Group 4 Final Report

Integrating AI-based Pose Detection and Fitness Assessment Model with SAAB's SAFE Subsystem for Enhanced Airport Security and Operational Efficiency

Authors: David Mihalcea
Abhinav Kumar Choudhary
Cheuk Lam Lam
Ciaran Gruffeille
Aaron Cherian
How Nam Boon
Wenyan Zhong

Course: Applied Artificial Intelligence
Date: 11/4/2023

Abstract

The maintenance workers play a very important role in airports, and will continue to play this role in smart airports, however, there is no current technology that is being used to automatically classify if they are fit to be working or not. This concerns comes from the aftermath of COVID-19, which saw thousands of maintenance staff being laid off, causing the ones that remained with a job to be over worked, which increases the possibility of human error that can lead to a disaster. To counter this, the group has designed a system architecture that can exploit surveillance video to trigger an alert if any unfit workers should be seen. To do so an LSTM was trained using a combination of differently acquired data sets that would be integrated with the SAAB SAFE dashboard. The accuracy was achieved with the 95% and 75% for training and validation.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Airports play a vital role in the global transportation ecosystem, connecting people, cultures, and economies across the world. As air travel continues to grow, airports are facing increased pressure to ensure safety, security, and operational efficiency in their day-to-day functions. With the rapid technological advancements, the complexity of airport operations is also increasing. It is critical for the aviation industry to adapt and embrace the innovative solutions that address these challenges.

Airports depend on the efficient coordination of various departments and personnel for the safety and operational purposes. Among these essential groups, human maintenance operators play a critical role in ensuring that airport infrastructure, equipment, and systems are in optimal condition. Maintenance staff are responsible for performing routine checks, repairing, and maintaining crucial systems. They ensure that the airport functions without any interruption. However, despite their significant contributions, maintenance workers are often overlooked in airport security measures, which can lead to vulnerabilities in airport safety and operational efficiency. Ensuring the well-being and fitness of maintenance staff is crucial, as fatigue, intoxication, or other physical impairments can compromise their ability to perform tasks effectively and safely. This can result in increased risks of accidents, equipment malfunctions, and potential disruptions in airport operations.

This project focuses on the development and implementation of a novel AI-based surveillance system that detects and tracks human poses to classify the fitness levels of maintenance operators, with the goal of integrating this model with SAAB's inbuilt SAFE (Situational Awareness for Enhanced Security) subsystem. The integration of these advanced technologies will enable airport authorities to proactively identify potential risks associated with unfit, unhealthy, drunk, or violent maintenance operators, ensuring a secure and efficient aviation environment. By monitoring the physical well-being and performance of maintenance workers, airport authorities can proactively address potential safety risks, that could compromise the effectiveness of maintenance tasks and overall airport operations.

By leveraging state-of-the-art technologies, including pose estimation algorithms and advanced computer vision techniques, this system will accurately and non-invasely monitor the physical activities and movements of maintenance operators. Once integrated with SAAB's SAFE subsystem, the combined solution will provide real-time insights into the performance and well-being of maintenance operators.

The report also discusses the potential benefits and challenges of this integration, and how it can contribute to enhanced airport security and operational efficiency. By focusing on the often-overlooked maintenance workforce, this project aims to foster a more comprehensive and effective approach to airport safety, benefiting passengers, staff, and stakeholders alike.

Our project objective and successful criteria will be presented in Chapter 2. Project Management techniques would be presented in Chapter 3. Ethical considerations is discussed in Chapter 4 Detailed research and findings on existing and related AI solutions are presented in Chapter5.

The structure used to develop the AI-based pose detection and fitness assessment model, as well as the steps involved in integrating it with SAAB's SAFE subsystem are presented in Chapter6. In this project, we also create our own dataset,DaViD. Details would be presented in Chapter 7. The training strategy and the proposed architecture of the LSTM Classifier, DaVi is presented in Chapter 8. The SAAB SAFE integration methodology is presented in Chapter 9. The proposed system architecture is detailed in Chapter 10.The result of our proposed system and model is prsented in Chapter 11. Finally, conclusion and future work is disscussed in Chapter 12

Chapter 2

Project Brief

2.1 Objective

With greater technological advancements in the fields of artificial intelligence and machine learning, a need is felt to provide greater autonomy to machines in assisting humans in surveillance and other security aspects. The primary objective of this project is to develop an AI-based pose detection and fitness assessment model for human maintenance operators and integrate it with SAAB's inbuilt SAFE subsystem to enhance airport security and operational efficiency. The sub-objectives are:

1. To study the existing research and findings carried out in the fields of enhancing surveillance at airports using AI and ML methods.
2. To study the various existing state-of the-art pose-estimation methods and select the one that suits the best to our use case.
3. Collection of datasets as per requirements.
4. Implement the selected pose-estimation method and integrate the same with an appropriate classifier to detect various activities.
5. Integrate the AI sub-system with SAAB's SAFE subsystem.
6. Evaluate the performance in a real-time environment.

2.2 Scope for Contribution

The scope for contribution in this project can be divided into several areas. The key areas of contribution include:

Research and Development: Investigating and reviewing the state-of-the-art pose detection models and techniques to identify the most suitable solutions as per the project's specific requirements and to improve the accuracy, efficiency, and robustness of the AI-based pose detection and fitness assessment model.

Data Collection and Annotation: Collecting data is a challenge and custom dataset were prepared to cater for the needs of the project. The dataset was annotated and pre-processed to ensure its quality and relevance to the project's objectives.

Model Implementation and Optimization: Implementing and optimizing the selected pose detection model (YoloV7 in our case), integrating it with SORT (simple online and real-time tracking) to track each selected person, and then classifying the state of the maintenance operator into fit/unfit, using the LSTM classifier. The project involved experimentation with different model architectures, hyperparameters, and additional features to enhance the overall performance of the integrated system.

System Integration and Testing: The project offered an opportunity to work closely with SAAB's technical team to ensure seamless integration of our AI-based pose detection and fitness assessment model with the SAFE subsystem. Extensive testing and validation of the integrated solution can be carried out in real-world airport scenarios to evaluate its performance and identify any areas for improvement.

Training and Support: As a furtherance to the existing project, training and support to airport staff and maintenance operators can be provided to help them effectively utilize the integrated system and adopt best practices for ensuring their safety and well-being.

Future Enhancements and Research: We can identify potential areas for further development and improvement of the integrated solution, such as incorporating additional data sources or advanced AI techniques.

By providing opportunities for contribution across a wide range of areas, the project encourages collaboration, innovation, and knowledge sharing among team members and stakeholders, ultimately leading to a more successful and impactful solution.

2.3 Expected Outcomes

Upon successful completion of the project, the integrated AI-based pose detection and fitness assessment model is expected to achieve the following outcomes:

Accurate Pose Detection and Tracking: The integrated solution should accurately detect and track the poses of human maintenance operators in various airport settings, considering the unique challenges and requirements of airport maintenance operations.

Real-time Fitness Classification: The LSTM classifier should classify the fitness levels of maintenance operators in real-time, enabling airport authorities to proactively identify potential risks and take necessary action to ensure the safety and efficiency of maintenance and operational tasks.

Enhanced Airport Security: The integration of the AI-based pose detection and fitness assessment model with SAAB's SAFE subsystem should contribute to improved airport security measures by providing valuable insights into the physical well-being and performance of maintenance personnel.

Improved Operational Efficiency: The integrated solution should help airport authorities optimize maintenance workflows and resource allocation, leading to more efficient airport operations and reduced downtime due to maintenance-related issues.

Scalability and Adaptability: The developed solution should be scalable and adaptable to different airport environments and scenarios, ensuring its applicability across a wide range of settings and requirements.

By achieving these outcomes, the integrated AI-based pose detection and fitness assessment model with SAAB's SAFE subsystem will contribute to a more comprehensive approach to airport security, focusing on the often-overlooked maintenance workforce and ensuring a safer, more efficient aviation environment for passengers, staff, and stakeholders alike.

2.4 Methodology

The methodology followed in this project can be broken down into the following main steps:

Literature Review and Model Selection: An extensive literature review was conducted to study the existing state-of-the-art human pose estimation models, including OpenPose, AlphaPose, YoloPose, and MediaPipe. These models were compared based on their performance, accuracy, robustness, computational efficiency, and suitability for the specific use case of monitoring maintenance operators in an airport setting. After careful evaluation, the YOLO pose detection model was selected as the most appropriate solution for this project.

Incorporating SORT: A SORT algorithm was used to track each person in a frame. Then the keypoints were extracted and pre-processed.

Classifier Selection: After the key points were extracted from the pose estimation technique that our model employs the next task was to classify the keypoints extracted to make a classification. A Long Short-Term Memory (LSTM) classifier was applied to the extracted key points to classify the activities of the maintenance operators. The Long Short-Term Memory LSTM classifier was chosen due to its ability to effectively handle time-series data and model long-range dependencies, which is essential for accurate activity recognition in this context.

SAAB's SAFE subsystem: The integration with SAAB's SAFE subsystem contributes to a more comprehensive approach to airport security, focusing on the often-overlooked maintenance workforce.

By following this methodology, the project successfully developed and integrated an AI-based solution for detecting and tracking the poses of human maintenance operators and classifying their fitness levels.

2.5 Success Criteria

The project will be deemed successful if the integrated AI-based pose detection and fitness assessment model with SAAB's SAFE subsystem demonstrates a measurable improvement in airport security, operational efficiency, and the overall well-being of human maintenance operators.

Chapter 3

Project Management Techniques

To ensure the successful completion of the AI-based pose detection and fitness assessment model and its integration with SAAB's SAFE subsystem, several project management techniques were employed. These techniques helped to plan better, organize more efficiently, and control resources effectively, leading to better project outcomes.

Agile Project Management: We focused on the flexibility and iterative progress, allowing the team to adapt and respond to changes quickly. By breaking the project into smaller, manageable tasks or sprints, the team could develop, test, and refine the solution in incremental stages. Regular meetings and reviews ensured that progress was monitored, and any issues or challenges were addressed promptly.

Work Breakdown Structure (WBS): A work breakdown structure helped to break the project into smaller, manageable components, making it easier to estimate the effort, time, and resources required for each task. By creating a clear visual representation of the project tasks and their dependencies, the team could better understand the project scope and assign responsibilities effectively.

Gantt Chart: A Gantt chart is a visual representation of the project timeline, showing the start and end dates of each task, as well as their dependencies. By using a Gantt chart, the team could track the progress of each task and monitor the overall project timeline, ensuring that deadlines are met, and any delays are addressed promptly.

Risk Management: Proactive risk management is crucial for the success of any project. We identified potential risks and their impacts, and then developed contingency plans and mitigation strategies to address them. Regular risk assessments were conducted throughout the project to ensure that any new risks are identified and managed effectively.

Resource Allocation: The group determined the resources required for each task, such as personnel, equipment, and budget. The team ensured that resources were used optimally and that any potential bottlenecks or resource conflicts are resolved. Further, to avoid duplication of efforts, the team was often broken into two sub-teams, each of which were given independent tasks and the progress was regularly monitored.

Communication Plan: A well-defined communication plan is crucial for ensuring that all team members and stakeholders are kept informed of the project's progress, milestones, and any issues that arise. The communication plan outlined the roles and responsibilities of each team member as the project developed.

Monitoring and Control: Regular monitoring and control of project activities ensured that the project stays on track in terms of time and quality. We set performance indicators and tracked the progress against these indicators. To effectively handle the extensive array of project tasks and system elements, every team member was assigned a primary role and

specific responsibilities. Throughout the project's duration, everyone was responsible for overseeing their designated areas, guaranteeing the accomplishment of tasks, and bringing up any concerns during regular team evaluation meetings.

Easy access to Codes. A GitHub repository was made, wherein all the codes were stored for easy access to all group members. It ensured that tasks are not delayed or hampered due to unavailability of required codes. Further, it allowed all members to contribute and share their work on a common platform.

By employing these project management techniques, the team could effectively plan, execute, and monitor the AI-based pose detection and fitness assessment model integration with SAAB's SAFE subsystem, ensuring the successful completion of the project and the achievement of its objectives. The broad work carried out by the team is shown in Figure 3.1.

