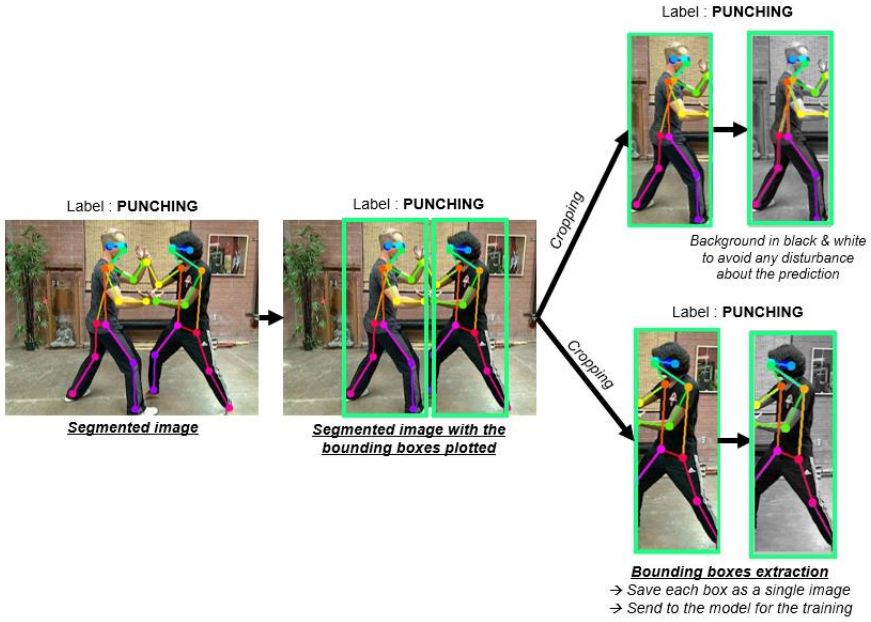


Figure 4.16: Data pre-processing for the features approach

In order to get best precisions, and to not bother the model with the useless background, the keypoints have been plotted on a black background (*Figure 4.17*) with the aim to focus only on the keypoints in the prediction.

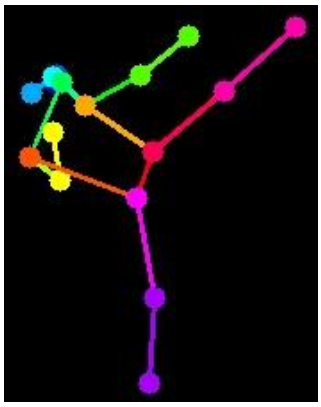


Figure 4.17: Keypoints plot on a black background

After that, the cleaned images can be saved into pickles to be used later in the training after cropping and segmentation.

Training

Neural Networks can be used to classify pictures according to the elements which want to be detected, analysing the features of the image.

A CNN contains some convolutional, pooling and fully connected layers (number depending on the amount of data). The last layer is the output which gives a probability of an action detection on an image (as much as the number of possible classes) in a classification problem. The final layer gives a probability value ranging from 0 to 1.

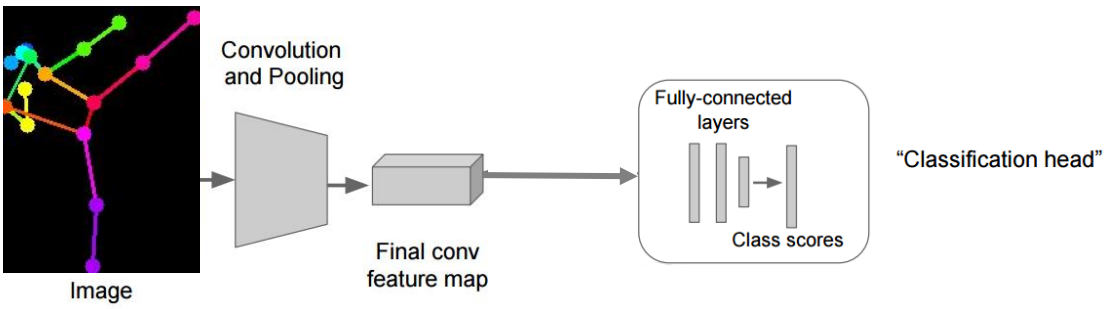


Figure 4.18: Pipeline used for the features approach training

The convolutive neural network model requires training using training images with the labels known. During the training, weight parameters will be modified to learn the key characteristics of the images. These are initialized randomly, then adjusted empirically as a dataset is processed to improve accuracy. The number of parameters being very large, training is very expensive and takes a lot of time. Indeed, the whole issue of a good prediction lies in the ability to model to define useful characteristics to detect in the image. Then, a distance is used to calculate the loss between the predicted label and the ground truth.

However, it is not necessary to build a CNN in its entirety to develop a prediction desired. Indeed, the advantage of this type of algorithms is that it is scalable and adaptable for other predictions. Transfer Learning, which corresponds to this transposition of the algorithm for a new prediction, consists in retaining the induced convolution layers and replacing the fully connected prediction at network output (*Figure 4.19*).

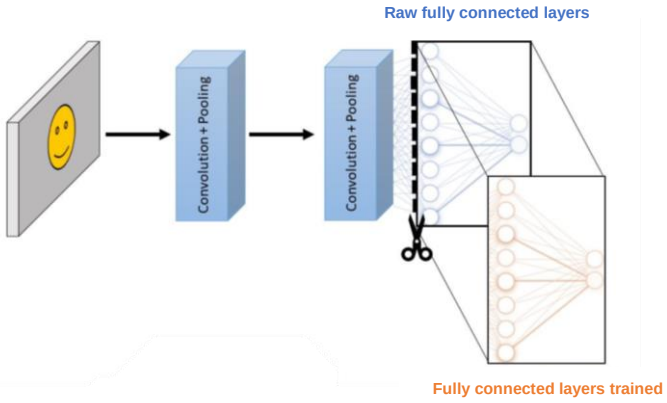


Figure 4.19: Transfer learning principle

Since the dataset is quite small with only 670 pictures, it would not be adequate to build a model from scratch. Consequently, transfer learning is applied with the pre-trained model ***VGG16***, which are widely used and recognized for their good performance.

The pre-trained model chosen for this approach is a *VGG16* model (*Figure 4.20*). Other pre-trained models such as *InceptionV3, Darknet, Resnet* or *MobileNet* could also been used for this task. But because *VGG16* is known for its good performances on images' prediction, it has been kept for this approach.

VGG16 is a version of the convolutive neural network VGG-Net. This version consists of several layers, including 13 convolution and 2 fully connected, and therefore must learn the weights of 15 layers. It takes input a colour image of size 64X64 pixels and can realize a classification in order to predict a human action on a segmented image.

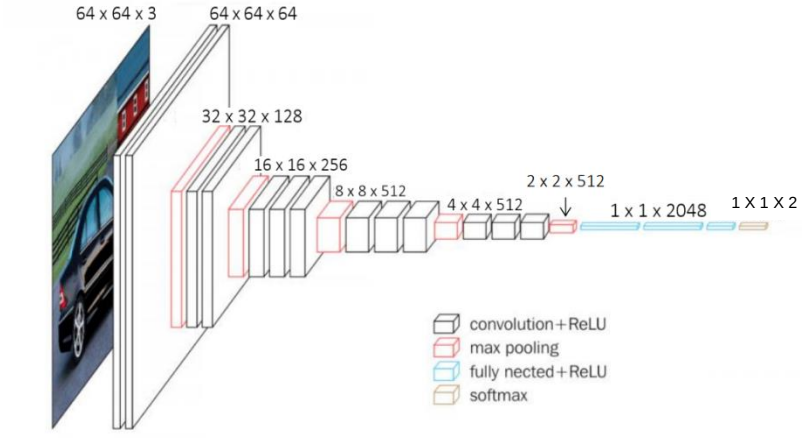


Figure 4.20: VGG16 architecture

To apply transfer learning, the 13 convolution layers are kept, and the fully connected layers are replaced with 2 new ones (*Figure 4.21*), returning a vector of size 2, which contains the predicted labels for the testing images. For this new network, the training is only done on the two new fully connected layers. This means:

(2,048 *4096+4096) + (4096*4096+ 4096) + (4096*2+2) = 25,182,210 weights to retrain

1st FC layer *2nd FC layer* *Final prediction*

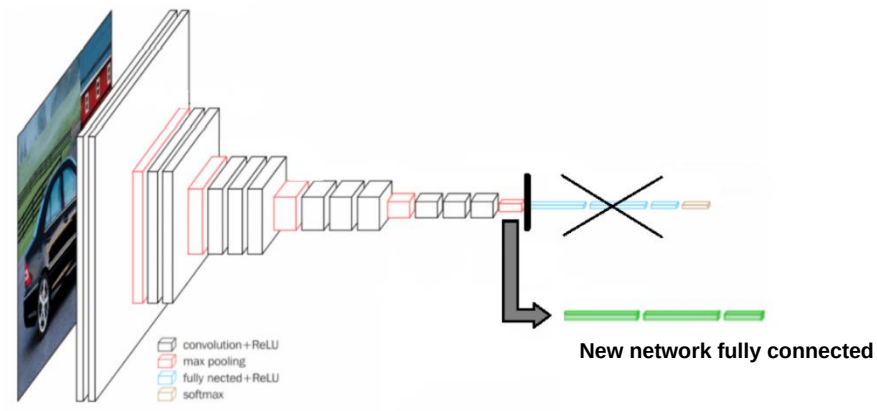


Figure 4.21: VGG16 architecture with transfer learning considerations

```
In [2]: CNN.summary()
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 64, 64, 3)]       0

 block1_conv1 (Conv2D)       (None, 64, 64, 64)        1792

 block1_conv2 (Conv2D)       (None, 64, 64, 64)        36928

 block1_pool (MaxPooling2D)  (None, 32, 32, 64)        0

 block2_conv1 (Conv2D)       (None, 32, 32, 128)       73856

 block2_conv2 (Conv2D)       (None, 32, 32, 128)       147584

 block2_pool (MaxPooling2D)  (None, 16, 16, 128)       0

 block3_conv1 (Conv2D)       (None, 16, 16, 256)       295168

 block3_conv2 (Conv2D)       (None, 16, 16, 256)       590080

 block3_conv3 (Conv2D)       (None, 16, 16, 256)       590080

 block3_pool (MaxPooling2D)  (None, 8, 8, 256)         0

 block4_conv1 (Conv2D)       (None, 8, 8, 512)         1180160

 block4_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

 flatten (Flatten)           (None, 2048)              0

 fc1 (Dense)                 (None, 4096)              8392704

 dropout (Dropout)           (None, 4096)              0

 fc2 (Dense)                 (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 dense (Dense)               (None, 2)                 8194

Total params: 39,896,898
Trainable params: 25,182,210
Non-trainable params: 14,714,688
_________________________________________________________________
```

Frozen convolutional layers: not trained

Fully connected layers: trained with the dataset

Figure 4.22: VGG16 layers in details

```python
# Model
model = VGG16(weights="imagenet", include_top=False, input_shape= [64, 64, 3])
# print(model.summary())

for layer in model.layers:
   layer.trainable = False

output_vgg16_conv = model.output

x = Flatten(name='flatten')(output_vgg16_conv)
x = Dense(4096, activation='relu', name='fc1')(x)
x = Dropout(0.3)(x)
x = Dense(4096, activation='relu', name='fc2')(x)
x = Dropout(0.3)(x)
x = Dense(2, activation='softmax')(x)

CNN = keras.models.Model(inputs=model.input, outputs=x)
CNN.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# print(CNN.summary())

X = X / 255
y = tf.keras.utils.to_categorical(y, num_classes=None, dtype='float32')
history = CNN.fit(X, y, epochs=5, verbose=1, batch_size=64)
```

Figure 4.23: Code detailed for the transfer learning part

The *ReLU* function activation allows to separate negative values in the convolution output and prevent the exponential growth in the computation.
In the algorithm, the loss function must be minimized using weights and parameters optimization. Since action recognition is a classification problem, it is needed to use an adequate loss function. In this case, *categorical cross entropy* suitable for classification is used.

Evaluation

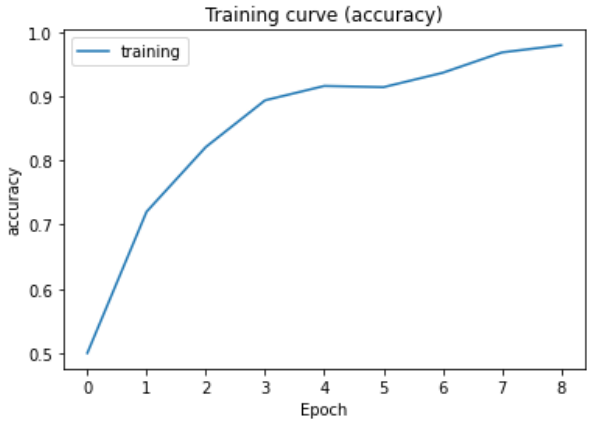


Figure 4.24: Training curve for the accuracy

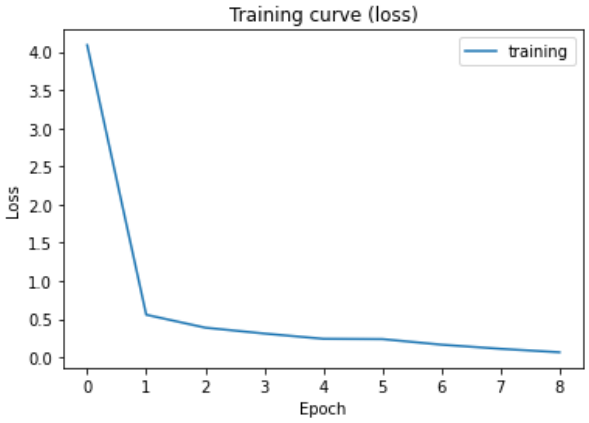


Figure 4.25: Training curve for the loss

Even if the model's results seem to be quite precise (*Figure 4.24 and 4.25*), it is essential to check the model's performance regarding a potential overfitting or underfitting for the features approach. Indeed, the built architecture must be trained properly, that is, not too little, not too much. In fact, a repetitive training is done by successive iterations (epochs). At each epoch, the dataset is fully exploited to learn.