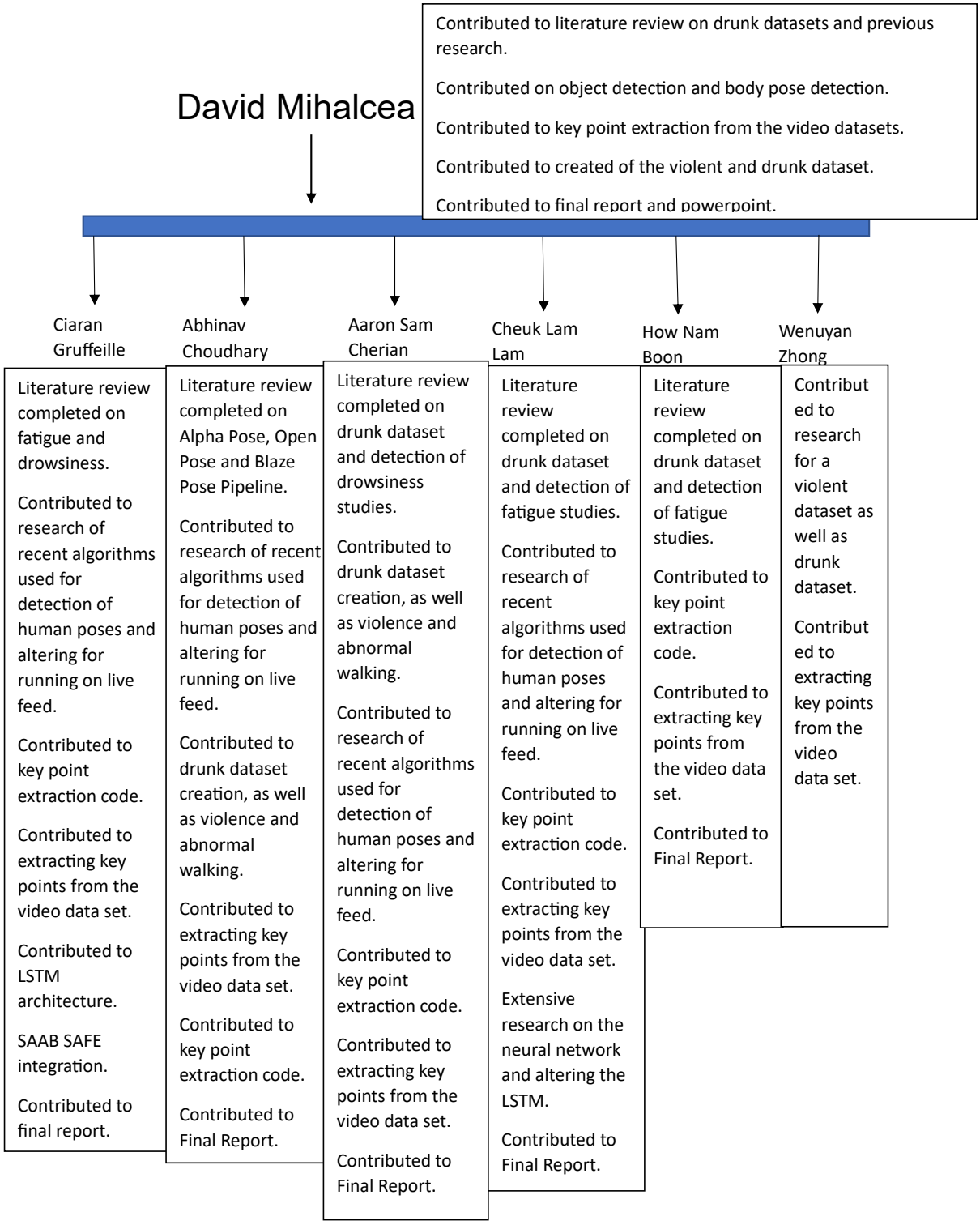

Figure 3.1: Team structure and distribution

Chapter 4

Ethical Concerns

The result of this project may face several ethical issues, primarily related to privacy, data security, fairness, and transparency. It is crucial to address these concerns to ensure that the development of our project is responsible and its deployment is ethical.

The continuous monitoring and tracking of maintenance workers can raise concerns about the invasion of privacy. It is essential to ensure that the system adheres to relevant data protection regulations and guidelines, such as General Data Protection Regulation (GDPR), and that personal data is collected, processed, and stored in a manner that respects individual privacy rights.
Possible solutions:

- Obtain informed consent from maintenance workers before collecting and processing their data.
- Implement strict access controls to limit access to sensitive data to authorized personnel only.
- Anonymize data whenever possible to protect the identities of individuals.

The data collected is vulnerable to potential misuse or unauthorized access. This poses a significant risk to the privacy of maintenance workers and the security of the airport. Such risks can be mitigated by ensuring the security of data storage and transmission.
Possible solutions:
Possible solutions:

- Encrypt data during storage and transmission to prevent unauthorized access.
- Regularly update and maintain security protocols and infrastructure to protect against potential vulnerabilities or cyber-attacks.

AI models can inadvertently perpetuate existing biases if they are trained on biased or unrepresentative data, which may lead to unfair treatment of certain groups of maintenance workers. Ensuring fairness in the AI model's performance is crucial to prevent discrimination and promote equal treatment of all individuals.
Possible solutions:

- Use diverse and representative data samples during the training process to minimize potential biases.
- Continuously evaluate the performance of the AI model to identify and address any disparities in its outcomes for different groups of maintenance workers.

The "black box" nature of some AI models can make it difficult for stakeholders to understand the rationale behind the system's decisions, raising concerns about transparency and accountability.
Possible solutions:

- Implement explainable AI techniques to provide insights into the decision-making process of the AI model, helping stakeholders understand and trust its outcomes.
- Establish clear lines of accountability for the development, deployment, and maintenance of the system, ensuring that any issues or concerns can be addressed promptly and effectively.

Continuous monitoring and fitness assessments may have unintended consequences on worker autonomy, potentially leading to increased stress, reduced job satisfaction, or a feeling of constant surveillance.
Possible solutions:

- Clearly communicate the purpose and benefits of the system to maintenance workers, emphasizing its role in enhancing safety and operational efficiency.
- Encourage open dialogue with workers to address any concerns or challenges they may face because of the system's implementation.

By addressing these ethical issues proactively, we can ensure responsible and ethical development and deployment of the AI-based pose detection and fitness assessment model integrated with SAAB's SAFE subsystem, ultimately leading to a safer and more efficient airport environment that respects the rights and well-being of all individuals involved.

Chapter 5

Literature Review in Smart/Safe Airports and related AI solutions

5.1 Existing Research Works on Human Pose

Human pose estimation techniques are increasingly being used to detect unfit workers, especially in industries where safety is critical, and where proper posture and movement are essential to prevent injuries. There has been a lot of research towards detecting fatigue, drowsiness, intoxication, and violent behaviour among workers using human pose estimation techniques. The goal of this project is to explore the potential of the existing techniques in improving workplace safety and reducing work-related injuries by identifying individuals who may not be physically fit to perform certain tasks, are drunk or tired. We reviewed various techniques, including pose estimation frameworks, deep learning, and computer vision algorithms, as well as their applications in monitoring workers' movements, identifying intoxication, and violent behaviours and thereby predicting potential injuries.

There have been studies that detect the state of drunkenness by variations in postural sway. As per [1], postural sway of humans increases with age. The older age groups demonstrated larger sway areas, higher sway velocities, and longer sway path lengths compared to the younger age groups. The study measured postural control among normal people of age 20 to 70 years by using posturographical parameters. The standards used were gender, age, body weight, cigarette, and alcohol consumption. It was found that if the person consumed a modest amount of alcohol the total sway increased by .01%. However, there was also an increased sway in the person if they were older in age or were hypermobile. In fact, if the person was hypermobile the body sway increased by 0.06%. However, there was no real difference in sway between males and females.

As per [2], an alternative approach to posturography using an inexpensive Kinect depth camera to localize the Center of Mass (CoM) of an upright individual was proposed. The method measures postural sway directly from CoM trajectories. It is obtained from tracking the relative position of three key joints. The study compared this method with the output of a NeuroCom SMART Balance Master using 15 healthy participants. The CoM position was calculated frame-by-frame taking the Euclidean average of the left-hip, right-hip and mid-spine joints as shown in Figure 5.1

Each person's posture was viewed from the side, Anteroposterior (AP), from the rear and mediolateral (ML). This was calculated frame by frame.

The overall movement was calculated and became the resultant time series (RD) throughout the experiment.

$$RD = \sqrt{\left(ML_i - \bar{ML}\right)^2 + \left(AP_i - \overline{AP}\right)^2} \tag{5.1}$$

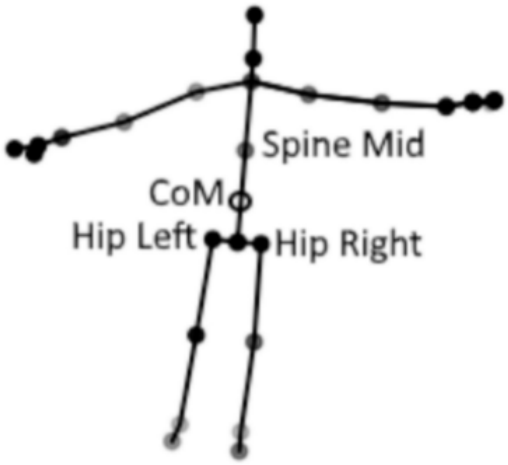

Figure 5.1: Details of joint position to estimate CoM: [2]

The total sway was calculated using RD previously calculated and N, the number of points in the time series. This allowed to find the RMS sway.

$$RMS_{\text{sway}} = \sqrt{\frac{\sum_{i=1}^{N} RD^2}{N}} \tag{5.2}$$

The study found that the proposed pipeline using a Kinect depth camera agrees well with the Balance Master for static assessments of balance with eyes open. However, the disagreement increased under more challenging conditions. The study recommended further investigation with a wider variety of devices and a larger database.

Fatigue detection via pose variations

Human pose estimation techniques can be applied to monitor the movements and posture of workers in real-time. These techniques can be used to detect if a worker's posture is incorrect or he is unfit, which may help to prevent potential injuries. By providing real-time feedback, workers can be alerted to correct their posture and movements, thereby reducing the risk of injury, and improving overall workplace safety. In research carried out by [3], researchers have proposed systems that utilize depth cameras or Kinect sensors to monitor workers' movements and provide feedback on improper lifting techniques or body posture. As per [4], most of the existing systems use either contact or non-contact approaches to detect drowsiness, depending on the application. In contact approaches, the worker needs to wear or touch some equipment for measurement purposes, which have been found to be cumbersome, not user-friendly and time consuming. Most non-contact approaches rely on facial landmarks, retina scan and variation in eye-blinks to detect drowsiness, which is often inaccurate and unreliable. Non-contact approach, as proposed in this project, using real time human pose estimation techniques to detect unfit worker or fatigue is a novel way that overcomes all the drawbacks of contact approaches.

Human pose estimation can be used to identify signs of fatigue and stress in workers. Studies have shown that fatigue and stress can lead to a decreased ability to maintain proper body posture and may increase the risk of accidents and injuries. By analysing workers' movements, human pose estimation algorithms can detect subtle changes in posture and behaviour that may indicate fatigue or stress, allowing employers to take preventative measures, such as providing workers with additional breaks or implementing stress reduction programs.

Drunk detection by percentage of eye closures (PERCLOS) method

Similarly, it was found that drowsiness and fatigue can also be detected using same detection strategies. As per [5], behavioural parameters, such as eye closure ratio, eye blinking, head position, facial expressions, and yawning are non-invasive measures for drowsiness detection. The Percentage of eye Closures (PERCLOS) is also one of the most frequent used metrics in drowsiness detection based on eye state observation. PERCLOS is the ratio of eye closure over a period, and then on the result of PERCLOS, eyes are referred as open or closed. Yawning based detection systems analyse the variations in the geometric shape of the mouth of drowsy driver such as wider opening of mouth, lip position, etc. Likewise, there are lot of research that use facial expressions or thermal imaging techniques to detect drunk state of drivers. But these studies are not based on human pose and primarily use only eye or face for detection purposes. Further, the efficiency of these methods is restricted to the proximity of the person to the cameras.