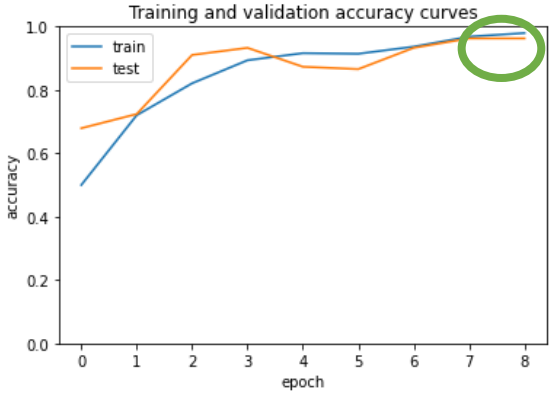


Figure 4.26: Learning curves for the accuracy

Figure 4.27: Learning curves for the loss

After conducting a cross validation (*Figure 4.26 and 4.27*), the curves indicate that for the 8 epochs configured there is no overfitting. Indeed, the curves are very close and fluctuate little (0.1 average magnitude). 8 epochs is the optimal value since after that, the algorithm begins to overfit. Therefore, at this stage, the final accuracy obtained is **97,9%** for the training data, and **96,3%** for the unseen data (*Figure 4-28).*

```
Epoch 1/9
9/9 [==============================] - 2s 161ms/step - loss: 4.0887 - accuracy:
0.5000 - val_loss: 0.6482 - val_accuracy: 0.6791
Epoch 2/9
9/9 [==============================] - 1s 116ms/step - loss: 0.5582 - accuracy:
0.7201 - val_loss: 0.5183 - val_accuracy: 0.7239
Epoch 3/9
9/9 [==============================] - 1s 110ms/step - loss: 0.3890 - accuracy:
0.8209 - val_loss: 0.2578 - val_accuracy: 0.9104
Epoch 4/9
9/9 [==============================] - 1s 110ms/step - loss: 0.3121 - accuracy:
0.8937 - val_loss: 0.1811 - val_accuracy: 0.9328
Epoch 5/9
9/9 [==============================] - 1s 110ms/step - loss: 0.2431 - accuracy:
0.9160 - val_loss: 0.3010 - val_accuracy: 0.8731
Epoch 6/9
9/9 [==============================] - 1s 110ms/step - loss: 0.2392 - accuracy:
0.9142 - val_loss: 0.2754 - val_accuracy: 0.8657
Epoch 7/9
9/9 [==============================] - 1s 110ms/step - loss: 0.1664 - accuracy:
0.9366 - val_loss: 0.1771 - val_accuracy: 0.9328
Epoch 8/9
9/9 [==============================] - 1s 110ms/step - loss: 0.1122 - accuracy:
0.9683 - val_loss: 0.1237 - val_accuracy: 0.9627
Epoch 9/9
9/9 [==============================] - 1s 110ms/step - loss: 0.0668 - accuracy:
0.9795 - val_loss: 0.1232 - val_accuracy: 0.9627
```

Figure 4.28: Training process

Predictions and limited performances

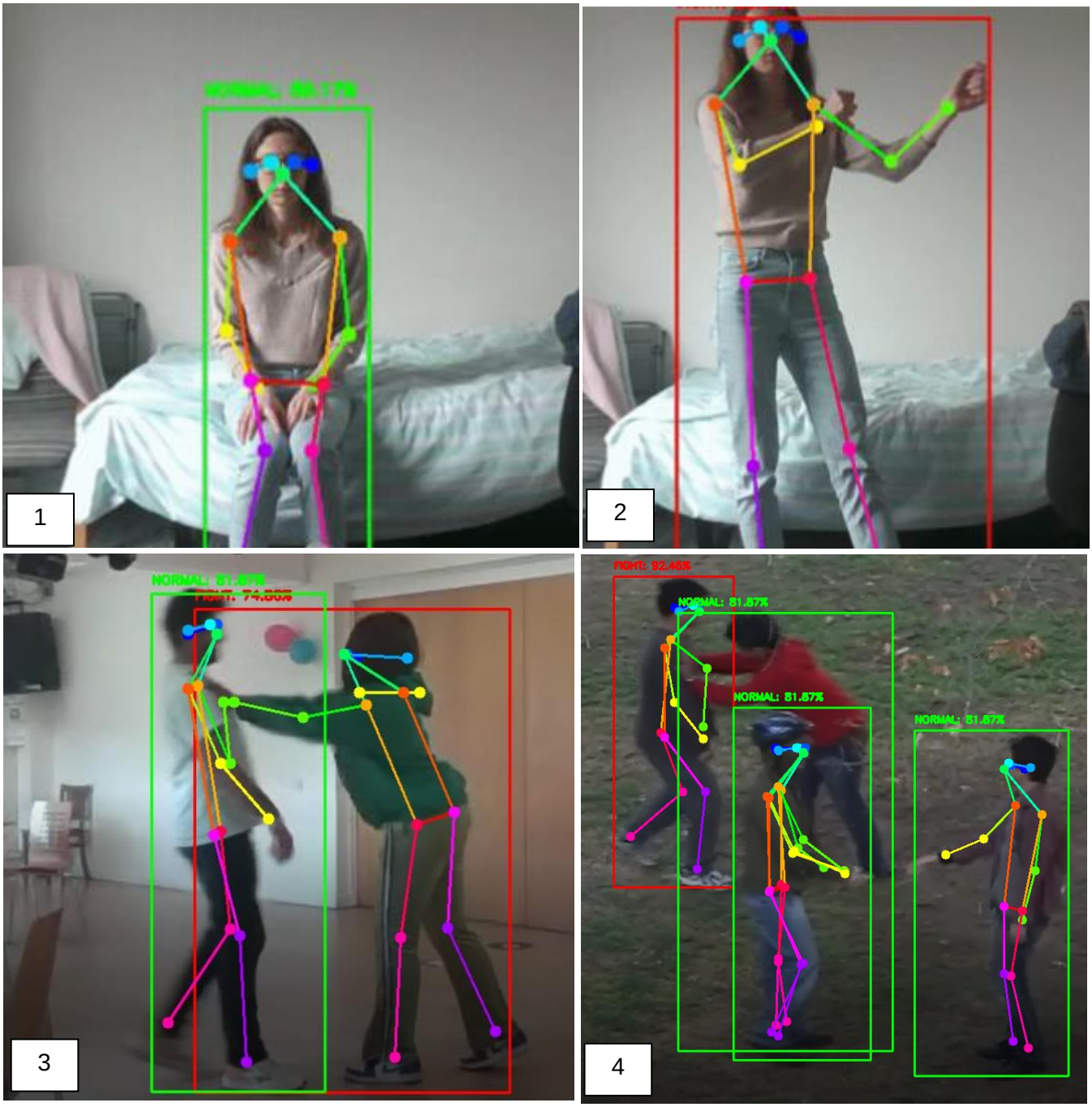


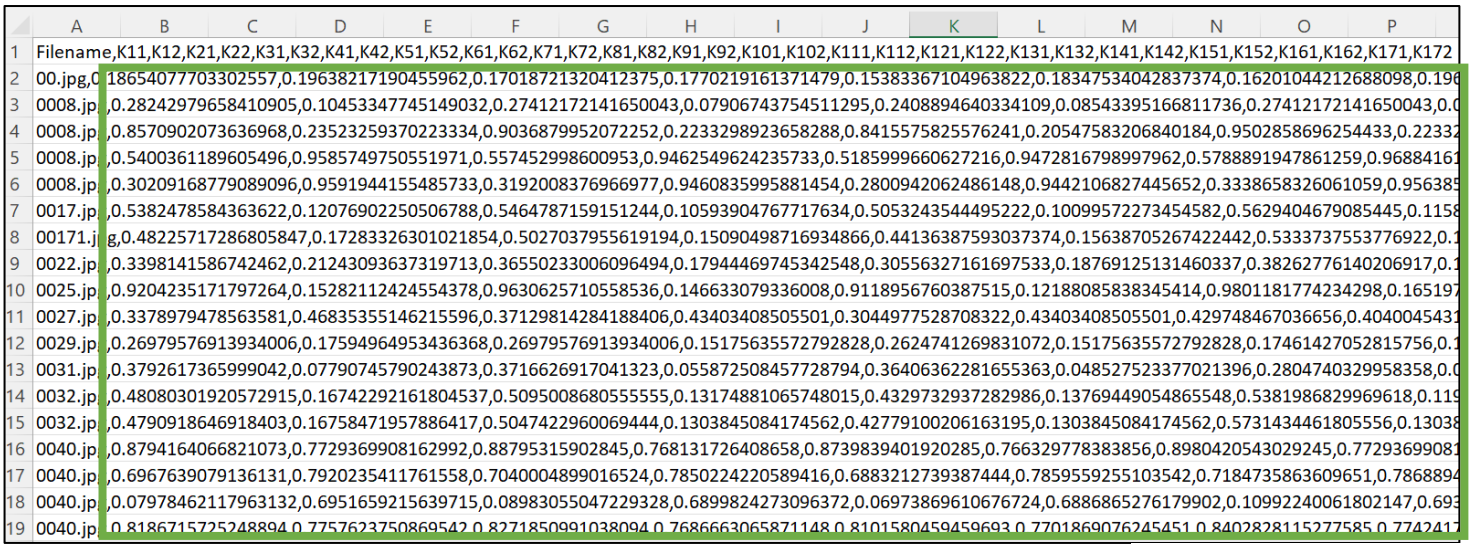
Figure 4.29: Frames extraction using VGG16

The VGG16 model provides average results. Indeed, some fighting poses which should appear as obvious are sometimes not detected by the model or considered slightly as a fight (*Figure 4-3-3-16*). The overall results are not precise enough, but the major problem is that the predictions take also a consistent time to be done. More, the video stream is not smooth and fluid. Indeed, even with a standard GPU, this features approach can't be lead just using frame by frame predictions and a dataset of pictures. It would be necessary to consider previous frames in order to predict an entire movement and not a succession of poses. This could be achieved using an LSTM model, still with a dataset of pictures, or changing the dataset with video sequences.

4.3.3.2 Keypoints Approach: SVM or Neural Network (NN)

Pre-processing

In the same way as for the features approach, the pre-processing for the keypoints approach requires to crop the images if several peoples are present on it. But the main difference here is that the 34 keypoints are not plotted visually on the images anymore, but they are stored in a CSV file (*Figure 4.30*).



Keypoints stored

Figure 4.30: CSV file sample

In the CSV, keypoints are normalized considering the height and the width of each cropped image. Thus, each keypoint value is between 0 and 1.
After that, the CSV will be read as a dataframe without the filename for the training (*Figure 4.31*)

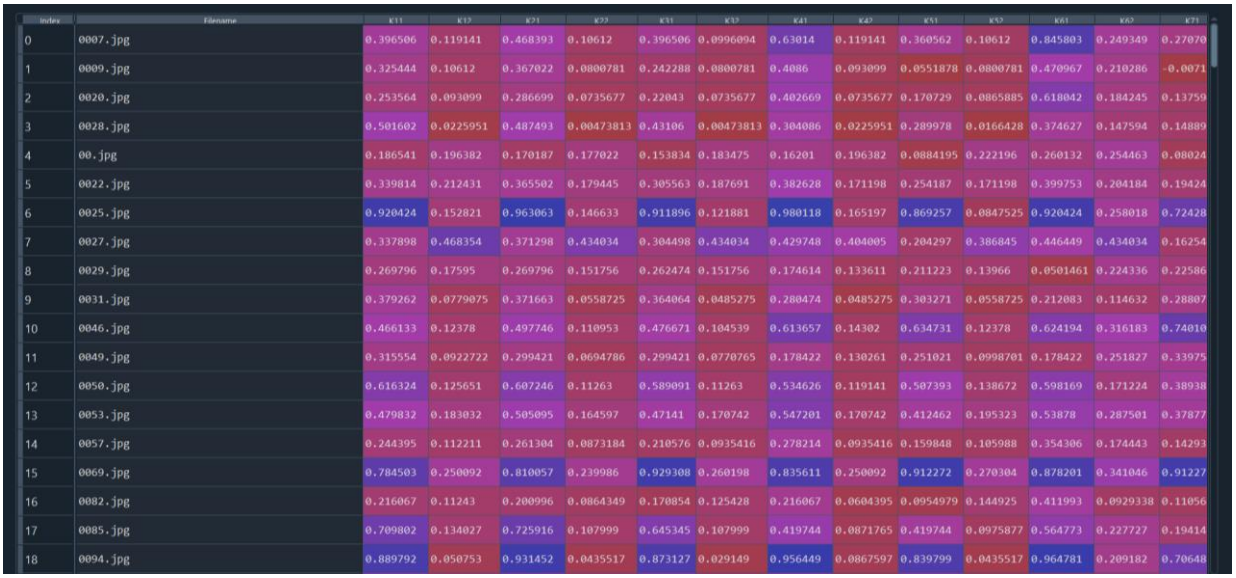


Figure 4.31: Dataframe example

During the inference, the model can be applied for prediction on images or video stream. But in order to apply properly the SVM or Neural Network classifier on video launch, it is required to normalize the keypoints not just according to the size of each bounding box, but also considering the size of the image frame. This is why before normalizing the keypoints by the bounding box's height and width, it is needed to centralize each bounding box (*Figure 4.32*).