Advantages of pose estimation methods over techniques that rely on facial expressions and eye blinking

Pose estimation methods analyse the entire body, including posture, joint angles, and movement patterns. This allows the system to detect signs of fatigue, intoxication, violent behaviour, or unfitness that may not be apparent through facial expressions or eye blinking alone. Facial expression recognition and eye blinking analysis can be affected by lighting conditions, occlusions (e.g., glasses or face masks), or the orientation of the face. Pose estimation, on the other hand, is less sensitive to these factors as it considers the entire body and can still provide valuable insights even when the face is partially or fully obscured. Maintenance operators often work in dynamic environments with varying lighting conditions, backgrounds, and camera angles. Pose estimation models can be more robust in these challenging conditions, accurately detecting and tracking human poses even when the subject is partially occluded, or the background is cluttered. Pose estimation can detect various indicators of fatigue, drunkenness, or unfitness, such as unstable posture, irregular gait, slow or uncoordinated movements, or an inability to maintain balance. This provides a more comprehensive assessment of the operator's condition than relying on facial expressions or eye blinking alone. Compared to facial expression recognition, pose estimation is less invasive and privacy intrusive as it does not rely on the analysis of facial features. This can be particularly important when considering privacy concerns and compliance with data protection regulations, such as GDPR.

However, it is essential to consider that using pose estimation alone may not be sufficient to classify drunk, fatigued, violent, or unfit maintenance operators accurately. Combining pose estimation with other techniques, such as facial expression recognition, eye tracking, or physiological measures, can improve the overall performance and reliability of the classification system.

In this project, which primarily focuses on human pose for detection purposes, we aim to bring the fact that by continuously monitoring workers' movements and posture, human pose estimation techniques can help predict potential injuries and accidents. Machine learning models can be trained to recognize patterns associated with injury risks, such as repetitive motions or awkward postures, and provide early warnings for workers and supervisors to take corrective action. These predictive capabilities can help organizations proactively address potential hazards, reducing workplace injuries and improving overall safety.

5.2 Existing Pose-Estimation Frameworks

Few popular frameworks that focus on human pose include open pose, alpha pose, yolo pose and media pipe. The group deliberately went through each one of these, during the course of the project, to select the best possible framework for our use case.

5.2.1 Media Pipe

Media Pipe is an open-source framework developed by Google that allows developers to build machine learning pipelines for processing real-time video, audio, and other sensor data. It has been widely used for various applications, including human pose estimation, face detection, and hand tracking. Media Pipe comes with a set of pre-trained models optimized for specific tasks.

Features	**YOLOv7 Pose**	**MediaPipe Pose**
Topology	17 Keypoints **COCO**	33 Keypoints **COCO** + **Blaze Palm** + **Blaze Face**
Workflow	Detection runs for all frames	Detection runs once followed by tracker until occlusion occurs
GPU support	Support for both CPU and GPU	Only CPU
Segmentation	Segmentation not integrated to pose directly	Segmentation integrated
Number of persons	Multi-person	Single person

Figure 5.2: Comparative study between YOLOv7 and Media Pipe framework [6]

However, integrating a custom model to the existing models in media pipe is challenging. In [7], Open pose was compared with Blaze Pose (which is a light wight framework based on media pipe, used for real time pose estimation) on various parameters and was found to be more accurate compared to Blazepose. The paper suggests that Blazepose wrongly identified the objects in the background as keypoints and labelled them inaccurately. Further media pipe could detect keypoints of only one person in a frame and was found not to support multiple persons. In [8], this problem was addressed using YOLO, an object detection model, to detect multiple persons in a frame. The persons detected by YOLO were cropped, then the pose for each person was estimated and finally the image was aggregated together in a single frame.

The research mentioned above including the comparison shown in Figure 5.2 suggested that Open Pose and Yolo Pose are more accurate and better suited for our task.

5.2.2 Open Pose

Open Pose is a popular framework for human pose estimation, which involves predicting the positions of the body joints in images or videos. As per [9], Open Pose framework consists of a deep convolutional neural network that predicts body part locations and part affinity fields (PAFs) to estimate the associations between body parts. The method can estimate the 2D poses of multiple people in real-time with high accuracy. The introduction of Part Affinity Fields (PAFs) is a significant contribution to the field, as it allows for more accurate part association and distinction between body parts of different individuals in a scene. OpenPose achieves real-time pose estimation, making it suitable for various real-world applications that require real-time processing. It addresses the challenging problem of multi-person pose estimation, which is an essential aspect of many applications. The ability to handle an arbitrary number of people in an image or video makes OpenPose a versatile solution for our project.

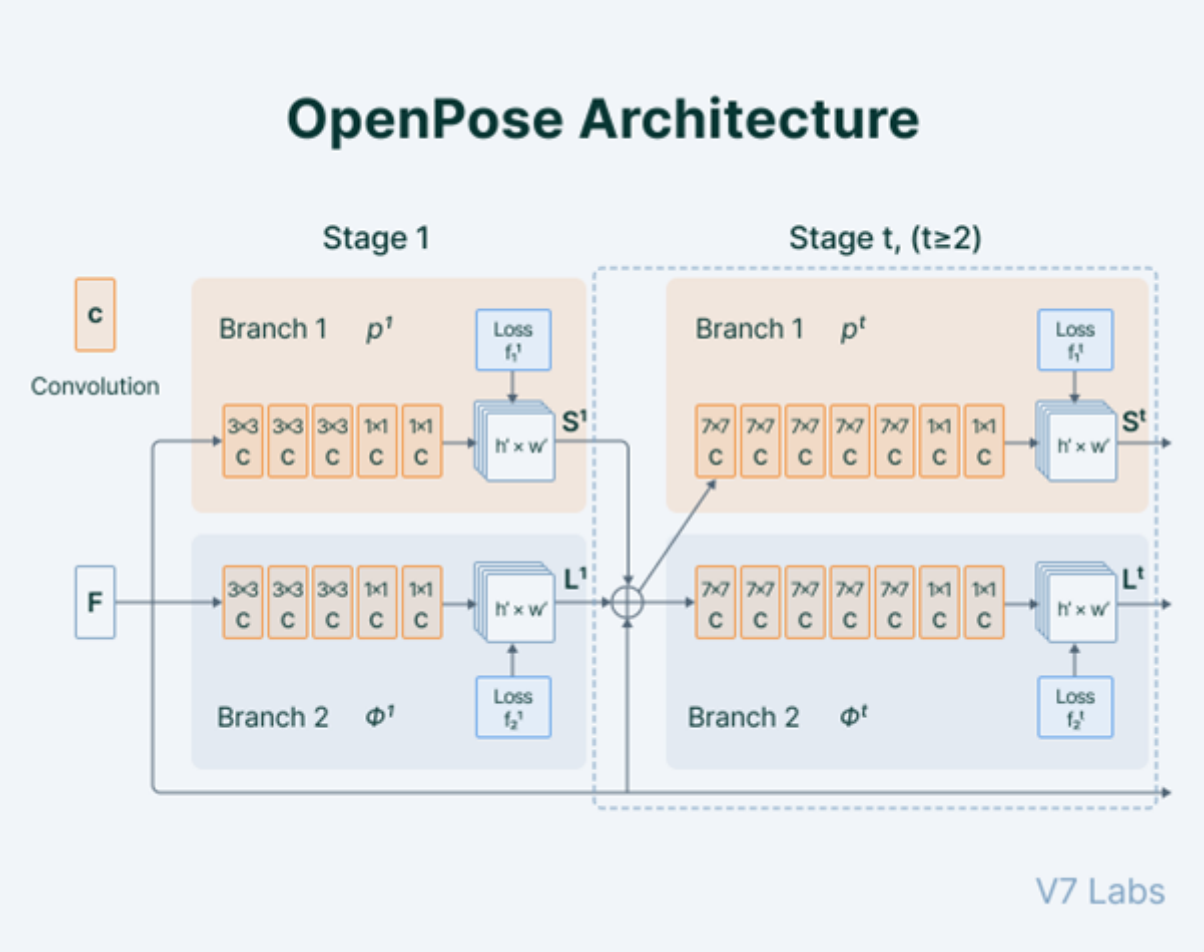

Figure 5.3: OpenPose Architecture [9]

The Figure 5.3 above depicts OpenPose's architecture, which is a multi-stage CNN. Essentially, depending on the number of individuals in the input, the predictions from the two branches, together with the features, are concatenated for the following stage to construct a human skeleton. CNNs are employed in phases to refine the prediction.

Limitations:

1. Open Pose's performance is highly dependent on the quality of the initial person detection. If the person detector fails to detect an individual or generates false positives, it will directly impact the pose estimation results.

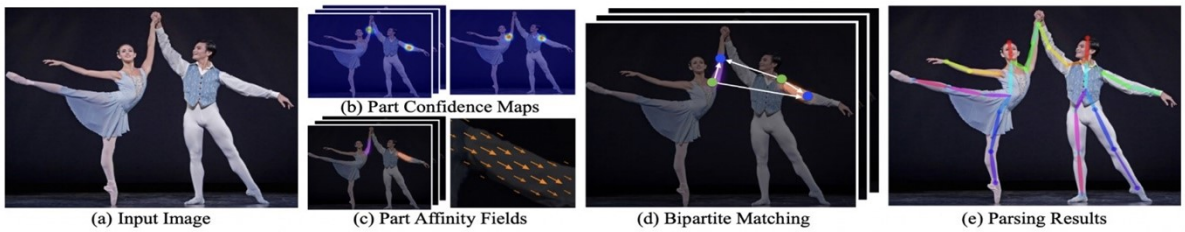

Figure 5.4: OpenPose Pipeline
[9]

2. While Open Pose can handle an arbitrary number of people, its performance may degrade as the number of individuals in a scene increases. This is particularly relevant for crowded scenes, where the computational complexity of the method may become an issue.

3. OpenPose does not have object detection and tracker. We can implement SORT for tracking purposes but since YOLO pose has object detector, integrating SORT into YOLO or Alpha Pose was found to be easier, compared to Open Pose.

4. Although OpenPose is designed to be robust against occlusions, it was found to struggle with extreme poses, body part overlaps, or severe occlusions, which are common in real-world scenarios.

In conclusion, the OpenPose method presents several strengths, particularly in terms of its novel representation using PAFs, real-time performance, and multi-person pose estimation capabilities. However, there are also weaknesses as mentioned above, because of which we moved our focus to YOLO pose.

5.2.3 AlphaPose

AlphaPose is a system for whole-body pose estimation and tracking that uses deep learning techniques to accurately capture the movements of humans in real-time. It is designed to work with video data, and can be used in a variety of settings, from laboratory experiments to real-world scenarios. At its core, AlphaPose uses a combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to detect and track human poses. The system is trained on large datasets of annotated images, such as the COCO dataset, which contains over 200,000 images of people in various poses. One key feature of AlphaPose is its use of Symmetric Integral Keypoint Regression (SIKR), which allows it to localize keypoints quickly and accurately on the human body. This technique involves predicting the location of keypoints based on their symmetry with respect to other keypoints on the body. Another important component of AlphaPose is its use of Parametric Pose Non-Maximum-Suppression (P-NMS), which helps eliminate redundant detections by selecting only the most likely pose estimates. This technique involves assigning scores to each detection based on their likelihood, and then selecting only those with the highest scores. To further improve accuracy, AlphaPose also uses Part-Guided Proposal Generator (PGPG) during training, which helps generate proposals for body parts that are difficult to detect. Additionally, it employs multi-domain knowledge distillation to improve accuracy across different datasets. Overall, AlphaPose has been shown to achieve state-of-the-art performance on several benchmark datasets for whole-body pose estimation and tracking.

AlphaPose in tracking humans

AlphaPose has been shown to outperform other state-of-the-art systems for whole-body pose estimation and tracking in terms of both accuracy and speed. It has achieved top performance on several benchmark datasets, including COCO-whole-body, COCO, and PoseTrack. One key advantage of AlphaPose is its ability to accurately estimate the pose of multiple people simultaneously, even when they are in close proximity to each other. This makes it useful for a wide range of applications, from sports analysis to surveillance. Another advantage of AlphaPose is its real-time performance. It can perform accurate whole-body pose estimation and tracking jointly while running in real-time, making it suitable for use in a variety of settings.

Demerits/ limitations of AlphaPose in human activity classification

While AlphaPose has achieved state-of-the-art performance in whole-body pose estimation and tracking, there are still some limitations to its accuracy and performance. One limitation is that AlphaPose may struggle to accurately estimate poses in certain situations, such as when people are occluded or when they are performing complex movements. This can lead to errors in pose estimation and tracking. To address this limitation, researchers are exploring new techniques for improving the accuracy of AlphaPose. For example, they are investigating the use of multi-domain knowledge distillation to improve accuracy across different datasets, as well as the use of Part-Guided Proposal Generator (PGPG) during training to generate proposals for body parts that are difficult to detect. Another limitation of AlphaPose is its reliance on GPU hardware for real-time performance. This can make it difficult to deploy in certain settings where GPU resources may be limited. To address this limitation, researchers are exploring new techniques for optimizing AlphaPose's performance on CPU hardware. For example, they are investigating the use of quantization and pruning techniques to reduce the computational requirements of the system. Overall, while there are some limitations to AlphaPose's accuracy and performance, researchers are actively working on addressing these issues through new techniques and optimizations.

5.2.4 YOLO-POSE

YOLO-pose extends the YOLOv5 object detection framework by adding keypoint heads for pose estimation. It associates all keypoints of a person with anchors, which allows it to avoid the need for any further grouping. This approach inherently tackles the problem of keypoints of one person being mistaken for another, which can occur in bottom-up approaches [10].

The COCO dataset, which contains 17 landmark topologies,shown in Figure 5.6, was used to train YOLOv7 Pose. It is implemented in PyTorch, which makes customising the code quite simple [11].

As per [10], the authors introduced YOLO-pose, which can detect human key-points without using heatmap. The paper extended the YOLO architecture for multi-person pose estimation and a new loss function; the Object Key point Similarity (OKS) Loss was introduced.

The authors modified the YOLO architecture to incorporate pose estimation capabilities, allowing the model to perform both object detection and pose estimation tasks simultaneously. This single unified network reduced the need for separate object detection and pose estimation pipelines. The modified YOLO architecture was trained on OKS loss, which measures the similarity between predicted and ground truth key points for each object in the image, giving

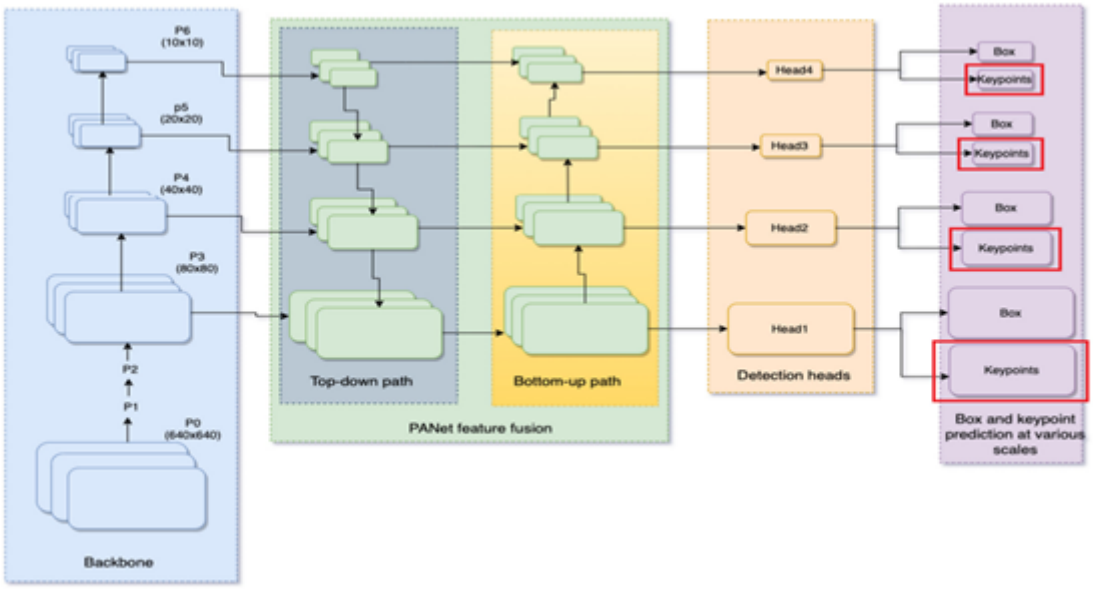

Figure 5.5: Figure showing architecture of YOLO Pose [11]

more accurate key point localization. This approach reduced the computational complexity and improved the accuracy.

Existing state of the art pose-estimation techniques follow either bottom-up or top-down approach. Top-down approach is used to first detect all persons in a frame and then detect key-points for different poses for each individual person. However, as the number of persons increase, the complexity of this approach increases exponentially. Also, they perform poorly in handling occlusion.

Since YOLO is already an object detection algorithm, extending it to include pose estimation is more straightforward than developing a completely new bottom-up architecture from scratch. The top-down approach is also more robust when dealing with occlusions or overlapping body parts, as it first detects individual people and then estimates their poses. This can help avoid issues related to part association that may arise in bottom-up approaches, which rely on heatmaps to detect all the key points of a person in the frame and then applies complex post-processing techniques based on probability to group all key-points into an individual person. Bottom-up approaches are simpler, offer constant runtime but less accurate compared to top-down approaches.

Thus, we chose YOLO-POSE as the pose detection model as it offered the following advantages:

- Fast processing: YOLO is known for its real-time object detection capabilities, making it suitable for use in dynamic airport environments with multiple moving objects.
- High accuracy: YOLO offers competitive accuracy levels compared to other pose estimation models, ensuring reliable detection, and tracking of maintenance operators.
- Scalability: The YOLO architecture is designed to scale well with varying input sizes, allowing it to adapt to different airport settings and camera resolutions.
- Robustness: YOLO demonstrates robust performance in handling occlusions, varying lighting conditions, and diverse human poses, which are common challenges in airport environments.

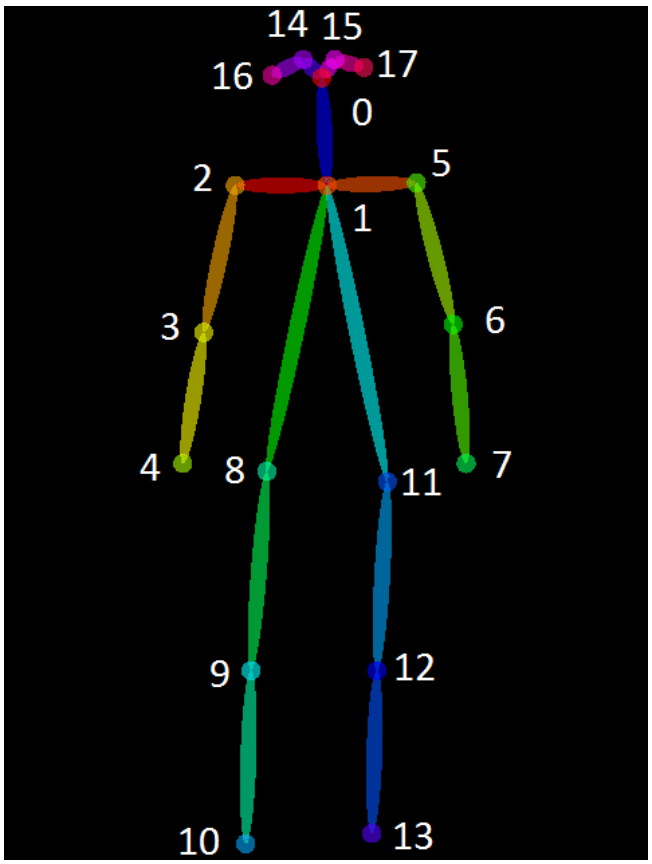

Figure 5.6: 17 keypoints output of YOLO [11]

Challenges. YOLO lacks tracking capabilities. It could effectively detect all persons in a frame but failed to track them. Hence, there was need to integrate YOLO with SORT algorithm, which could associate a tracker id to each detected person in a frame.

5.3 Simple Online and Real-time Tracking (SORT)

The problem with an object detection model, such as YOLOv7 was that for multiple persons in a frame, the YOLOv7, being an object detection model, could detect them, and draw a bounding box across them, but it failed to track each person in a frame, across multiple frames. So, we used SORT algorithm, which associates the entire trajectory for each object detected by the object detection algorithm such as YOLOv7 in a frame, for all the frames. So now, if there are multiple persons in a frame, each of them will be assigned a unique association ID and it will be associated for all the frames as shown in Figure below.

SORT, or Simple Online and Realtime Tracking, is a basic method to track multiple moving objects in video footage. It is designed for situations where we can only access past and current video frames, and it identifies objects quickly as the video plays. The implementation of SORT uses an object detection algorithm, such as YOLOv7 and a system to track the object. It has two main components:

Data association: It means matching or connecting objects in one video frame to the objects in the next frame. SORT uses a technique called the Hungarian algorithm to find the best match between objects detected in consecutive frames. It can be better understood as imagining as if we are following a group of people walking in a video. Data association

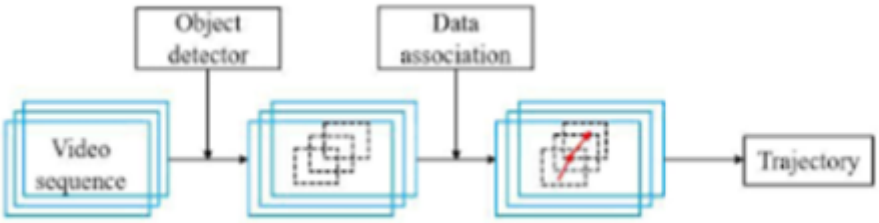

Figure 5.7: Process of tracking object using SORT [12]

helps to match each person from one frame to the next, so we can keep track of them as they move.

State estimation: This part predicts where the objects will be in the future based on their current and past positions. SORT uses a tool called the Kalman filter to make these predictions. The Kalman filter helps to reduce errors and noise in the data, making the predictions more accurate. Imagine we are tracking a group of people walking through a park using a video camera. We want to keep track of each person's position as they move. At the beginning, we use the first video frame to estimate the initial position and velocity of each person. We assume that each person moves at a constant speed, which helps us predict where they will be in the next frame. As we move to the next video frame, we compare our predictions with the actual positions of the people. To update our estimates, we combine the predictions with the new observations to get a more accurate idea of each person's position and velocity. This combination considers the uncertainties in both the predictions and the measurements, resulting in a better overall estimate. We continue this process for every video frame, constantly predicting and updating the positions and velocities of the people in the scene. This state estimation helps maintain smooth and accurate tracking of each person as they move through the park.

5.3.1 Algorithms used in SORT

Hungarian Algorithm

The Hungarian algorithm, also known as Kuhn Munkres algorithm, is an optimization technique used to find the optimal assignment in a weighted bipartite graph. In simpler terms, it helps to find the best way to match two sets of elements, considering the costs or distances between them. The algorithm's main goal is to minimize the total cost of the matching [13].

In the context of SORT (Simple Online and Realtime Tracking), the Hungarian algorithm a crucial role in the data association step. In multiple object tracking, data association involves matching detected objects in one video frame to the objects in the next frame. This matching must be done in a way that minimizes the overall cost, which is usually based on the distance between objects in consecutive frames. It can tell if an object in the current frame is the same as the one in the previous frame by using data association and id attribution [14].

When applying the Hungarian algorithm to SORT, a cost matrix is created where each element represents the cost of matching an object from the current frame to an object in the next frame. The Hungarian algorithm then finds the optimal assignment (matching) that minimizes the

total cost in the matrix. This way, the algorithm helps to maintain the correct identity of each object as it moves through the video sequence, ensuring accurate tracking.

Illustration

It can be understood with the following example:

Imagine we have a video surveillance system in a shopping mall, and we want to track the movement of people in the scene. Our system detects people in each frame of the video and represents them as bounding boxes. We need to match these bounding boxes across consecutive frames to create smooth and continuous tracks of the people.

Let's consider two consecutive video frames, Frame1 (t-1) and Frame2 (t). We have two lists of bounding boxes: a tracking list for Frame1 and a detection list for Frame2.

1. First, we need to define a similarity metric to compare the bounding boxes between the two frames. In this example, we'll use the Intersection Over Union (IOU) score. The IOU score measures the overlap between two bounding boxes, with higher values indicating more overlap.

2. Next, we create a cost matrix where each element represents the IOU score between a bounding box from the tracking list (Frame1) and a bounding box from the detection list (Frame2). The rows in the matrix correspond to the tracked objects, while the columns correspond to the detected objects.