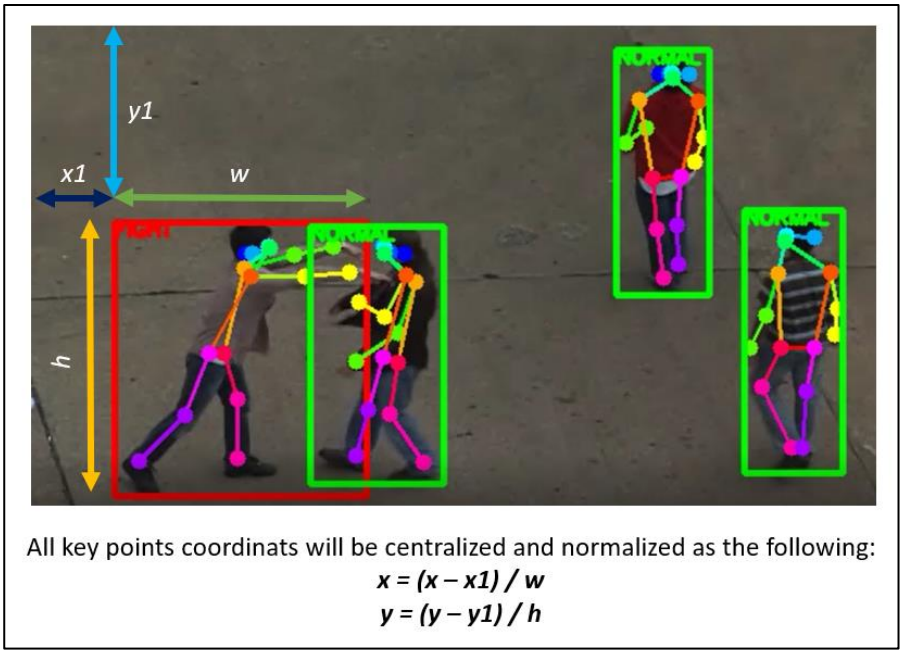


Figure 4.32: Data normalization example

The relevance of this normalization is that it is invariant to scale. It only keeps the "shape" of the skeleton, without considering the position in the image or the size. A proof of this is provided in the appendix.

SVM Classifier

For the training, two classifiers have been experimented and compared to choose the most performant one. The two selected are SVM and Multi-Layer Perceptron since they are known to be very accurate.

- **Training**

SVM (*Support-vector machine*) is chosen because it is a supervised learning model very adequate and robust for multi-class classification in high dimensional spaces.
It is set with a radial basis function kernel, a polynomial function of degree 3, and a regularization parameter equals to 1 thus a smaller margin will be accepted if the decision function is better at classifying all training points correctly.

- **Evaluation**

Evaluation metrics for this approach must be selected carefully. Indeed, it would be irrelevant to choose the *Accuracy* metrics. Because the dataset is unbalanced, *Accuracy* could be high (for instance, high capacity to predict Class 0), but not representative of the real capacity of the classifier, which could have more difficulty to predict Class 1. Instead, the *Precision* metrics will be prevailed to know the proportion of Class 1 (Fight) well predicted by the classifier.

Before explaining the metrics, let's remind the signification of different abbreviations:

TP	*True Positive*	Faulty fight considered as faulty
TN	*True Negative*	Proper fight considered as proper
FP	*False Positive*	Proper fight considered as faulty
FN	*False Negative*	Faulty fight considered as proper

Table 2.4: Metric abbreviations

With the aim of selecting the best classification model, the following main classification metrics are used to assess predictions. They are briefly introduced to explain their importance and suitability in the model evaluation.

- ***Precision***:

Precision metric tells us how many predicted samples are relevant in the data predicted as positive. It is a measure of a classifier exactness.

Precision : TP / Class 1 predicted

	Class 0	Class 1
Class 0	TN	FP
Class 1	FN	TP

Recall : TP / Class 1 true states

$$Precision = \frac{TP}{TP + FP}$$

- ***Recall***:

Recall metric shows how many relevant samples are selected. It is a measure of the classifier completeness.

$$Recall = \frac{TP}{TP + FN}$$

- ***F1 Score***:

F1 metric is the weighted average of the precision and recall.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

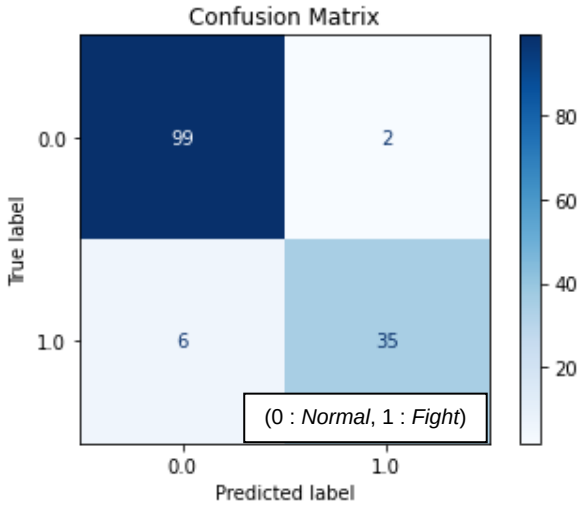


Figure 4.33: Confusion matrix for SVM classifier

	Precision	Recall	F1-score
Normal	0.94	0.98	0.96
Fight	0.95	0.85	0.90
Accuracy			0.94
Macro average	0.94	0.92	0.93
Weighted average	0.94	0.94	0.94
Precision	94.37%		

Figure 4.34: Classification report for SVM

The precision value at the end of the final epoch is **94,3% (***Figure 4.34*). The score doesn't reach 100% due to a few people considered as fighting whereas they are not (2 *False Positive*), and normal whereas they are fighting (6 *False Negative*). *False Negative* are more a concern than *False Positive*, since detecting a false fight isn't too damaging, but not being able to detect a fight and assimilating it as a normal behaviour can be dangerous for people in the airport.

- **Performance**

The results obtained *Figure 4.35* using the SVM classifier show that the model is adequate for a fight classification. Indeed, four fighting actions at stake (punching, kicking, shooting and pushing) are assimilated as a fight. At the oppositive standard actions such as walking, shaking hand, are correctly classified as a normal behaviour on both inferences (images and videos). When bystanders are present on the flow, the model can make the difference between people fighting and those standing out of the fight, even when fights are confusing with many protagonists concerned. It is not biased and works for every kind of people in the same way.

The model makes the prediction frame by frame, this is why the rate frame selected for best performance was around 30 FPS in order to get as smoother results as possible. But compared to the VGG16 model, SVM still provides predictions quicker and enables to get a quite smooth video sequence for other FPS values.

Furthermore, the model still classifies correctly when the camera angle is changed since the keypoints are normalized before any classification. Thus, the camera can be placed in a corner to get a high view or at human size. Low light luminosity conditions are not a problem for this model. Thus, it can be used in large set of environments.

Figure 4.35: Frames extraction from a video using SVM

- **Limits**

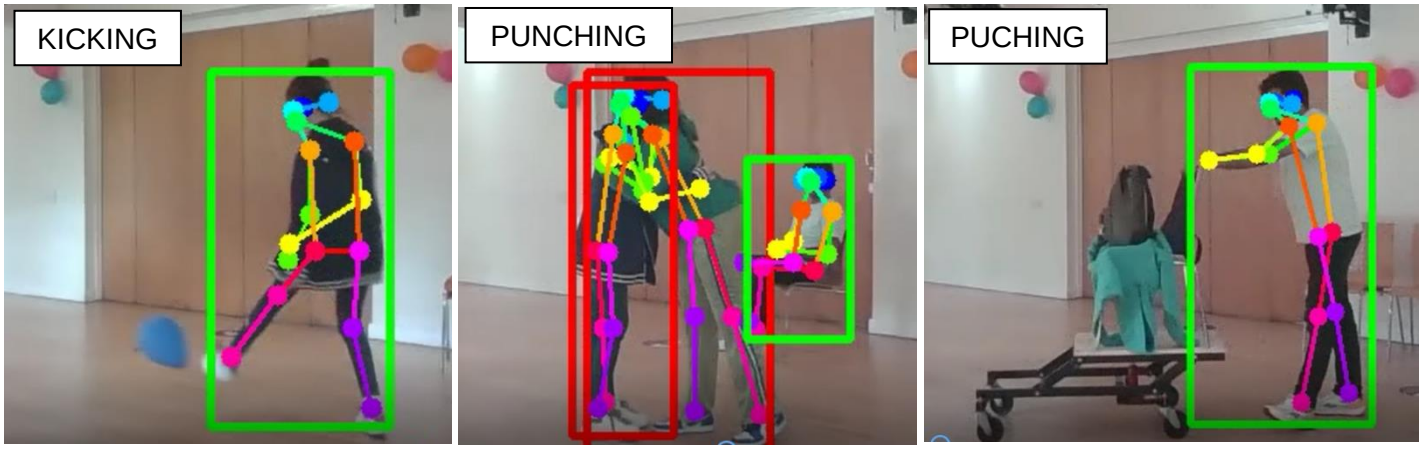


Figure 4.36: Confusing scenarios for SVM

Some ambiguous behaviours have been tested on *Figure 4.36* in order to fully verify the model's performance. Firstly, kicking a ball as a child could do within an airport waiting for the plane, or pushing a trolley with suitcases have correctly been classified as a normal behaviour. However, an ambiguity has been detected when people are giving a hug to each other's. This error could be corrected adding this action in the normal side of the dataset.

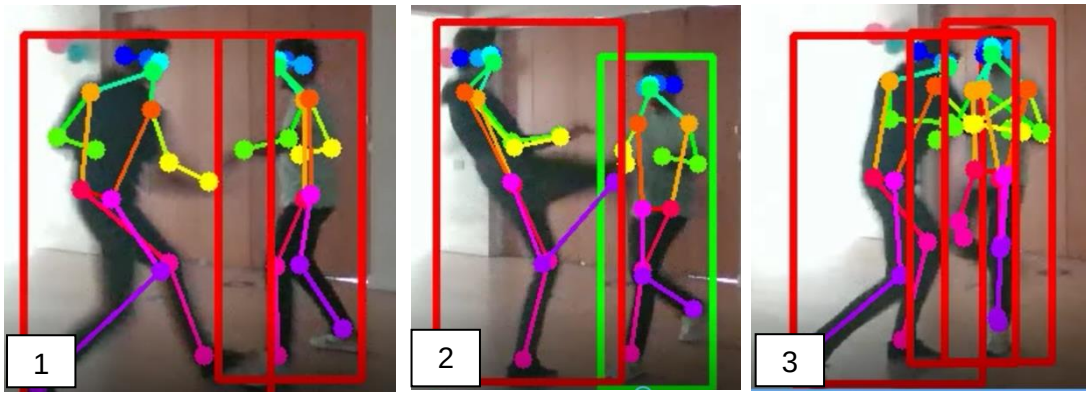


Figure 4.37: Smoothness limits for SVM

Secondly, it has been observed that when a person is fighting, the predictions can sometimes swing between a normal behaviour and a fight (alternating red and green on the frames continuously). Indeed, when someone is fighting, he can take a normal pose quickly during the fight, changing the way he fights for instance (from kicking to punching). On *Figure 4.37*, it is visible that the person at right on the second frame is changing his posture to go to frame 1 to frame 3. This change is classified as a normal behaviour even if he is still taking part into the fight. This problem is because frames are considered one by one and not as image's packages. To solve this problem, another model such as LSTM would have been more suitable.

More, because the segmentation part is ahead the model, its limits are still visible in matter of seeing the full body in the frames to get good predictions.

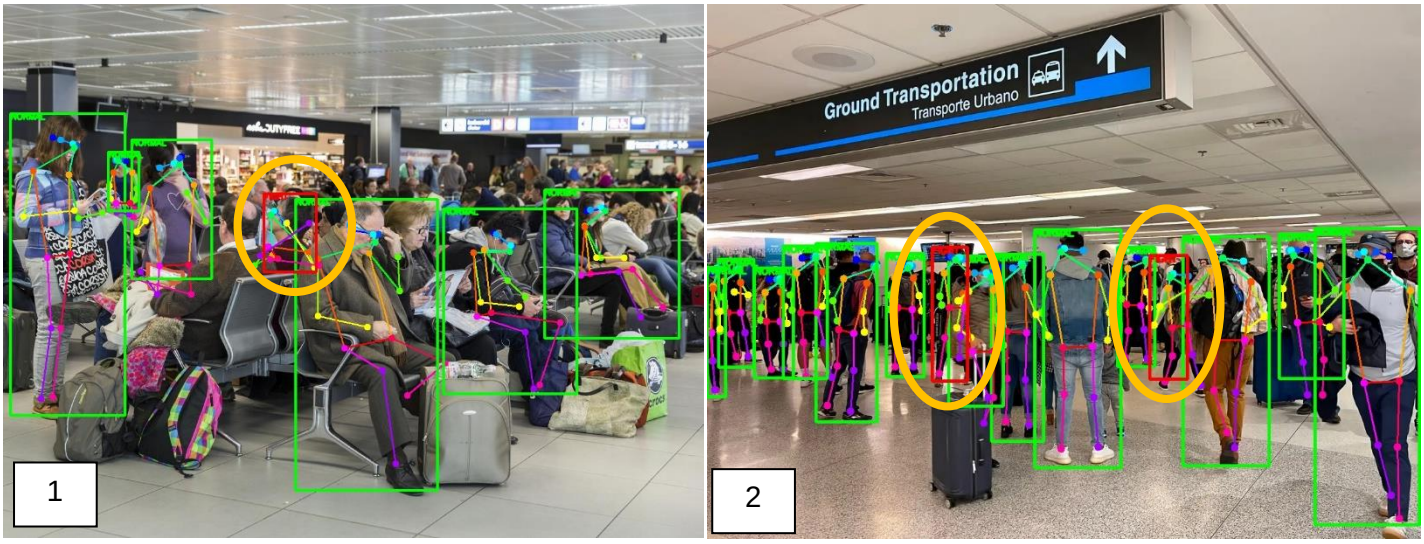


Figure 4.38: SVM's limits in an airport

Analysing SVM's limits in an airport, it is visible that on *Figure 4.38* - 1 that some poses when they are not entirely distinguished by the model can be confusing. Here, a man sitting is classified as fighting whereas is just sitting face back to the camera. On *Figure 4.38* - 2, many people are present on the stream and even if occlusions are well distinguished by *HRNet*, it can sometimes conduct to wrong classifications by SVM. This is why to have fully performant results it would be interesting to place the camera in a top corner somewhere in the airport, to have a high view of people.

Neural Network

We use a Multilayer Perceptron (MLP) to classify the poses estimated on the image. MLPs are known to be good with regression, and more specifically with classification. This model would serve the exact same purpose as the SVM. The goal is to try different approaches to obtain the best performance.

- **Model Hyperparameters**

The retained characteristics, after fine tuning it on our dataset, are as follows:

Architecture

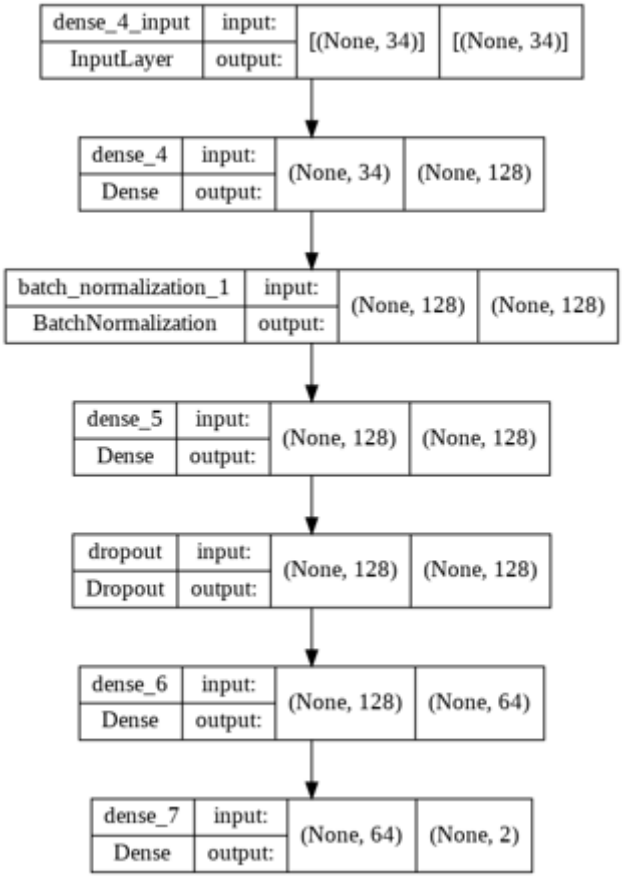


Figure 4.39: MLP architecture

Depth: as the input data are only 34 features vectors, we choose to have a quite shallow network, to avoid vanishing gradient issue.

Layer choices: we use a batch normalisation layer between 2 dense layers and a dropout layer to counteract overfitting on the dataset.

Optimizer	RMS prop
Learning rate	1e-3
Loss function	Sparse Categorical cross entropy
Epochs	20
Batch size	30
Activation function	Sigmoid

Figure 4.40: Model hyperparameters

Those parameters were chosen as they gave the best results.

- **Performance on training dataset**

After dividing the dataset with an 80-10-10 repartition (80% training data, 10% test data and 10% validation data), we train our model.

Figure 4.41: Training, testing accuracy and loss function curves during the model training

Those curves demonstrate a bit of overfitting, but overall, we manage to converge with our model.

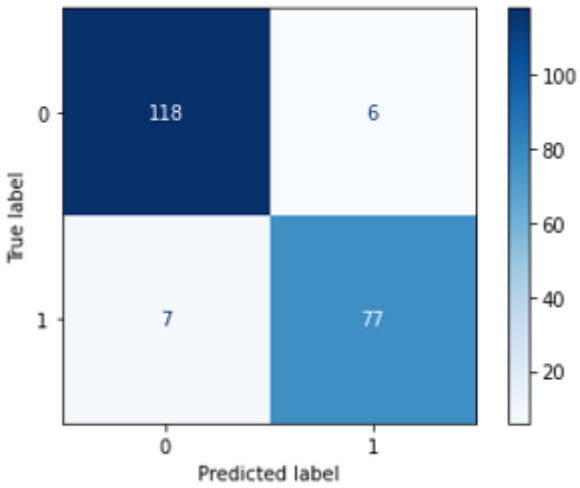


Figure 4.42 : Confusion matrix on the validation dataset for MLP

	Precision	Recall	F1-score	support
Normal (0)	0.97	0.94	0.95	124
Fight (1)	0.91	0.95	0.93	84
Accuracy			0.94	208
Macroavg	0.94	0.94	0.94	208

Figure 4.43: Classification report on the validation dataset for MLP

We obtain good results, over 90% in every metrics. The 0.95 in recall for the fight class is indicative of our model ability to theoretically catch most of the fight occurrences. We use our validation videos to assess performance in real conditions.

Figure 4.44: Classification example for MLP

- **Limits**

When used on more challenging images and videos (like our custom test videos or the SDAH dataset), the classification results are a bit more mitigated. This model presents most of the weaknesses the SVM suffers from. The models can prove itself to be too sensitive on some videos, easily classifying ambiguous and not-so-ambiguous poses as fighting poses. It can be an issue, as we can tolerate a bit of imprecision to achieve optimal recall, but it needs to retain a certain precision to be an affective alarm. We also need to remember that we are doing predictions on a pose estimation model that has its own imprecisions when exposed to hidden body parts. We need to consider the final precision of our model as a product of the precision scores from all preceding models (object detection model and pose estimation model).

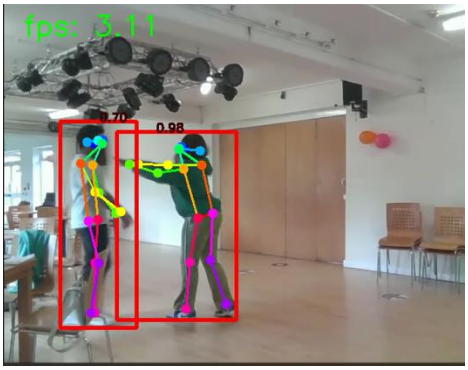


Figure 4.45: Misclassification example for MLP

Figure 4.46: Airport example of the misclassification

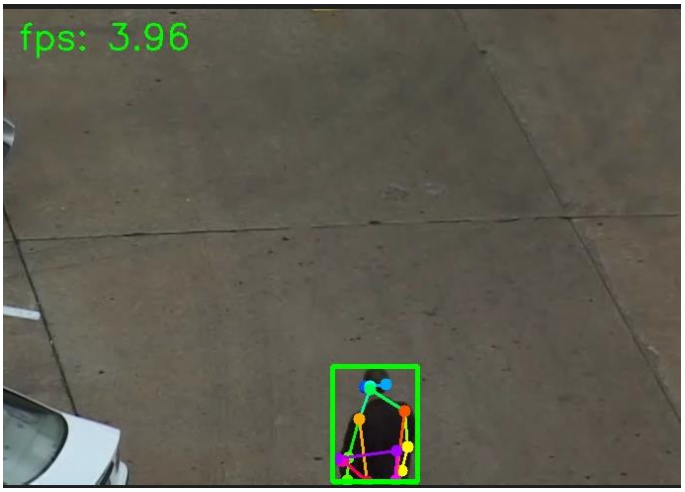


Figure 4.47: Example of the random plotting (body partially visible)

4.3.4 Conclusion regarding the final model

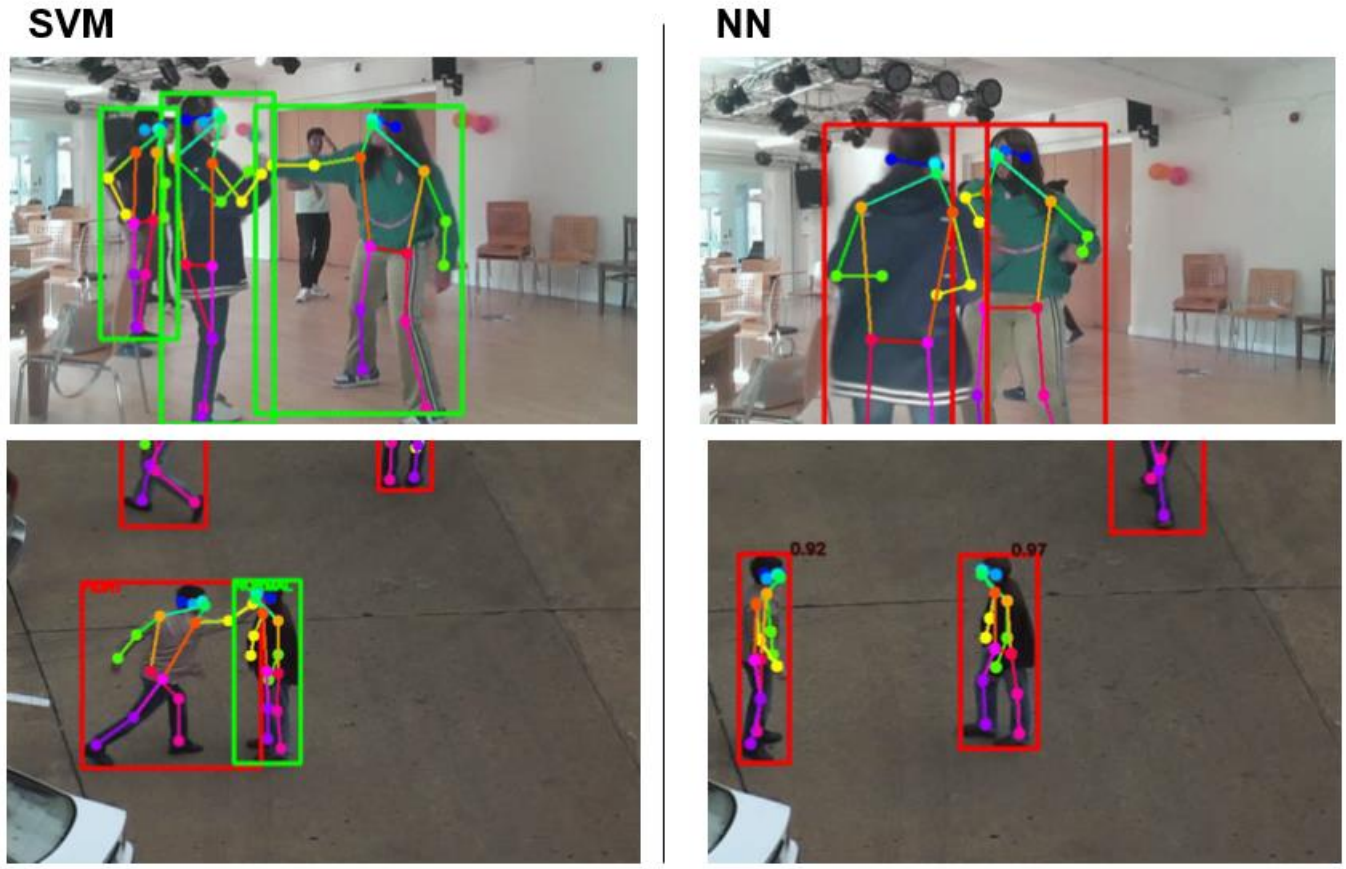


Figure 4.48: Models comparison

For the final system, the keypoints approach is kept instead of the features approach. Indeed, the VGG16 model is not smooth enough to be used in real-time scenario. If another neural network has been processed, the approach could have revealed itself more relevant, but within the project's amount of time given, the keypoints approach prevails in the given case.

Comparing both SVM and Neural Network models for the keypoints approach on *Figure 4.48*, it is seen that they get similar scores. However, the Neural Network (NN) considers a fight without interruption compared to SVM which switch between normal and fighting behaviour during a fight considering the fighting poses transitions as normal, because of the frame-by-frame prediction (first video on *Figure 4.48*). However, in the second video, the NN classifies normal poses as a fight when people are just standing. A mistake that SVM does not make. Indeed, it seems like the NN has overfitted to the dataset, and cannot take into considerations the full diversity of poses that can happen in various scenarios. The SVM model is most performant when the height of the camera view is high (as seen on the second video in the bottom row on *Figure 4.48*) which would be more realistic in a airport setting, thus this model has been chosen as a final model for the project. Though, the NN could have been improved without the time constraint.