3. We then apply an algorithm, such as the Hungarian algorithm, to find the optimal assignment of bounding boxes between the two frames that minimizes the total cost (in this case, maximizing the total IOU score). This algorithm finds a one-to-one correspondence between tracked objects and detected objects that results in the highest overall similarity.

4. Once the optimal assignment is found, we update the tracks with the new associations. If a tracked object has no corresponding detection in the new frame, it may be considered as occluded or temporarily lost. Conversely, if a new detection has no corresponding tracked object, it may be considered as a new object entering the scene.

5. Finally, we repeat this process for all consecutive frames in the video sequence, ensuring that tracked objects maintain their identities as they move.

In summary, the Hungarian algorithm is a key component of the SORT framework, specifically in the data association step, where it helps find the best match between detected objects in consecutive video frames while minimizing the overall cost. By defining a similarity metric, creating a cost matrix, and using an optimization algorithm like the Hungarian algorithm, we can establish accurate relationships between objects and maintain their identities throughout the video sequence.

Kalman Filter

The Kalman filter is used to predict the future location of the object by consider its past motion. The filter uses a state model that includes the objects' position, velocity, and acceleration, as well as the measurement model that incorporates the measurement noise.

In this case, the Kalman filter is initialized with an initial state estimate of the object's position and velocity, where the initial state estimate is obtained from the object detection in the first frame. At each time step, the Kalman filter predicts the future location of the object based on its past motion. The prediction is made using the state model, which includes the position, velocity and acceleration of the object.

When a new measurement of the object's position is available in the current frame, the state estimate of the object is updated based on the measurement. The measurement update which contains the measurement noise is done using the measurement model. Lastly, the Kalman filter estimates the state of the object at each time step based on the prediction and measurement update steps. The state estimate includes the position and velocity of the object.

In general, the algorithm can predict the future location of the object based on its past motion, which helps to improve the accuracy and reliability of the tracking. On the other hand, the Kalman filter is advantageous in the SORT algorithm in reducing the effect of noisy measurements and occlusions in the video stream, which can cause problems in object tracking [15][16].

5.4 LSTM Classifier

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture capable of learning long-range dependencies in sequential data due to their unique gating mechanisms, which allow them to retain and forget information as needed over time. An LSTM classifier is used for human activity classification to differentiate between human poses because it is particularly effective in handling time-series data. Human activity recognition often involves processing and analysing sequential data, such as sensor readings from wearable devices, video frames, or motion capture data. LSTM networks can learn and remember patterns that span long periods of time, which is useful for recognizing complex human activities that consist of multiple sequential actions. Advantages of LSTM Classifier:

Ability to handle time-series data: LSTMs are designed to manage time-series data and capture temporal dependencies, making them ideal for activity classification based on pose key points.

Handling of long-range dependencies: LSTMs can model long-range dependencies in the data, allowing them to recognize activities that occur over an extended period or involve complex sequences of movements.

Robustness to noise: LSTMs are less sensitive to noise in the data, making them suitable for processing KeyPoint extracted from real-world airport environments, which may contain noise or other imperfections.

Adaptability: LSTMs can be easily fine-tuned and adapted to specific use cases or scenarios, allowing for the customization of the activity classification model based on the unique requirements of airport maintenance operators.

5.4.1 LSTM vs GRU

When determining what type of neural network to use, it was clear for sequential data processing such as the processing of key points extracted from body poses that LSTMs (Long Short Term Memory) and GRUs (Gate Recurrent Unit) are best. The reason they perform similarly is

because they were designed to address the same 'vanishing gradient' problem by introducing gating mechanisms that allows for the neural network to selectively remember and forget information of time based on relevance of the information. However, the LSTM is a three-gate mechanism and the GRU is a two-gate mechanism. The LSTM has an input gate, a forget gate and an output gate, which allows the neural network to control the inflow of new information into the cell, as well as what it forgets and outputs of the cell, whereas, the GRU only has a reset gate and an update gate, which allows it to decide what information it should forget and what information it should add onto [17]. The distinct differences in gates are designed to give each RNN a distinct advantage, as the LSTM is better for long term dependencies in data as it can selectively forget or remember information, whereas the GRU is more computationally efficient than the LSTM while still providing similar benefits. The research from multiple previous studies suggests they perform very similarly and are hard to distinguish from one another, however, the trend from these research papers shows that the GRU is computationally more efficient in all cases, while the LSTM tends to perform better over an extended period given its advantages [17]. The goal of this project was to provide the most robust system we could provide, and in the interest of doing so the group chose the LSTM because it performed well in sequential tasks such as those to process key point arrays, and it had an edge over the GRU over a longer period [18].

Chapter 6

Proposed AI Subsystem

6.1 YOLOv7 for Human Pose Estimation:

We employed the cutting-edge object identification model YOLOv7 for this assignment, which can identify several objects in an image or video and forecast their related bounding boxes and class labels. The YOLO (You Only Look Once) architecture, which is renowned for its quick and precise object identification capabilities, is used in YOLOv7. Convolutional layers and feature maps are used in YOLOv7's deep neural network to detect a person's body's major features. The YOLOv7 architecture is made up of a backbone network that extracts feature maps from the input picture and many detection heads that forecast keypoints for everyone in the image. The joints of the arms, legs, chest, and head are typically the key points.

YOLO-pose associates all the key points of a person with anchors, instead of using heatmaps. It is based on YOLO object detection framework. It associates each person with 17 key points, which correspond to different joints of human body. Each of these key points is again identified with its location and confidence. Thus, for each anchor, 51 elements are predicted by key-point head and the box head predicts 6 elements. For an anchor with n key points, the overall prediction vector is defined in equation 6.1

$$P_v = \{C_x, C_y, W, H, \text{box}_{\text{conf}}, \text{class}_{\text{conf}}, K_x^1, K_y^1, K_{\text{conf}}^1, \ldots\ldots, K_x^n, K_y^n, K_{\text{conf}}^n\} \tag{6.1}$$

Where, Cx and Cy are X and Y coordinates of bounding box, respectively. W, H are Width and Height of bounding box, followed by confidence score of bounding boxes, class of bounding box. Then, for next 17 key points, we have 3 elements each representing (x,y,confidence).

6.2 Simple Online and Realtime Tracking (SORT)

We integrated YOLOv7 pose estimation model with SORT, to achieve real-time multiple persons tracking in video sequences. Here's an outline of how we combined the two methods:

- The YOLO algorithm was applied on each video frame to detect people in the scene. YOLO will generate bounding boxes around each person along with the corresponding confidence scores.
- Detections with low confidence scores can be filtered out or non-maximum suppression can be applied to remove overlapping boxes, to obtain a refined list of bounding boxes for each frame.
- SORT algorithm's data association component was to match the bounding boxes of detected people in the current frame with the tracked objects from the previous frame.

This can be done by creating a cost matrix based on the IOU (Intersection Over Union) scores between the bounding boxes and using an algorithm like the Hungarian algorithm to find the optimal assignment that maximizes the total IOU.

- The state estimation component of SORT, such as a Kalman filter, was applied, to predict and update the positions and velocities of the tracked people. This helped to improve the tracking accuracy and handle noisy measurements from the object detection step.
- Update the tracks of the people with the new associations and handle new people entering the scene or tracked people leaving the scene.
- The above steps were repeated for each video frame in the sequence, ensuring that people are accurately tracked as they move through the scene.

By combining YOLO's object detection capabilities with SORT's data association and state estimation techniques, we could achieve real-time multiple persons tracking in video sequences.

6.3 Keypoint to Feature Transformation:

As we are aiming to classify the movements of people and not their body positions, we need to transform the keypoints extracted by yolov7 into momentum and angles. To do this we apply the equations 6.2 to each frame:

$$\begin{aligned} \text{Momentum } {}_{t}^{N} &= \sqrt{\left(L_{Nx(t+1)} - L_{Nx(t)}\right)^2 + \left(L_{Ny(t+1)} - L_{Ny(t)}\right)^2} \\ \text{Angle } {}_{Nt} &= \arctan\left(\frac{L_{Ny(t+1)} - L_{Ny(t)}}{L_{Nx(t+1)} - L_{Nx(t)}}\right) \end{aligned} \tag{6.2}$$

Where N is the keypoint with coordinates (x, y) at a time t. After applying these equations to all the keypoints we obtain the momentum and angle of each keypoint for each frame. We also combined the certainty of each keypoint by take the average of the certainty of the keypoint between two frames. This gives us a total of 51 features that we then input into the LSTM Classifier.

6.4 Action classification using LSTM:

Following the extraction of keypoints from an image or video sequence, the subject's action has to be classified. We employed a Long Short-Term Memory (LSTM) classifier for this work. Recurrent neural networks (RNNs) with the ability to process sequential input, such as LSTMs, are ideally suited for tasks like action categorization. The sequence of keypoints collected from YOLOv7 is sent into the LSTM classifier, which analyses the keypoints in a time-based manner to capture the temporal dynamics of the activities being taken. A probability distribution covering the many action classes, including drunk, limping, and aggressive conduct, is the LSTM classifier's output.

6.5 Integration with the SAFE system from SAAB:

We employed a modular design that provides smooth communication between the AI model and the SAFE system to integrate the AI model with SAAB's SAFE system. The SAFE system is in

charge of receiving the output from the AI model and initiating the necessary reactions, while the AI model oversees processing the input data (such as pictures or video streams) and providing the output (such as action categorization results). During the integration phase, the interfaces between the SAFE system and the AI model must be defined and made interoperable. The AI model may be implemented on SAAB's SAFE platform and utilised in real-world applications, such as checking for intoxicated or aggressive behaviour in airports.

Integrating Apache Kafka into the pose estimation project can provide several benefits, such as handling real-time data streaming and communication between different components of the system. Kafka enables real-time data streaming between our pose estimation model and other system components (e.g., SAAB's SAFE subsystem). Kafka provides a more robust, scalable, and efficient system for monitoring worker conditions in real-time, ultimately enhancing the overall effectiveness of the solution.

Kafka was our way of interacting with the SAFE instance implemented on HILDA, Cranfield's supercomputer. We did this by packaging the results from our classifier into a specific format and then sending it to the Kafka streaming platform that would then transform and push this message to the SAFE system. This simplified the process of sending messages from our classifier to the SAFE system.

Kafka enables real-time data streaming between our pose estimation model and other system components. (e.g., SAAB's SAFE subsystem)

To communicate we simply create a producer object in our code and sent a message when an unfit person was detected with their coordinates.

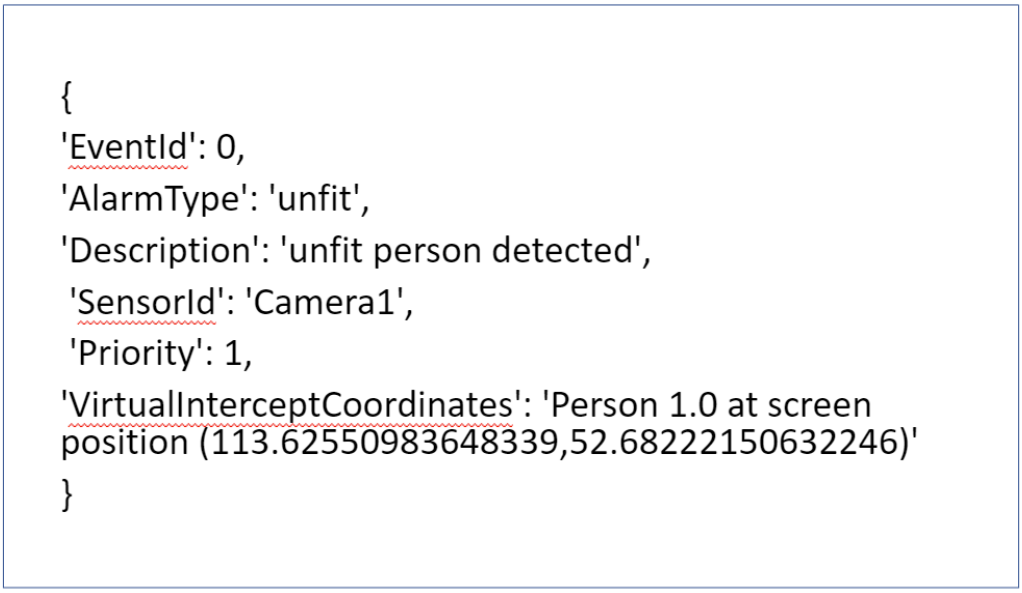

Figure 6.1: Event ID

Chapter 7

Dataset Creation

Datasets and data preparation are essential parts of every successful project in the field of computer vision and machine learning. Although datasets including videos for action recognition are not rare, there is no single dataset that caters to our needs. Therefore, our own database of videos is assembled for use in both Drunk and violent detection tasks.