4.4 Hardware selection

4.4.1 System design

The system functions are primarily comprised of carrying out video acquisition, monitoring and terminal model processing modules; the structure block diagram is shown on *Figure 4.49*. The NVIDIA Jetson Nano, HDMI interface display and CSI-Camera were used throughout the development process. The Jetson Nano provides a mobile GPU in the embedded space and offers application programming interfaces for AI and computer vision. Firstly, for the input video stream, a decoder such as H.264 is used in combination with hardware for decoding, and then the model is input. The model part is loaded with the help of DeepStream's GST-Nvinfer interface to load the model. For the test results, they will be presented on a display.
In the detection scenario, the project is based on the acquisition of the behavioural category, the acquisition of the area in which the pedestrian is located and the determination of the behavioural category, and the generation of the corresponding detection meta information, which is converted to send the detection results.
In summary, using SVM-based keypoints acquisition and *HRNet*, VGG-16's detection-related methods, with the help of Nano GPU hardware and software frameworks such as DeepStream, real-time detection of dangerous behaviour at the edge can be achieved, meeting the real-time requirements of intelligent monitoring and analysis scenarios, and realising real-time alerts in the event.

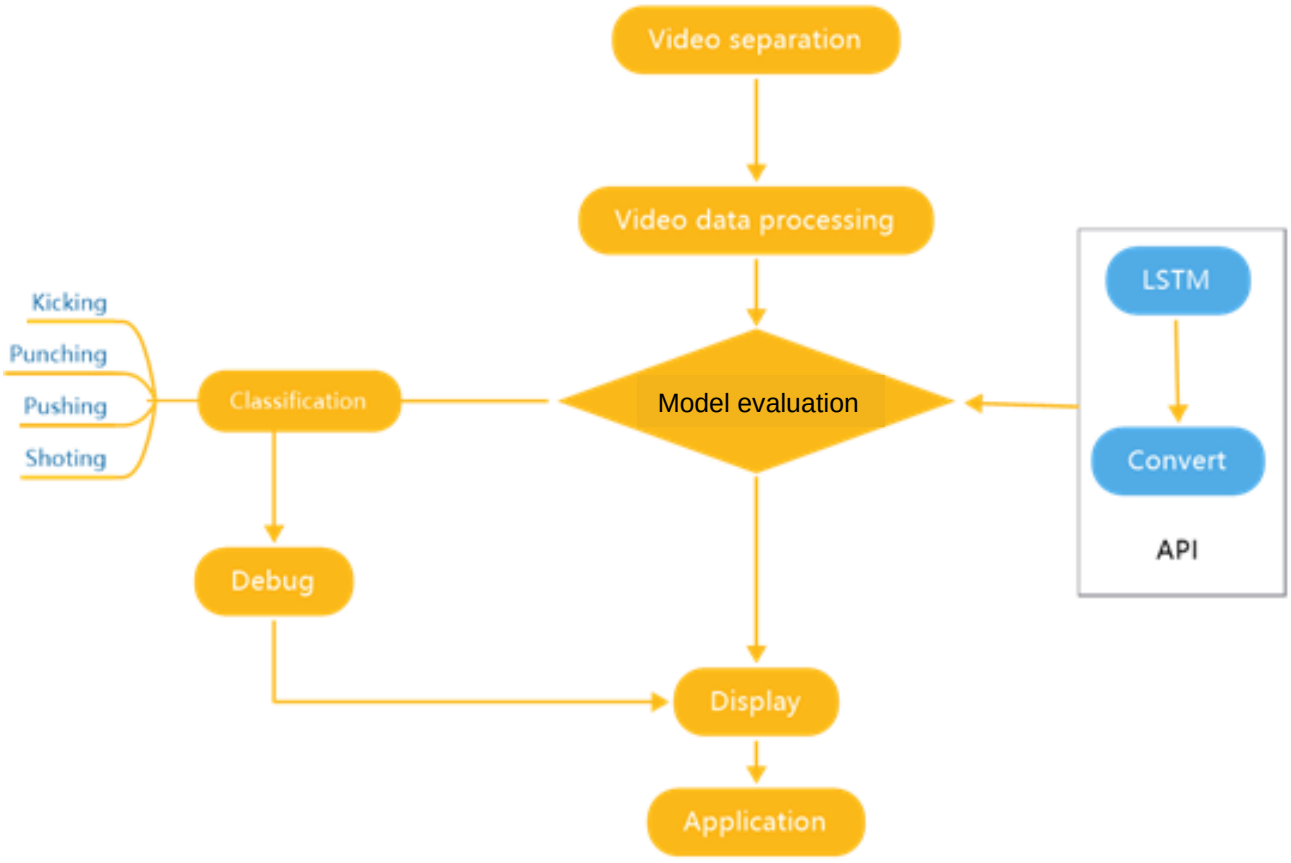


Figure 4.49: System architecture for the hardware

4.4.2 Hardware development platforms

Jetson Nano

As an embedded GPU development board, the Jetson Nano has been chosen because it includes 4GB of memory for encoding and decoding 4Kp30 video. The Jetson Nano used in this report operates on Ubuntu 18.04 with Arrch64 architecture and comes with python3, CV2 and CUDA10.2 and cuDNN software for embedded design integration into production systems. This camera model has a similar viewing angle as the CCTV shown.

Processor	
CPU	64-bit Quad-core ARM A57 @ 1.43GHz
GPU	128-core NVIDIA Maxwell @ 921MHz
Memory	4GB 64-bit LPDDR4 @ 1600MHz
*Video Encoder**	4Kp30
*Video Decoder**	4Kp60

Table 4.3: Jetson Nano features

CSI-Camera

The system requires a device to see its environment, so a CSI-camera was obtained to carry out this function. The CSI-camera is used for the front-end video capture, which is suitable for monitoring occasions with more points and meets the design requirements, being high-resolution, high framerate and high angle of view.

Camera		
	Sensor	IMX219
	Resolution	3280 × 2464
	Lens specifications	CMOS size: 1/4inch

Table 4.4: CSI-Camera features

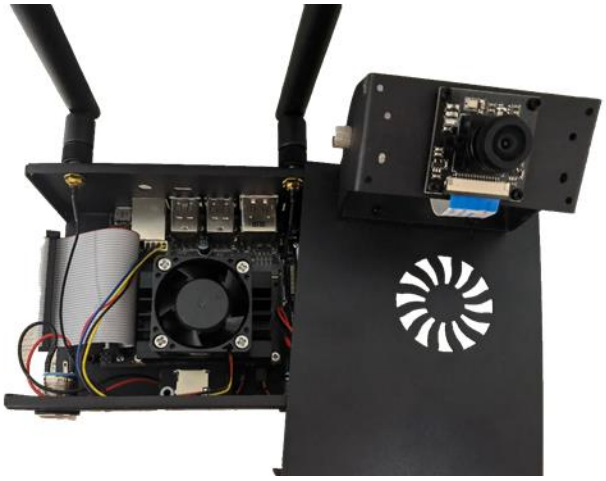


Figure 4.50: Assembled Jetson Nano

4.4.3 Debug

System commissioning is the testing of the work of the individual sub-modules. Checking and testing the individual hardware prior to application facilitates the commissioning and testing of the entire system.
Before connecting the whole system, an initial check is carried out on the individual electronic components and the connections between them are in good condition. Subsequently, the system is powered up and tested.

1. After powering up the system, the first step is to check whether the power supply is normal by looking at the indicators on the development board, the display, and the camera.
2. Once the system has started up, determine if the cooling fan is working by entering a code into the terminal.
3. Debug the CSI-Camera and turn on the camera via the terminal to check if the video path is working. At the same time, check if the input data conforms to the standard by checking the camera parameters.
4. Check the model operation by checking the system CPU and GPU calls via the command ***jtop***.

4.4.4 Hardware functionality test

Initial tests with the development board and display worked well however the CSI-Camera indicator was not reflective. A separate test of the CSI-Camera identified the camera as broken and was later replaced with a new one. After solving problems with the camera that arose during the commissioning process, a functional test of the entire system is carried out. The system can be switched on at the monitor for real-time monitoring after power-up.

Figure 4.51: Monitoring effect

In real applications, the majority of CCTV frames are uploaded to the terminal using TPC/IP, where they are unified for real-time detection. However, the conflict detection system is overloaded with complex code which places high demands on the hardware processing. The next section will detail much of the model integration with the hardware, highlighting many of the challenges that come with it.

5. Integration

5.1 Model building and training

The integration began with the following achieved objectives in order to be able to use the segmentation model and the classifier on the hardware:

- ✓ Connecting the Jetson Nano suitable with Jetpack 4.6 and CUDA 10.2
- ✓ Installing the required Python 3.6's libraries (*OpenCV 4.1.1, Pytorch, Cython, Torchvision, Pillow*…)
- ✓ Launch the video launch from the Nano's camera (command + launch from a python file)
- ✓ Enabling GPU functionality on the Jetson Nano

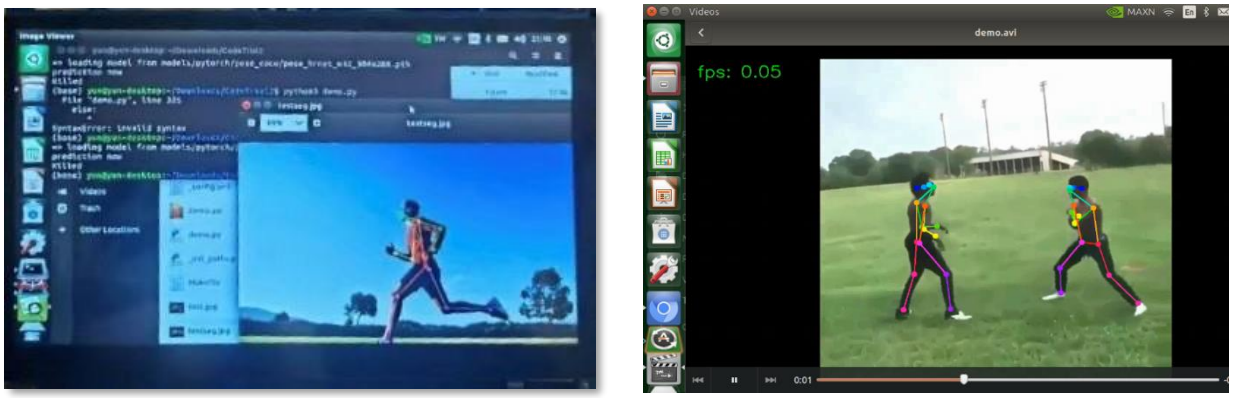


Figure 5.1: Image predictions on the hardware

The image definition was promising regarding the use of the Nano for both segmentation and prediction in real time.

5.2 Hardware system burn-in

The design system was finally tested on the Jetson Nano. The system used for the evaluation was Ubuntu 18.04 with Arrch64 architecture.

Figure 5.2: Flashing system OS

5.3 System debugging

Swap file creation

During the debugging process, the system was killed several times when the model was in use, and after checking with jtop, it was found that the GPU memory was insufficient. After adding swapfile, the memory was increased to 10.1Gb and the system could be applied normally. The following diagram is displayed on *Figure 5.3*.

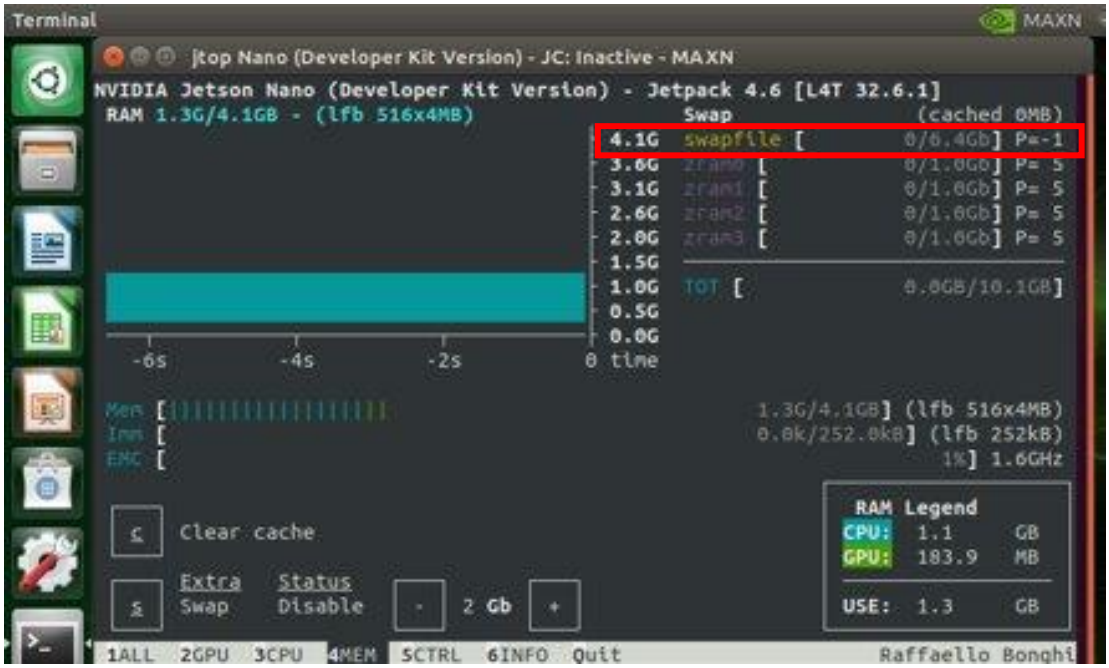


Figure 5.3: GPU monitoring

However, the swap file wasn't enough to solve the lack of memory's issue. It has just been succeeded to launch the segmentation on a pre-recorded video, but not in real-time.

Model reconsideration

In particular, the most involved part in term of process time was the faster RCNN's predictions from the segmentation (second arrow on *Figure 5.4*). A low image resolution has been set up with a lower frame rate and image size, but the problem still persisted.

```python
def main():

        # cudnn related setting
        cudnn.benchmark = cfg.CUDNN.BENCHMARK
        torch.backends.cudnn.deterministic = cfg.CUDNN.DETERMINISTIC
        torch.backends.cudnn.enabled = cfg.CUDNN.ENABLED

        args = parse_args()
        update_config(cfg, args)

        box_model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
        box_model.to(CTX)
        box_model.eval()

        pose_model = eval('models.'+cfg.MODEL.NAME+'.get_pose_net')(
            cfg, is_train=False
        )

        if cfg.TEST.MODEL_FILE:
            print('=> loading model from {}'.format(cfg.TEST.MODEL_FILE))
            pose_model.load_state_dict(torch.load(cfg.TEST.MODEL_FILE), strict=False)
        else:
            print('expected model defined in config at TEST.MODEL_FILE')

        pose_model = torch.nn.DataParallel(pose_model, device_ids=cfg.GPUS)
        pose_model.to(CTX)
        pose_model.eval()
```

Hardware's limitation WITHOUT the swap

```python
    # Loading an video or an image or webcam
    if args.webcam:
        vidcap = cv2.VideoCapture(0)
    elif args.video:
        vidcap = cv2.VideoCapture(args.video)
    elif args.image:
        image_bgr = cv2.imread(args.image)
    else:
        print('please use --video or --webcam or --image to define the input.')
        return

    if args.webcam or args.video:
        if args.write:
            save_path = '/Users/clair/Desktop/scenarios/segmentation.avi'
            fourcc = cv2.VideoWriter_fourcc(*'XVID')
            out = cv2.VideoWriter(save_path,fourcc, 30.0, (int(vidcap.get(3)),int(vidcap.get(4)))) #24
        while True:
            ret, image_bgr = vidcap.read()
            if ret:
                last_time = time.time()
                image = image_bgr[:, :, [2, 1, 0]]

                input = []
                img = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
                img_tensor = torch.from_numpy(img/255.).permute(2,0,1).float().to(CTX)
                input.append(img_tensor)

                # object detection box
                pred_boxes = get_person_detection_boxes(box_model, input, threshold=0.9)

            # pose estimation
            if len(pred_boxes) >= 1:
                for box in pred_boxes:

                    # Draw the bounding boxes on the image
                    #cv2.rectangle(image_bgr, (int(box[0][0]),int(box[0][1])), (int(box[1][0]),int(box[1][1])) ,

                    center, scale = box_to_center_scale(box, cfg.MODEL.IMAGE_SIZE[0], cfg.MODEL.IMAGE_SIZE[1])
                    image_pose = image.copy() if cfg.DATASET.COLOR_RGB else image_bgr.copy()
                    pose_preds = get_pose_estimation_prediction(pose_model, image_pose, center, scale)
                    if len(pose_preds)>=1:
                        for kpt in pose_preds:
                            draw_pose(kpt,image_bgr) # draw the poses

            if args.showFps:
                fps = 1/(time.time()-last_time)
                img = cv2.putText(image_bgr, 'fps: '+ "%.2f"%(fps), (25, 40), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0,

            if args.write:
                out.write(image_bgr)

            cv2.imshow('demo',image_bgr)
            if cv2.waitKey(1) & 0XFF==ord('q'):
                break
        else:
            print('cannot load the video.')
            break

    cv2.destroyAllWindows()
    vidcap.release()
    if args.write:
        print('video has been saved as {}'.format(save_path))
        out.release()
```

Hardware's limitation WITH the swap and decreasing the image's resolution

Figure 5.4: Hardware limitations

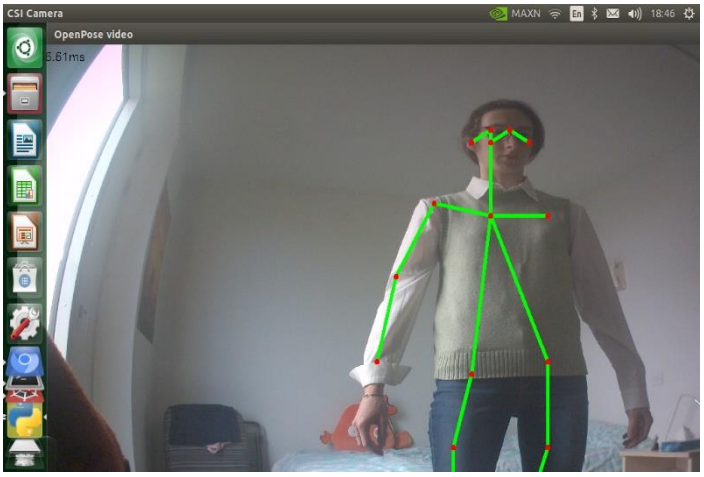


Figure 5.5: OpenPose example on the hardware

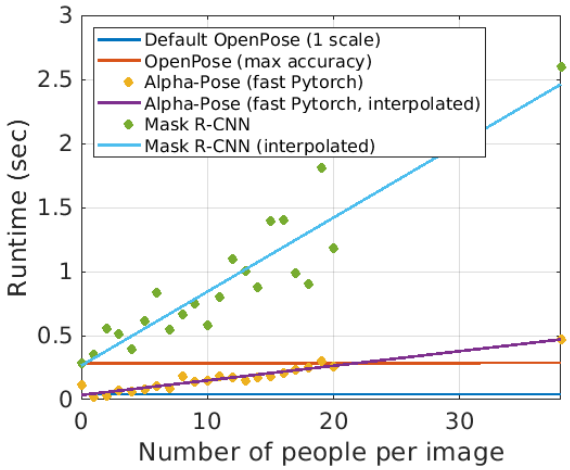


Figure 5.6: Segmentation models comparison

The hypothesis that the segmentation model was too important to be used on the hardware has been verified, when the launch of a lighter model (*Openpose*) on the hardware was a success (*Figure 5.5*).

Thus, another segmentation model would have been more suitable to use regarding the runtime (*Figure 5.6*). Though, it was difficult to plan the performance limitation of the model on the hardware before testing it. It could have been decided to entirely switch the segmentation model from *HRNet* to *Openpose*. But, after evaluating the balance between both on *Table 5.1,* model accuracy has been prevailed over the hardware perspective. Thus, HRNet has been kept being used on a computer even after encountering the hardware issues.

Model	***HRNet***	***Openpose***
Keypoints	Plot the keypoints of invisible body parts	Plot the visible keypoints
Accuracy	Very accurate	Semi-accurate
Functionality on hardware	Not working (model too big)	Working

Table 5.1: HRNet VS OpenPose

Torch2Trt

After that, *Torch2TrT* package has been tried in order to help the system to launch the model optimising it. This converter works by appending a conversion function (e.g. convert_*ReLU*) to the original *PyTorch* function call (e.g. *torch.nn.ReLU.forward*). Sample input data is passed through the network, just as before, except that now whenever *torch.nn.ReLU.forward* encounters a registered function (), the corresponding converter (convert_*ReLU*) is also called afterwards. The converter is passed the arguments and return statements of the original *PyTorch* function, as well as the *TensorRT* network being built. The input tensor of the original *PyTorch* function is modified to have an attribute _trt, which is the *TensorRT* counterpart of the *PyTorch* tensor. The conversion function uses it _trt to add layers to the *TensorRT* network and then sets the properties of the output tensor associated with _trt. Once the model is fully executed, the final tensor return is labelled as the output of the *TensorRT* network and an optimised *TensorRT* engine is constructed.

```
import torch
from torch2trt import torch2trt
from torchvision.models.resnet import alexnet

# create some regular pytorch model...
model = resnet(pretrained=True).eval().cuda()

# create some example data
x = torch.ones((1, 3, 224, 224)).cuda()

# convert to TensorRT feeding sample data as input
model_trt = torch2trt(model, [x])
```

Figure 5.7: Torch2trt implementation

Unfortunately, for reasons of time, this solution couldn't be processed until the resolution.

Due to these Jetson Nano memory limitations, it has been decided to realise the final demonstration and the testing phase on a computer with GPU. Indeed, even if the Jetson Nano could support our model, it would have been too slow to use it properly in real-time. In order to do that, four NVIDIA P100 GPU cards would have been required to have the model fully performant on the hardware.

6. Testing phase

6.1 Aim

The testing process aims to check whether the system is compliant against the user requirements. Various tests are carried out in a suitable environment where the testing process will reveal any defects in the model. The testing phase will also reveal any other possible risks that have not been previously identified. As seen on *Figure 6.1*, the expected result and the observed result are compared, and the accuracy of the requirements is checked.

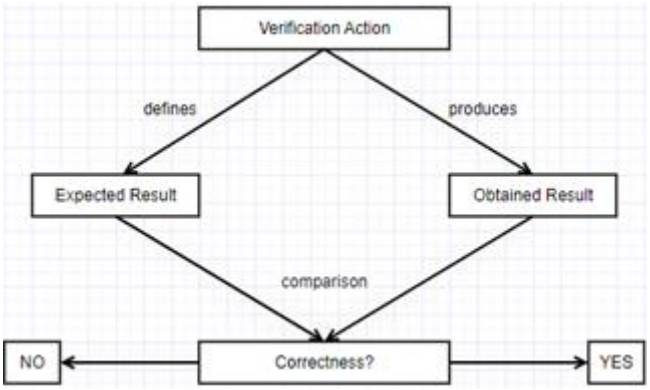


Figure 6.1: Verification sequence

6.2 Methodology

The tests were done in CSA using a computer with GPU. We need to carefully consider the location of the camera within CSA to represent where typically CCTV cameras would sit in an airport. 12 different scenarios were prepared for the test processes, and about 8 volunteers were needed for testing.

We started the testing phase with basic movements to test whether the system correctly detects actions such as punching, pushing, kicking, and shooting. In the first three scenarios, two people come face to face, and punch, push, kick and shoot actions are applied respectively. Afterward, we included more people in the testing process and tested how accurately the system worked in crowded environments and at what stage it started to lose accuracy. At this stage, we tried to create a real-life simulation with people exhibiting normal behaviour as well as people in conflict. We also repeated baseline conflicts at 30 FPS to test the performance of the hardware we used, and we have observed whether they have been correctly detected. Considering that the cameras will be in different locations within the airport, we changed the camera angle. Distance tests were carried out, to see the extent at which conflict can be detected. We also tested the system's operation, considering possible adverse environmental conditions such as low light. Finally, we realised possible confusing scenarios such as passengers pushing luggage, children playing games (kicking a balloon), and hugging, and observed whether the system made the correct predictions in these situations.

During these tests, different requirements are checked, such as whether the system detects the conflict correctly (like punching, pushing, kicking), whether the system works correctly in crowded environments, whether it is affected by adverse external conditions (low light), whether the system correctly detects the conflict in near and far from the camera, and whether the camera is used at an appropriate angle. The test scenarios performed are shown on *Figure 6.2*.

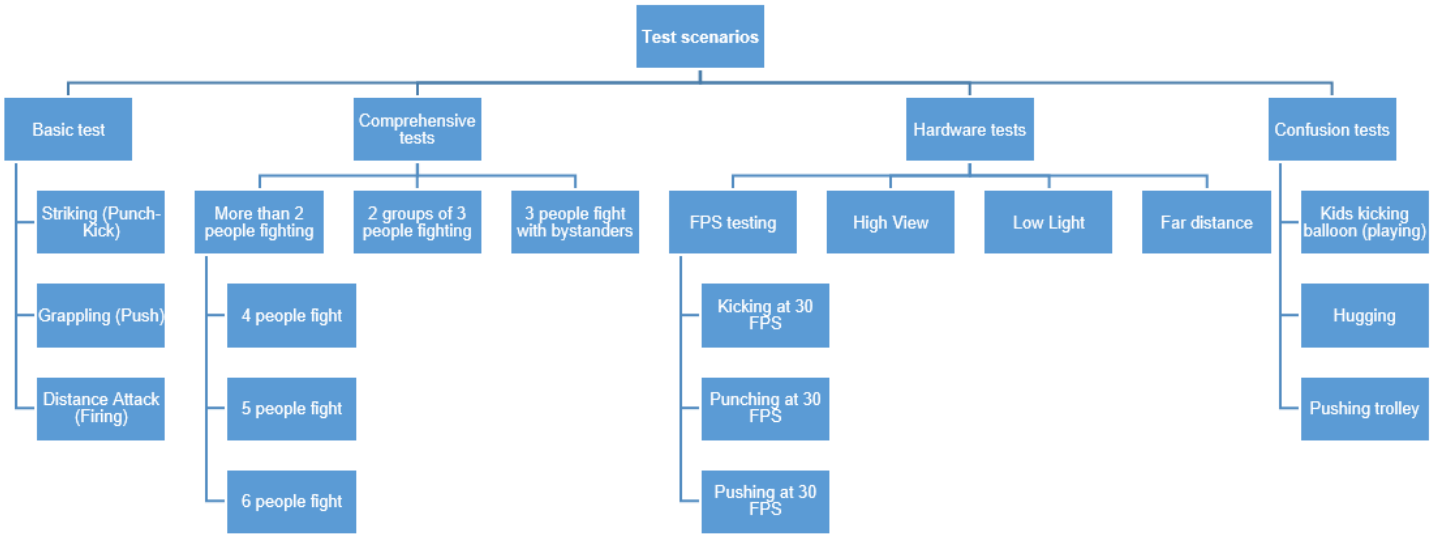


Figure 6.2: Testing Scenarios

6.3 System compliance

Quality assurance is a process that is applied to determine whether the system meets user requirements. It aims to determine and maintain the requirements for the reliability of the developed system.