7.1 Related Work

Table 7.1 summarizes attributes of other datasets. Most datasets contain in-the-wild videos extracted from Youtube or Movies. The Body Language Dataset (BoLD) [19] collected a large-scale of in-the-wild data segmented from movies and reality TV shows. The Dataset of perceived Intoxicated Faces (DIF)[20] and Violent crowds datasets [21] are datasets on YouTue videos. Although in the wild datasets capture a diverse range of situations and are more representative of real-world scenarios. However, the data may be noisy, and containing irrelevant or redundant information.
On the other hand, for the lab data, The Geneva multi-modal emotion portrayals (GEMEP) dataset [22] contains more than 7 000 audio-video portrayals of 18 emotions portrayed by 10 actors in a controlled setting. The videos are filmed from 1 angle in front of the actor. The Violence detection dataset[23] contains high-resolution videos with non-professional actors filmed from 2 different angle performing different tasks such as punching, pushing and hugging. In the lab datasets provide greater control over the data, ensuring that it is clean and consistent. However, they may not accurately reflect real-world scenarios.

Table 7.1: The attributes comparison of our dataset with other widely used datasets for human activities recognition and detection. (Controlled: in the lab, Uncontrolled: in the wild)

Datasets	#Class	#Videos	Setting	Context
GEMEP[22]	18	1260	Controlled	Emotion
BoLD[19]	26	9876	Uncontrolled	Emotion
DIF[20]	2	169	Uncontrolled	Intoxication
Violent crowds[21]	2	246	Uncontrolled	Violent
Violence detection[23]	2	350	Controlled	Violent
Our dataset	2	420	Mix	

7.1.1 Data Collection

A mix of in-the-wild and in-the-lab datasets can have several benefits, as it can combine the strengths of both approaches while mitigating some of their limitations. For our project, the

dataset is a mix of our own recording, a selection of videos from GEMEP and Violence detection datasets and videos from the internet. The dataset is organized into directories as shown in Figure 7.2. After keypoints extraction and transformation on all the subjects in the videos, The dataset has expanded to 992 data points with 440 in fit and 522 in unfit.

Recording:

The video recordings were done with the help of the staff at the Sports Hall on the Cranfield Campus that allowed us to use their hall as well as providing the cones. The recordings were done using a grid, where the cones were added to add confusion to the person, as well as, adding obstacles that were meant to be avoided. There were 3 people recording from different angles, and each person would call for the person to walk in that direction, providing different frames of views for the data set. This process was repeated by everyone present and the videos recorded were aimed to be 2 minutes long, providing us many 3 second videos for training the LSTM.

Figure 7.1: Drunk dataset preparation

GEMEP:

Although the full dataset contains more than 100 videos only 145 videos of the GEMEP Core Set are available for research purposes free of charge. From 145 videos, only videos from the anger and irritation class are selected as unfit data. On the other hand, on relief videos are used as fit data as the actor in other videos does not has much movement.

Violent detection dataset:

Although the datasets contains 230 violent videos, our dataset only selected 40 as some of the videos contain redundant data where not every actor in the videos is performing violent action. As it will be time-consuming to track and label the keypoint manually, only videos which every actor is violent are selected while all 120 non-violent video would be used as fit data.

Online:

We collected a few videos from online platforms such as YouTube to use it for our Drunk dataset. These videos were real-case scenarios of people actually drunk which were recorded which helped

us to evaluate the sway and the behaviour of drunkenness. These videos were trimmed to multiple videos of 3 sec length with the help of a python script. This gave us a total of 50 videos where we used an online platform like YouTube to prepare our dataset.

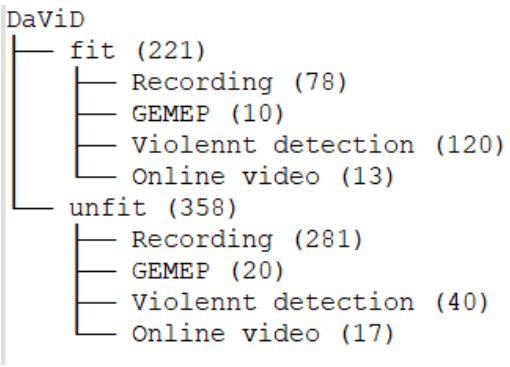

Figure 7.2: The structure of the video repository with the 579 videos, split into fit (221 clips) and unfit (358 clips). The number in the bracket represent the number of videos in the folder

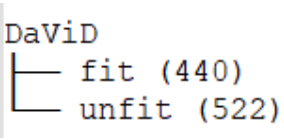

Figure 7.3: The structure of the data repository with the 962 data

Chapter 8

Action Classification Model

Following the extraction of keypoints from an image or video sequence, the subject's action has to be classified. We employed a Long Short-Term Memory (LSTM) classifier for this work. Recurrent neural networks (RNNs) with the ability to process sequential input are ideally suited for tasks like action categorization. Our classifier,DaVi, is a network which comprises recurrent, ReLu and softmax layers. In this chapter, we would discuss the model architecture and its performance.

8.1 Training strategy

The data shown in Figure 7.3 are split in training and validation data in a 80/20 split. As the videos in the dataset has different length. In other to reduce training time, the data should be train in batch. Thus, all the data has to be padded out before sending to train in batch. During training, the padded data would then be packed so that the the useful data is recovered and the network would not be biased toward the padded data. This significantly reduce training time.

Table 8.1: Summary of Hyper-Parameters altered and tested

Hyper-Parameter	Value tested
LSTM Layer	2,3
LSTM cells	256,512
Dense layer	1,2
Dense layer unit	64,128
Dropout rate	0.1,0.4,0.5
Batch size	50,100

Previous work

When designing DaVi, several architectures were considered. Most stat-of-the-art video classifiers are CNN LSTM where feature extraction is done with the CNN layers. Only the LSTM layers of these model are studied.
A Video Classification CNNLSTM with Resnet backend model[24] was studied as it achieves around 0.9 accuracies in its study. It consists of 3 LSTM layers with a softmax classifier. The other model that was studied is the DeepConvLSTM[25]. It consists of 2 LSTM layers with dropouts and a final softmax classifier. It achieved over 0.92 accuracy in its studies. As shown in Figure 8.1, the CNNLSTM and DeepConvLSTM has similar training accuracy. However, CNNLSTM is greatly overfitted as the validation accuracy is much lower.

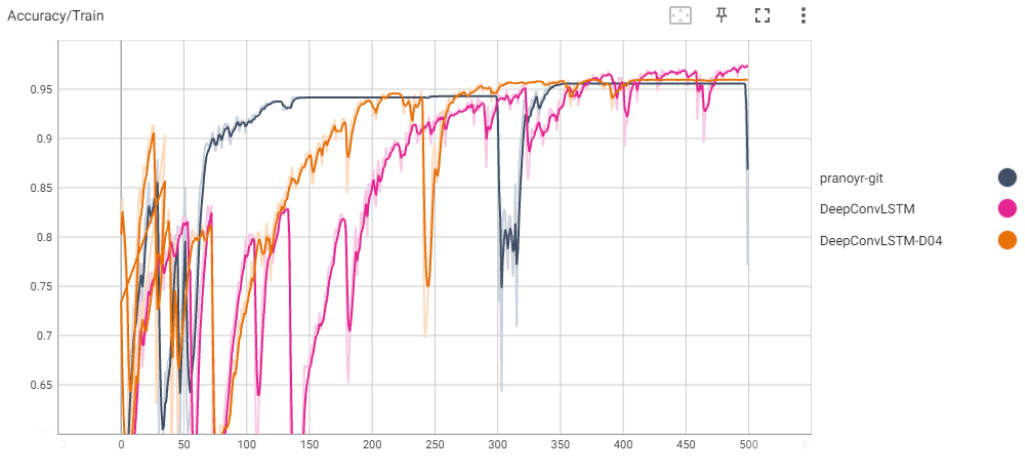

Figure 8.1: Training Accuracy for CNNLSTM(Dark grey), DeepConvLSTM (Pink), DaVi(Orange)

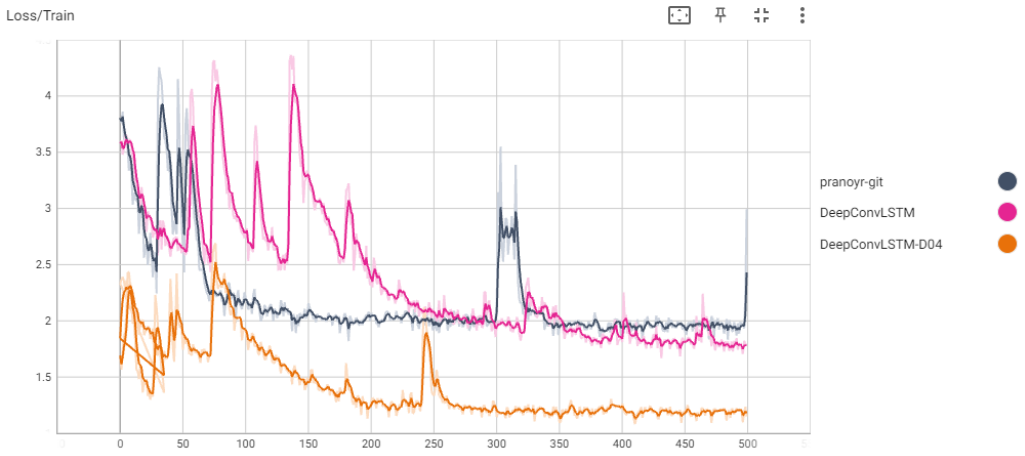

Figure 8.2: Training loss for CNNLSTM(Dark grey), DeepConvLSTM (Pink), DaVi(Orange)

Figure 8.3: Validation Accuracy for CNNLSTM(Dark grey), DeepConvLSTM (Pink), DaVi(Orange)

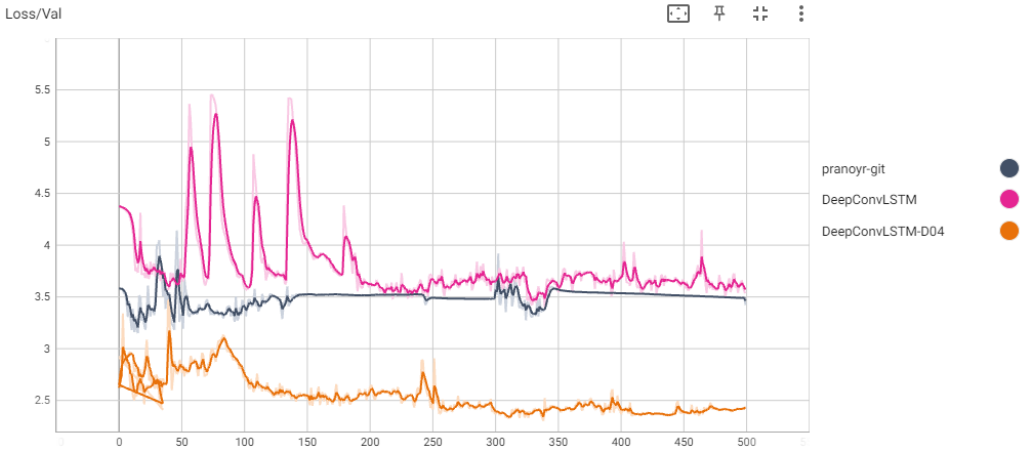

Figure 8.4: Validation loss for CNNLSTM(Dark grey), DeepConvLSTM (Pink), DaVi(Orange)

8.1.1 DaVi Architecture

It is concluded that the lstm model in DeepConvLSTM works better for our data and it is then used as our base model and further modification was made to improve accuracy and reduce loss. Hyper-parameter such as dropout rate, hidden size have been altered and tested.The details of the Hyper-parameter that are changed and tested are summerized in Table8.1. The performance of the different models is shown in Appendix A.1. After training and testing the different model. The final model architecture is shown in Figure 8.5. Davi consists of 2 lstm layers with dropout rates of 0.4 then a dense layer, The units in the dense layers are ReLUs. Finally, the output of the model is obtained from a softmax layer (a dense layer with a softmax activation function), yielding a probability distribution over classes. the shorthand description is $R(256) - R(256) - D(64) - Sm$ Where $R(n)$is recurrent LSTM layer with n cells, $D(n)$ a dense layer with n units and Sm a softmax classifier.
The most notable difference between Davi and the studied network is the extra ReLu dense layer before the softmax classifier. From Figure 8.2 and 8.4, it significantly reduces loss both in training and validation.