There are four different methods for verifying requirements. These are analysis, inspection, demonstration, and testing. Inspection is performed by five senses or physical measurements of a system. The demonstration aims to obtain and verify the results as planned. The test is to consider the outputs to be obtained considering the requirements and to verify the system with pre-planned data. And finally, analysis is the verification of the system using models and calculations.

The necessary controls are carried out during the project process and the necessary quality is ensured by ensuring early detection of errors during the project development phase. With the tests carried out, it is checked whether the system works by the specified criteria. As Table 6-1 shows, most of the requirements have been verified. In this project, system reliability and quality were ensured by verifying requirements such as conflict detection accuracy, correct detection according to camera angle and distance of the camera and being unaffected by adverse external factors. Only a few requirements have not been met. For example, the need for the system to operate continuously 24 hours a day could not be met. This is because the memory of the hardware used (Nvidia Jetson nano) runs out in a short time, and the system shuts down.

		Requirements Priority	Requirements	Verification Method (analysis, inspection, demonstration, test)	Verification Success Criteria	Final Verified Value	Compliant?
	User Requirements	1	**The System shall detect for conflicts within the airport environment**	Test	Detect conflicts(pushing, punching, kicking, shooting)	-	✓
		1	**Operators shall be notified within 5 seconds when conflict is detected and present the confidence level**	Demonstration	Warning within 5 seconds	-	X
		2	**Considering Article 4 GDPR Definitions and Article 9 GDPR Processing of special categories of personal data, the protection of data such as physical, physiological, location, cultural, and social identity of individuals shall be ensured**	Demonstration	GDPR compliance	-	✓
		2	**System source-code shall be updated remotely**	Demonstration	Remote update via SSH and VNC	-	X
System	Operational Requirements	2	**System accuracy shall be at least 90%**	Analysis	90% accurate detection	94.37%	✓
		1	**The system shall detect for conflict concerning pushing, punching, kicking, and even firing**	Test	Detect conflicts (pushing, punching, kicking, shooting)	-	✓

	Functional Requirements	1	**The system shall detect the conflicts near and far from the camera.**	Test	Distance independent conflict detection	-	✓
		2	**The system shall be capable of detecting 10 people**	Test	Detect 10 people	10	✓
		2	**Depending on the hardware, the system shall record at least 20 minutes of video**	Analysis	Record at least 20 minutes of video	20 min	✓
		2	**The system shall be able to take at least one video feed**	Demonstration	Ability to work with video feed	-	✓
		3	**The camera to be used shall have a minimum viewing angle of 135 degrees**	Test	Minimum 135 degree viewing angle	170	✓
	Constructional/ Physical Requirements	1	**The system shall be protected against external attacks (flammable hardware, cutting the wire, data disconnection)**	Inspection	No damage to the operation of the system	-	✓
	Non-Functional Requirements	1	**Cost shall be at most £1000**	Analysis	Cost under £1000	£480.35	✓
		1	**The conflict detection system shall work continuously for 24 hours a day**	Test	24-hour operation of the system without freezing	-	X
		2	**The system shall be able to detect conflicts in low light conditions**	Inspection	Conflict detection in low light	-	✓

Table 6.1: System requirements verifications

1 ("must have")
3 ("less important")

7. Final thoughts

The final model is able to detect conflict limited to punches, kicks, pushes and firing however there are limitations with the model and more so with the selected hardware so that it can be used in an airport environment. The following points highlight the challenges in the model and provide suggestions on potential improvements that could be made.

The main problem regarding the SVM model for the keypoints approach was about the movement fluidity. Since the ambiguous behaviours can be managed from the dataset and that the results from videos captured at a greater height, the smoothness is the only limit that cannot be managed. It was expected that a frame-by-frame prediction would be effective for our system, however, there is no doubt that considering a whole movement with at least several frames for a prediction would have given better results, even if the SVM classifier is already very performant for such a task. With the model we have, it would have made more sense to include a middle class between a normal and a fighting behaviour when the model predicts a fighting behaviour at 50% accuracy. This can benefit the operator through a warning rather a full alert.

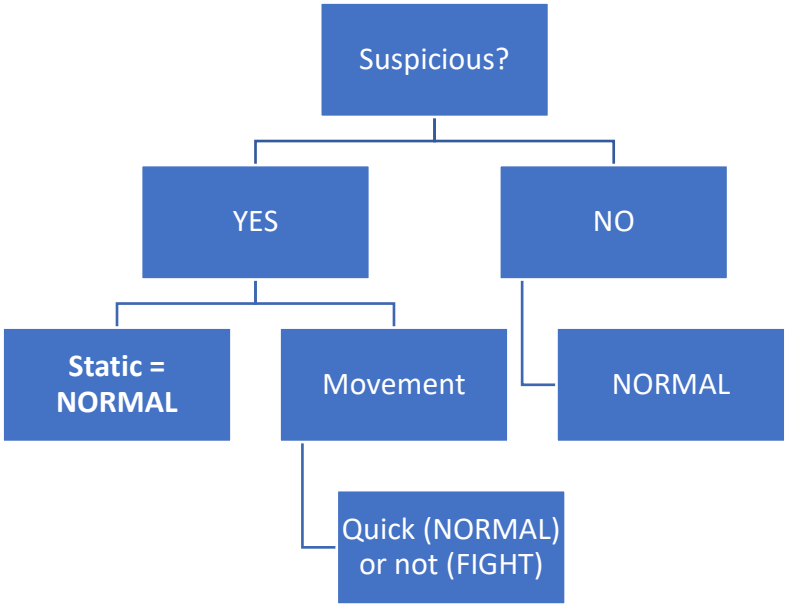


Figure 7.1: Model further development with LSTM

Another method would be to use a rule-based classification system based on the pose classification of a person over time. That solution would require a tracking system that has memory, keeping people past positions available for use. It would also require a rule to discriminate between an ambiguous non-fighting pose and a true fighting position.
The proposed metrics to differentiate would be the velocity of the movements made by the person in a suspicious pose, assuming that fighting movement would be faster than other ambiguous poses that happen not to be fighting.
That solution was explored, using the overlapping of bounding boxes as metrics to track individuals: if a predicted box would overlap with a predicted one the last frame, it is identified as the same person. If not, a new instance of person is created. Predictions which are non overlapping are considered out of the frame and thus eliminated.

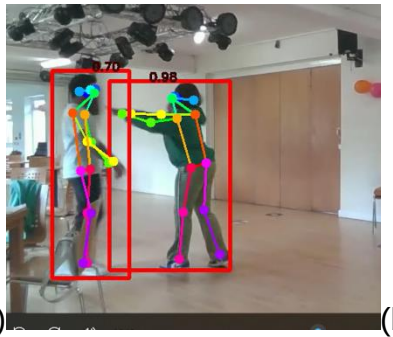
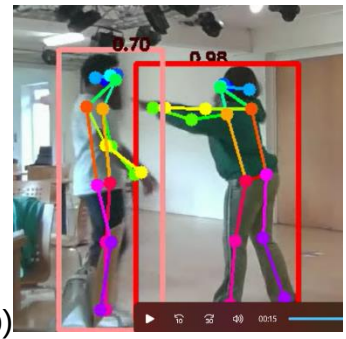
(a)  (b) 

Figure 7.2. Example how a rule-based algorithm (b) can eliminate an ambiguous behaviour that would be considered violent normally(a). It appears in pink.

To be usable, the tracking algorithm needs to be improved (as the overlapping criteria is far from enough, completely failing in a crowed place). Nonetheless, that could be something to explore to improve our model.

Security Hardware Challenges

The keypoints recognition approach used in our security systems is good for use in multi-person scenarios but is likely to cause misinterpretation of children's behaviour, making it impossible to detect real dangerous behaviour. Face recognition capabilities may solve this problem. Moreover, as an important transport node, airports could consider entering a database of criminals to increase security levels. Or another solution would be to use 3D pose estimation using multiple cameras so that the height of a person can be determined, this can help with problems of occlusion and better determining a child.

Technical Hardware Challenges

The system would have benefited from a dashboard, which is the interface between the operator and the system. This would provide the operator with what is detected and the threat level, perhaps even providing suggestion however this was not implemented, the same goes for the alarm system which would provide audible feedback to the operator. One of the technical problems with the Jetson Nano is that the GPU is not able to conduct video transmission and behaviour detection at the same time. In an actual airport setting, the GPU of the terminal will be under much higher computational load, which could be accelerated by using a model converter optimised for the Jetson Nano such as torch2trt, but these problems, including video transmission, could be better solved on a Peer 2 Peer (P2P) cloud platform such as Amazon Web Services. The Jetson Nano would act as a video streamer and a Human Machine Interface with the use of a display, providing a video stream to the cloud. Another solution would be to use a computer with a high-powered GPU such as the NVIDIA P100 which is an AI specific GPU, capable of running the complex conflict detection model. As a final point, airports are known to be large and crowded scenarios where we believe that the use of

a 3D depth of field capturer when capturing video would provide additional benefits to target recognition.

With the suggested improvements it would be possible for the conflict detection system to be used in an actual Airport setting in enhancing security surveillance however the system still requires much development as many of the requirements have not been met particularly on the hardware side.

8. Conclusion

This report details the development of a conflict detection system. A number of requirements were compiled so that the system can be suited to detect conflicts in an airport setting. With this, an ethical review was carried out which spilled out concerns and benefits around safety, security and bias/discrimination. Two approaches were taken with model development being the feature approach and keypoint approach. At the middle time of the project, it has been decided to keep the keypoints approach to enable better performance. Within this approach, two models have been developed, an SVM and a Neural Network. At the end, SVM was most suitable for the airport environment considered.

In the same time, a hardware solution was developed to launch the most performant model and get an entire system ready. However, it was realised a Jetson Nano would not be able to cope with running the conflict detection model. Therefore, it has been decided to launch the final model on a PC with GPU.

As it stands the development of the system has met its aim in detecting conflicts within crowds however this is without the specified hardware. Recommendations are put forward to provide alternatives to the hardware issues such as using cloud computation (P2P) or using high performance computing.

9. Content of figures

10. Content of Tables

11. References

Andriluka, M. (2014). MPII Human Pose Database. Human-pose.mpi-inf.mpg.de. Retrieved 10 April 2022, from http://human-pose.mpi-inf.mpg.de/#overview.

Ben Mabrouk, A., & Zagrouba, E. (2018). Abnormal behavior recognition for intelligent video surveillance systems: A review. In *Expert Systems with Applications* (Vol. 91, pp. 480–491). Elsevier Ltd. https://doi.org/10.1016/j.eswa.2017.09.029

Boeing: India Will Lead South Asia Aviation Growth; Fleet To Quadruple By 2040. (2022). Retrieved 5 April 2022, from https://simpleflying.com/india-aviation-growth-boeing/

Denman, S., Kleinschmidt, T., Ryan, D., Barnes, P., Sridharan, S., & Fookes, C. (2015). Automatic surveillance in transportation hubs: No longer just about catching the bad guy. *Expert Systems with Applications*, *42*(24), 9449–9467. https://doi.org/10.1016/J.ESWA.2015.08.001

"Examples of Functional Requirements (Plus Types and Benefits) | Indeed.com." https://www.indeed.com/career-advice/career-development/functional-requirements-examples (accessed Apr. 10, 2022).

FAA Proposes Fines Against 5 Passengers for Allegedly Interfering with Flight Attendants. (2021). Retrieved 5 April 2022, from https://www.faa.gov/newsroom/faa-proposes-fines-against-5-passengers-allegedly-interfering-flight-attendants?newsId=26140

"Functional vs. Nonfunctional Requirements (With Examples) | Indeed.com." https://www.indeed.com/career-advice/career-development/common-functional-and-non-functional-requirements (accessed Apr. 10, 2022).

Gonçalves, P. J. S., Lourenço, B., Santos, S., Barlogis, R., & Misson, A. (2020). Computer Vision Intelligent Approaches to Extract Human Pose and Its Activity from Image Sequences. *Electronics 2020, Vol. 9, Page 159*, *9*(1), 159. https://doi.org/10.3390/ELECTRONICS9010159

Gugale, R., Shendkar, A., Chamadia, A., Patra, S., & Ahir, D. (2020). Human Suspicious Activity Detection using Deep Learning. *International Research Journal of Engineering and Technology*. www.irjet.net

IATA revises long-term global air travel forecasts. (2022). Retrieved 5 April 2022, from https://japantoday.com/category/features/travel/iata-revises-long-term-global-air-travel-forecasts

Leslie, D. (2019). Understanding Artificial Intelligence Ethics and Safety: A Guide for the Responsible Design and Implementation of AI Systems in the Public Sector. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3403301

Li, T., Chang, H., Wang, M., Ni, B., Hong, R., & Yan, S. (2014). Crowded Scene Analysis: A Survey. *IEEE*, *25*(3), 367–386. https://doi.org/10.1109/TCSVT.2014.2358029

Menzel, D. S., & Hesterman, J. (2017). *Airport security threats and strategic options for mitigation*.