Bi-directional LSTM

After the model architecture is decided. Bi-directional LSTM layers are also trained and tested. It was found that a Bi-directional LSTM model would overfit our data and the training time has increased as well. Thus no further modification was made and the architecture shown in Figure 8.5 is finalized.

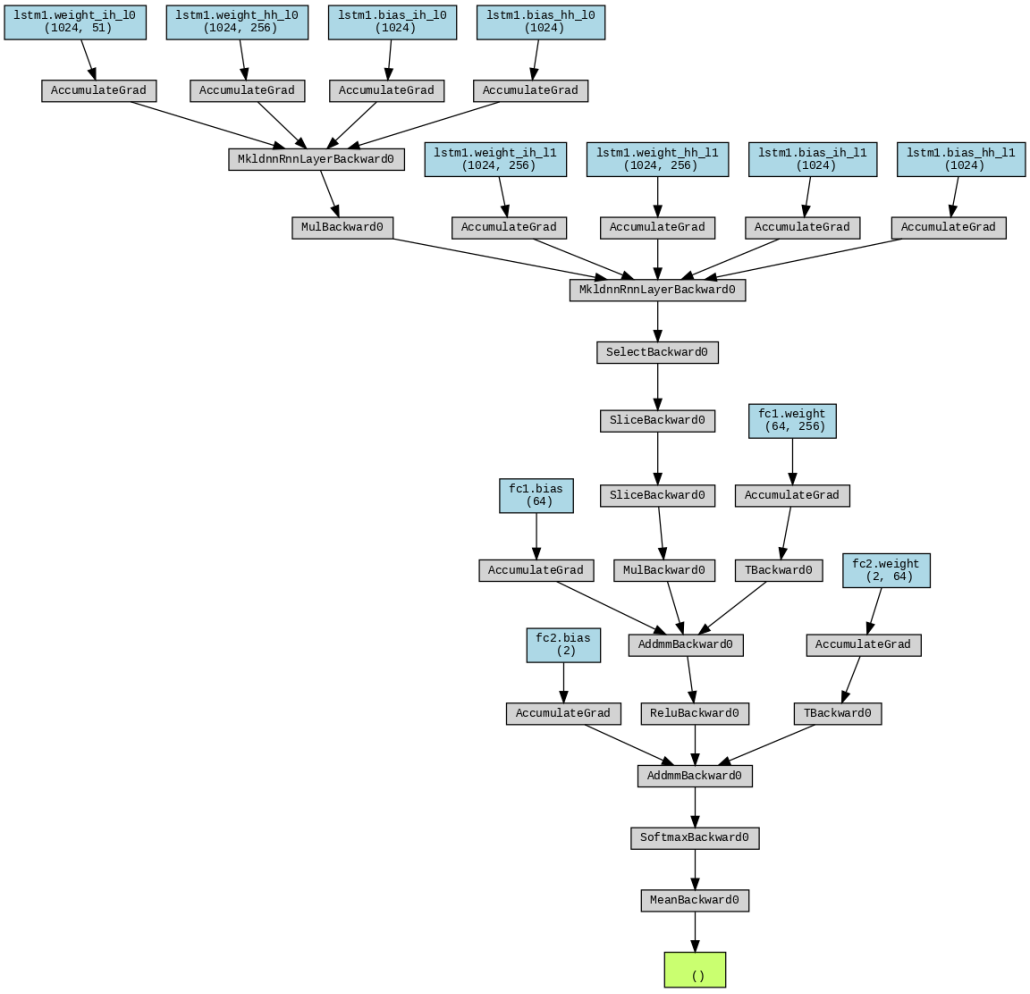

Figure 8.5: DaVi Architecture

Chapter 9

SAAB SAFE Subsystem

The SAFE (Situational Awareness for Enhanced Security) system is a security and surveillance solution developed by Saab, a Swedish company specializing in aerospace, defence, and security technologies. The SAFE system is designed to provide an integrated approach to manage and respond to various security challenges, ranging from public safety and emergency response to critical infrastructure protection and transportation security.

SAFE combines information from various sources like sensors, cameras, and databases, and presents it in an easy-to-understand and accessible format to operators and decision-makers. This helps streamline communication, improve situational awareness, and facilitate a more effective response to incidents.

Overall, the SAFE system by Saab aims to enhance security and safety by providing a holistic, real-time view of the operational environment, facilitating informed decision-making, and streamlining response efforts.

9.1 Kafka

Apache Kafka is a distributed streaming platform that has been designed for high-throughput, fault-tolerant, and scale-able real-time data streaming and processing. It is widely used for building real-time data pipelines and streaming applications. It has a publish-subscribe model, where producers write data to topics, and consumers read data from those topics. Topics are partitioned and replicated across multiple nodes in a Kafka cluster, ensuring high availability and fault tolerance. It is designed to handle a large volume of events and messages, making it suitable for big data applications and real-time data streaming.

Kafka was our way of interacting with the SAFE instance implemented on HILDA, Cranfield's supercomputer. We did this by packaging the results from our classifier into a specific format and then sending it to the Kafka streaming platform which would then transform and push this message to the SAFE system. This simplified the process of sending messages from our classifier to the SAFE system.

To communicate with the Kafka server, we create a producer object in our code with a properly formatted data object comprised of the alarm type, in our case "unfit" as well as the id and position on the screen. Kafka then forwards this information to the SAFE server which creates an alarm.

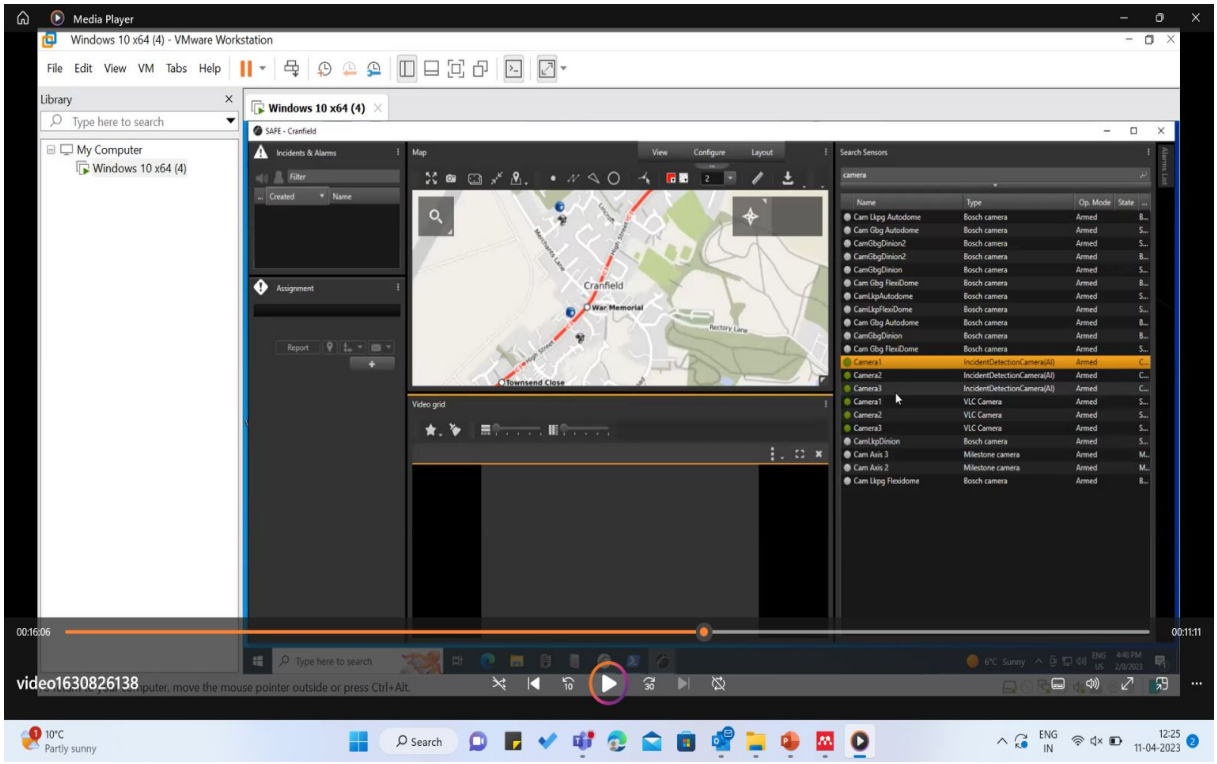

Figure 9.1: SAAB SAFE Interface Dashboard

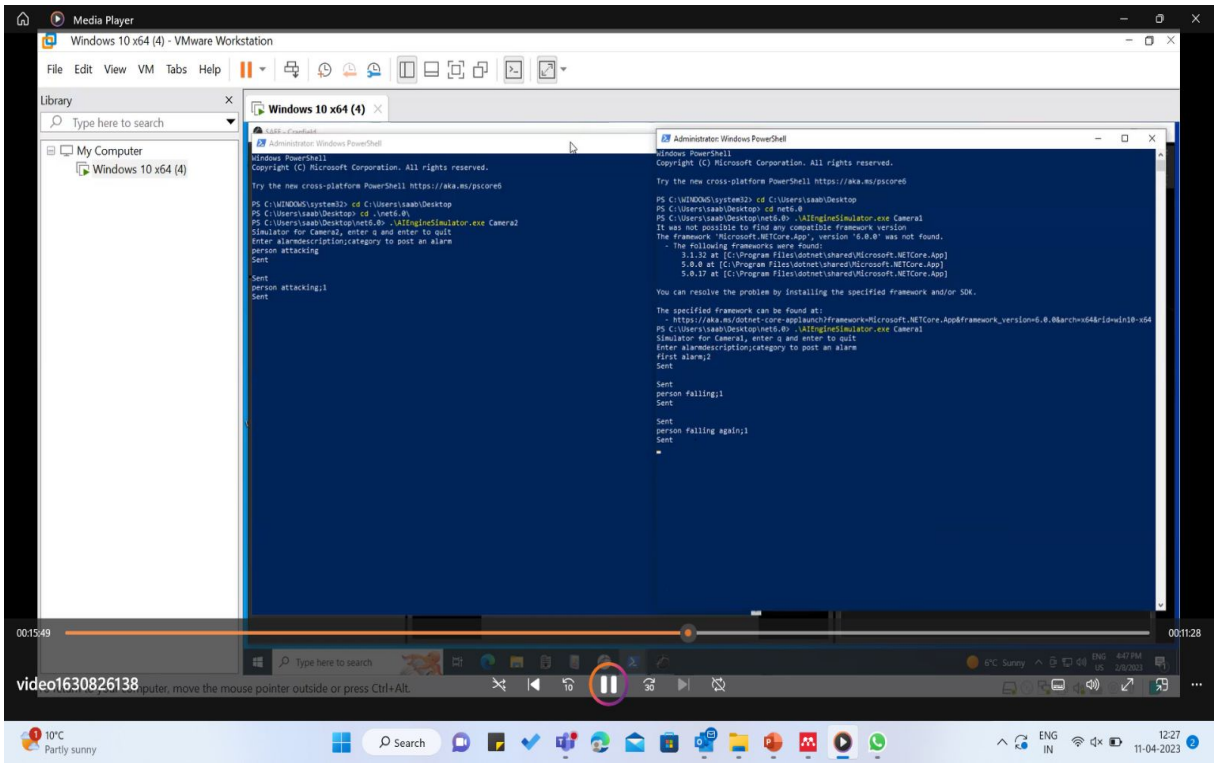

Figure 9.2: Triggering of Alarm

Chapter 10

System Architecture

System architecture refers to the conceptual design of the structure and overall organization of a system which defines how it behaves. It encompasses the components or elements of the system, their relationships, and the rules governing their interaction. System architecture provides a high-level view of the system, illustrating each component of the system and how they work together to achieve the main goal. System architecture helps stakeholders, such as developers, managers, or customers, understand the system's design and functionality at a high level. It also guides the development and evolution of the system, ensuring that it meets the desired requirements, performance, and quality objectives.