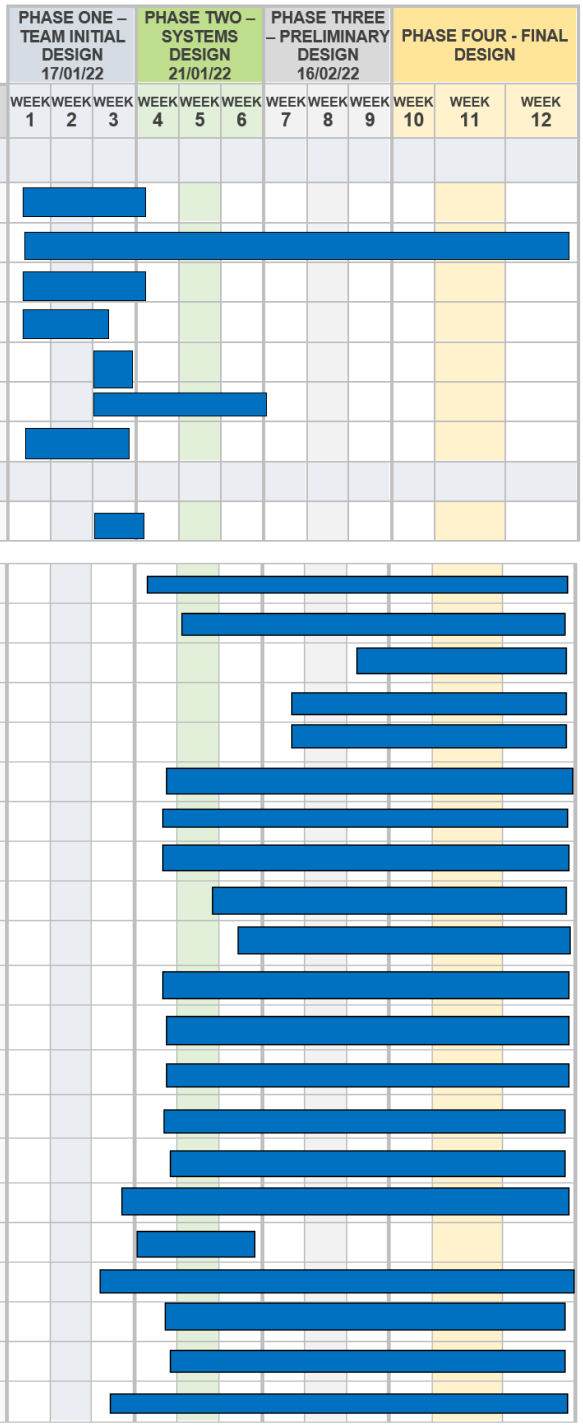
Pan, H., Yin, J., Ku, H., Liu, C., Feng, F., Zheng, J., & Luo, S. (2019). Fighting Detection Based on Pedestrian Pose Estimation. *Proceedings - 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2018*. https://doi.org/10.1109/CISP-BMEI.2018.8633057

Penmetsa, S., Minhuj, F., Singh, A., & Omkar, S. N. (2014). Autonomous UAV for Suspicious Action Detection using Pictorial Human Pose Estimation and

Classification. In *Electronic Letters on Computer Vision and Image Analysis* (Vol. 13, Issue 1).
Policy Brief: Airport networks and the sustainability of small airports - ACI World Store. (2019). Retrieved 5 April 2022, from https://store.aci.aero/product/policy-brief-airport-networks-and-the-sustainability-of-small-airports/
“Systems Engineering for ITS Handbook - Section 4 ITS Technical Processes.” https://ops.fhwa.dot.gov/publications/seitsguide/section4.htm (accessed Apr. 10, 2022).
“System Verification - SEBoK.” https://www.sebokwiki.org/wiki/System_Verification (accessed Apr. 10, 2022).
Ryoo, M., & Aggarwal, J. (2010). SDHA 2010 High-level Human Interaction Recognition Challenge. Cvrc.ece.utexas.edu. Retrieved 10 April 2022, from https://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html.
Understanding artificial intelligence ethics and safety. (2019). Retrieved 10 April 2022, from https://www.gov.uk/guidance/understanding-artificial-intelligence-ethics-and-safety
Unruly passenger incidents on the increase | Airlines. (2021). Retrieved 5 April 2022, from https://airlines.iata.org/analysis/unruly-passenger-incidents-on-the-increase
Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2017). Action Recognition in Video Sequences using Deep Bi-Directional LSTM with CNN Features. *IEEE Access*, *6*, 1155–1166. https://doi.org/10.1109/ACCESS.2017.2778011
“User Requirement Specifications (User Specs, URS) | Ofni Systems.” http://www.ofnisystems.com/services/validation/user-requirement-specifications/ (accessed Apr. 10, 2022).
Vallathan, G., John, A., Thirumalai, C., Mohan, S. K., Srivastava, G., & Lin, J. C. W. (2021). Suspicious activity detection using deep learning in secure assisted living IoT environments. *Journal of Supercomputing*, *77*(4), 3242–3260. https://doi.org/10.1007/S11227-020-03387-8/TABLES/4
Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., & Xiao, B. (2021). Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(10), 3349–3364. https://doi.org/10.1109/TPAMI.2020.2983686
“What are the four fundamental methods of requirement verification?” https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/1168/What-are-the-four-fundamental-methods-of-requirement-verification.aspx (accessed Apr. 10, 2022).
“What is Quality Assurance? - Definition from WhatIs.com.” https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance (accessed Apr. 10, 2022).
Zakhrchenko, I (2020) Basketball Pose-based Action Recognition.
Zhang, S., Liu, X., & Xiao, J. (2017). On geometric features for skeleton-based action recognition using multilayer LSTM networks. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, 148–157. https://doi.org/10.1109/WACV.2017.24

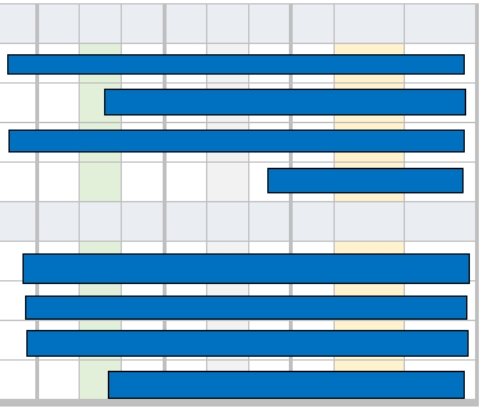
12. Appendix

12.1 Gantt chart



TASK ID	TASK	% DONE	ASSIGNED TO	START DATE	END DATE
1	Project Conception and Initiation – Conflict Detection				
1.1	Subject Description	100%	ALL	04/01/22	21/01/22
1.1.1	Context	100%	ALL	07/01/22	08/04/22
1.2	Business Need	100%	Karan	07/01/22	21/01/22
1.3	Existing System	100%	Marc-Olivier	11/01/22	14/01/22
1.4	Stakeholders	100%	Mehmet, Karan	11/01/22	21/01/22
1.5	Development Process	50%	Adeola	11/01/22	16/02/22
1.6	Proposed System	100%	Marc-Olivier	07/01/22	21/01/22
2	Project Definition and Planning				
2.1	Systems Design	50%	Mehmet	17/01/22	21/01/22
2.2	Procure Hardware	30%	Karan	21/01/22	08/04/22
2.21	Firmware, design circuitry	N/A	Ruolin	09/02/22	08/04/22
2.22	Combine model with hardware	N/A	Claire	16/03/22	08/04/22
2.23	Simulate circuitry	N/A	Roulin	21/02/22	08/04/22
2.24	Simulate entire system	N/A	Ruolin	21/02//22	08/04/22
2.3	Create requirements	100%	Mehmet	31/01//22	08/04/22
2.31	Create risks	100%	Adeola	31/01//22	08/04/22
2.4	Retrieve datasets	50%	Marc-Olivier	31/0/122	08/04/22
2.41	Balanced dataset	50%	Marc-Olivier	07/02/22	08/04/22
2.42	Predicting poses from skeleton on pictures (4)	50%	Claire	09/02/22	08/04/22
2.43	Assist in model development	50%	Marc-Olivier	31/01/22	08/04/22
2.5	Segmentation development on pictures and video launch (1)	100%	Claire	31/01/22	08/04/22
2.51	Launch the segmentation part on the dataset (2)	100%	Claire	31/01/22	08/04/22
2.52	Fine-tune model (LSTM or VGG16, transfer learning) (3)	50%	Claire	31/01/22	08/04/22
2.53	Apply the model on video stream (5)	50%	Claire	31/01/22	08/04/22
2.6	Systems Architecture	50%	Roulin	17/01/22	21/01/22
2.7	Budget	100%	Karan	21/01/22	16/02/22
2.8	Communication Plan	100%	Karan	14/01/22	11/04/22
2.9	Risk Management	100%	Adeola	31/01//22	11/04/22
2.10	Ethics	50%	Adeola	31/01/22	11/04/22
2.11	Research and Development	50%	Adeola	17/01/22	08/04/22

3	Project Launch and Execution					
3.1	Status and Tracking	50%	Karan	17/01/22	08/04/22	
3.2	Forecasts	N/A	Claire, ALL	16/02/22	11/04/22	
3.3	Project Updates	50%	ALL	17/01/22	11/04/22	
3.4	Chart Updates	N/A	ALL	16/03/22	11/04/22	
4	Project Performance / Monitoring					
4.1	Project Objectives	100%	ALL	17/01/22	08/04/22	
4.2	Quality Deliverables	100%	ALL	17/01/22	08/04/22	
4.3	Effort and Cost Tracking	100%	Karan	17/01/22	08/04/22	
4.4	Project Performance	N/A	ALL	16/02/22	11/04/22	



12.2 Supporting mathematical formulation

This proves that our normalization function is invariant to scale: the only thing that matters is the points positions relative to each other.

$$Let\ S_1 = \{A_0 \dots A_n\}\ and\ S_2 = \{B_0, \dots, B_n\}\ two\ sets\ of\ points\ (representing\ two\ keypoint\ skeletons)\ in\ a\ two-dimensional\ Euclidean\ space\ \{u_x = (1,0), u_y = (0,1)\}\ such\ as:$$

$$\forall k, j \in \{0, \dots, n\}, \exists \lambda \in \Re, A_j - A_k = \lambda(B_j - B_k)\ (a\ property\ indicating\ that\ the\ 2\ skeletons\ have\ the\ same\ shape: corresponding\ links\ are\ collinear).$$

$$We\ write\ the\ coordinates\ of\ a\ point\ as: \forall k \in \{0, \dots, n\}, A_k = (x_{ak}, y_{ak}), \forall k \in \{0, \dots, n\}, B_k = (x_{bk}, y_{bk})$$

$$Let\ T_{S1}: S_1 \to \Re^2\ be\ the\ normalizing\ function\ for\ S_1. \quad A \to T_{S1}(A)$$

$$Let\ T_{S2}: S_2 \to \Re^2\ be\ the\ normalizing\ function\ for\ S_2. \quad B \to T_{S2}(B)$$

$where\ T_{S1}(A) \cdot u_x = (x_a - x_{a0})/w_{S1}.$ and $T_{S1}(A) \cdot u_y = (y_a - y_{a0})/h_{S1}.$

$T_{S2}(B) \cdot u_x = (x_b - x_{b0})/w_{S2}.$ and $T_{S2}(B) \cdot u_y = (y_b - y_{b0})/h_{S2}.$

$with\ w_S = max_{a \in S}(a \cdot u_x) - min_{a \in S}(a \cdot u_x)$ and $h_S = max_{a \in S}(a \cdot u_y) - min_{a \in S}(a \cdot u_y)$

We will demonstrate that $\forall k \in \{0, \dots, n\}, T_{S1}(A_k) = T_{S2}(B_k).$

$$T_{S1}(A_k) \cdot u_x = (x_{ak} - x_{a0})/w_{S1} = (x_a - x_{a0})/(max_{a \in S}(a \cdot u_x) - min_{a \in S}(a \cdot u_x))$$

$$Let's\ say\ that\ max_{a \in S}(a \cdot u_x) = x_{am}\ and\ min_{a \in S}(a \cdot u_x) = x_{al}, m, l \in \{0, \dots, n\}\ and\ m \neq l, x_{am} \neq x_{al}$$

$$We\ are\ assured\ those\ values\ exist\ and\ are\ attained, as\ S_1\ is\ a\ finite\ set.$$

$$T_{S1}(A_k) \cdot u_x = (x_{ak} - x_{a0})/(x_{am} - x_{al}) = \lambda(x_{bk} - x_{b0})/\lambda(x_{bm} - x_{bl}) = (x_{bk} - x_{b0})/(x_{bm} - x_{bl})$$

We now need to prove that $max_{b \in S}(b \cdot u_x) = x_{bm}.$

As $max_{a \in S}(a \cdot u_x) = x_{am}$, $\forall k \in \{0, \dots, n\}, x_{am} \geq x_{ak}.$

$$x_{am} - x_{ak} \geq 0$$

$$\lambda(x_{bm} - x_{bk}) \geq 0$$

$$x_{bm} \geq x_{bk}$$

Proving that $max_{b \in S}(b \cdot u_x) = x_{bm}$

The reasoning is analogous with min. and with the y-coordinate.

With that, we can say that: $T_{S1}(A_k) = T_{S2}(B_k)$

This proves that our normalization function is invariant to scale: the only thing that matters is the points positions relative to each other.