The system architecture that has been built for our research on human posture estimation and behaviour categorization have several sub-systems that interact to get the desired result. These sub-systems comprise the AI model for human posture estimation, LSTM classifier for behaviour categorization, and the SAFE system for integrating the AI model with the SAAB firm architecture. For ease of understanding, we have divided it into several layers/stages:

Data Acquisition Layer: It is the first and crucial stage in the AI-based pose detection and fitness assessment system. It is responsible for capturing, pre-processing, and preparing video data from the surveillance cameras installed across the airport for further processing by the subsequent layers. The primary functions of this layer include real-time video feed capture and pre-processing to suit the requirements of pose detection model.

Human Pose Estimation Layer: It is the second stage in the fitness assessment system. It processes the pre-processed video data from the Data Acquisition Layer to detect and track maintenance workers and their poses in the video frames. The primary functions of the human pose estimation layer include:

- Human detection and tracking: It uses the YOLOv7-based pose detection model to detect and SORT algorithms to track humans (i.e., maintenance workers) present in the video frames. It identifies the location and dimensions of each detected person, maintaining their unique identifiers to track them across multiple frames and camera views.
- Pose estimation: Once humans are detected and tracked, the YOLOv7-based pose detection model estimates the pose of each maintenance worker by identifying key body joint points, such as head, shoulders, elbows, wrists, hips, knees, and ankles. The model extracts these key points, which represent the skeletal structure of the worker, and generates a pose representation for further processing.

Feature Extraction and Classification Layer: This is the third stage in the system architecture. It processes the pose information generated by the Human Pose Estimation Layer to identify the activities and fitness levels of the maintenance workers. The primary functions of this layer include Feature extraction, which involves extracting meaningful features from the pose information (keypoints) generated by the Human Pose Estimation

Layer. These features might include relative positions of body joints, joint angles, or motion patterns. The extracted features serve as input for the LSTM classifier and are designed to capture relevant information about the workers' activities and fitness levels.

Activity Classification Layer: Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), are used to classify the extracted features into different activity categories and fitness levels. LSTMs are particularly suitable for this task because they can model long-term dependencies in time series data, making them effective in capturing and understanding the complex motion patterns of maintenance workers. The LSTM classifier is trained on a labelled dataset containing examples of various activities and fitness levels of maintenance workers. During the training process, the model learns to recognize patterns in the feature data that correspond to different activities and fitness levels. The LSTM classifier produces an output representing the predicted activity and fitness level of each maintenance worker. This information can be used to assess the workers' performance, identify potential safety risks, and inform interventions to address any issues.

Data Storage and Management Layer: This layer is responsible for storing and managing the generated pose information, classification results, and any relevant metadata. It can use a secure database or cloud storage system to ensure data integrity, privacy, and compliance with data protection regulations. The layer maintains regular data backups to prevent data loss due to system failures, hardware issues, or other unforeseen incidents. It also incorporates data recovery mechanisms to restore the data from the backups in case of data corruption or loss. The layer enforces data retention policies and complies with data protection regulations, such as GDPR. It ensures that the collected, processed, and stored data adheres to the relevant legal and ethical guidelines, including data minimization, storage limitation, and the right to erasure. This, however, is beyond the scope of the project and can be incorporated as a future enhancement to this project.

Integration with SAAB's SAFE subsystem: The Integration with SAAB's SAFE Subsystem layer is a vital component of the AI-based pose detection and fitness assessment system. This layer connects the AI-based model with the existing SAAB's SAFE subsystem, allowing for seamless exchange of information and insights between the two systems. This layer establishes a secure communication channel between the AI-based pose detection and fitness assessment model and the SAAB's SAFE subsystem. To ensure compatibility between the AI-based model and the SAAB's SAFE subsystem, this layer maps and transforms the data exchanged between the systems into a format that both systems can understand and process. The layer enables event triggering and automation based on the insights generated by the AI-based model.

For example, if the model detects a maintenance worker who is violent or intoxicated or identifies a fitness issue, the layer can trigger an alert or notification within the SAAB's SAFE subsystem, initiating an appropriate response or intervention. This component ensures that the AI-based model's insights are integrated into the SAAB's SAFE subsystem's user interface, visualization, and reporting tools.

By following this system architecture in Figure 10.1, the AI-based pose detection and fitness assessment model integrated with SAAB's SAFE subsystem can efficiently detect, track, and assess the fitness levels of maintenance workers in real-time, providing valuable insights to support airport security and operational efficiency.

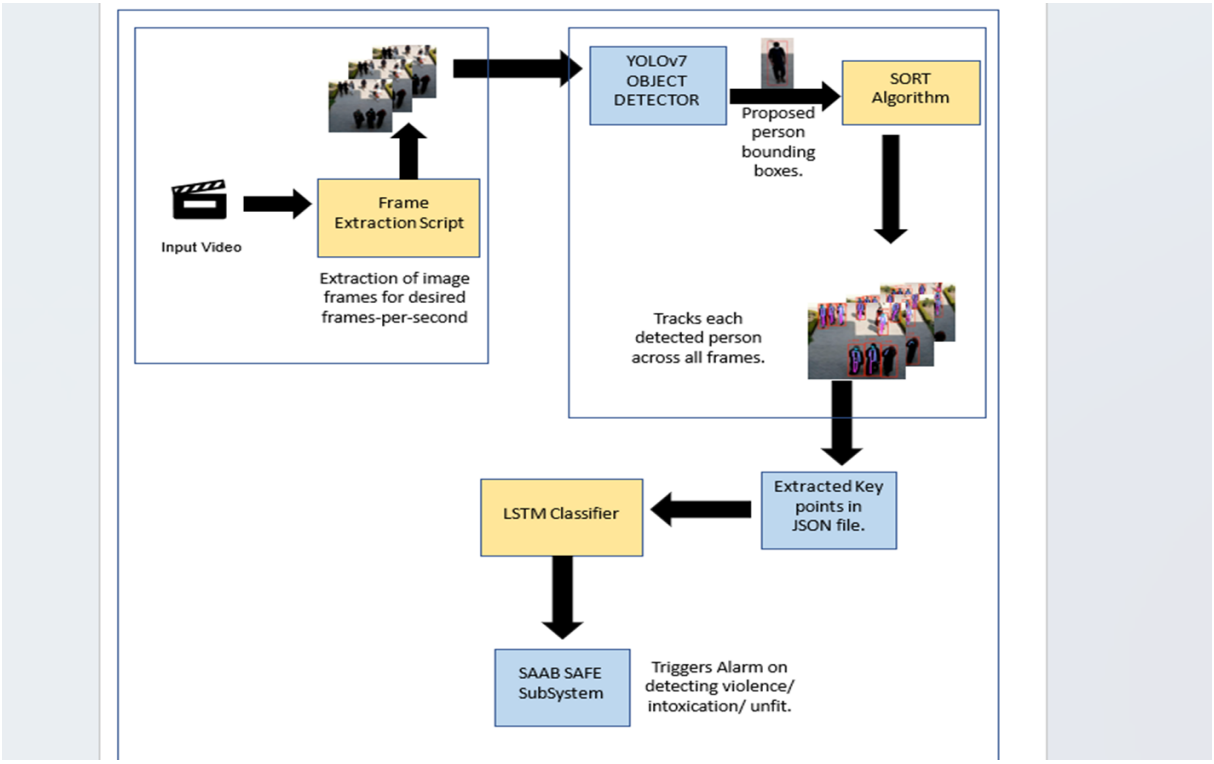

Figure 10.1: System Architecture of Our Project

Chapter 11

Results Demonstration, Evaluation and Comparison

The system is tested within our dataset and also on local computers to simulate real-world applications. SAFE integration is also evaluated.

11.1 Test dataset result

In Figure 11.1, DaVI was performing slightly worse in terms of providing more false positives. This means that when the DaVi was inaccurate, it was more likely to pass off a fit worker as unfit. This poses a problem because it would alert managerial staff of a fit worker, for them to have the knowledge to know who to further check. Putting more work into the managerial staff. However, it is concluded that the model having more false positives than false negatives is much better and safer. as it is way more dangerous to let an unfit worker continuously work.

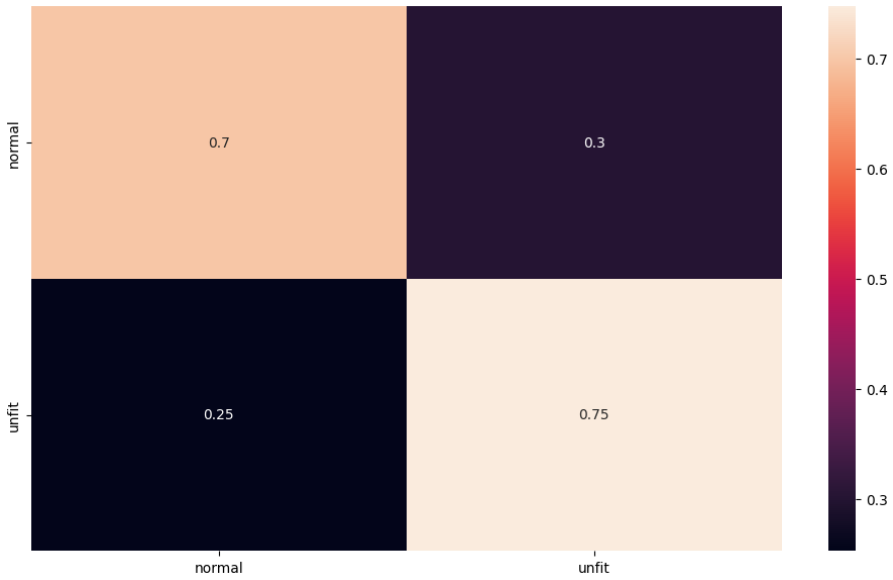

Figure 11.1: Confusion Matrix of Davi for test data

11.2 Real time interference

The entire system is tested with webcam on a local computer. When running the system, keypoints would be extracted using yolov7 and tracked by SORT. The indexed key points are then stored in a 60-frame sliding window. The key points data are then transformed to

Figure 11.2: Example of message that will push to Hilda

momentum and angles simultaneously. The transformed features are then inputted into the DaVi classifier for classification. When this buffer is filled with a majority of unfit postures and movements the classifier determines that the baseline is unnatural and therefore classifies the person as being unfit. This has the positive of allowing the workers to continue with their daily routines such as bending down or picking up items without the risk of them being classified as unfit for work. Furthermore, the message to the SAFE subsystem is only sent when there is unfit activity detected for an average of 30 frames, a message would then be sent out to an admin for them to confirm with the worker before taking further action if needed.

11.3 SAFE Integration

All the code was written and executed locally and successfully, and after many meetings with the head and HILDA and the SAAB engineers, we settled on a specific architecture to be used. This architecture included retrieving the raw feed from the cameras through the switch, running our algorithm on the feed and sending our output message to Kafka which would finally be fused to the SAFE server and client. However, we were unable to fully test the integration with SAAB SAFE as we had no access to a SAFE client and we had limited time to test on HILDA. We are however confident that the code currently present on HILDA is ready to run.

Chapter 12

Conclusion

12.1 Conclusion

In conclusion, this project successfully demonstrated the potential of using an AI-based pose detection and fitness assessment system to identify drunk, violent, or unfit maintenance operators in an airport environment. By leveraging state-of-the-art pose estimation techniques, such as YOLOv7, a SORT tracker and incorporating an LSTM classifier, our model was able to effectively detect and track human poses and classify the fitness levels of maintenance workers based on their body movements and postures.

After further consideration, the neural network provided to be most successful at classifying the live feed from cameras requires a much larger dataset, of fit and unfit behaviour that could be exhibited by maintenance staff in airports for the neural network to be able to make a better distinction. This mean that more video data of different unfit behaviours should be included, such as limping, a larger violent dataset, as well as a larger drunk dataset. This would help clearly identify the reason for the overfitting found, as well as improve the accuracy of the neural network when tested in working conditions.

The integration of this system with SAAB's SAFE subsystem allowed for seamless data exchange, visualization, and real-time monitoring of maintenance operators' fitness levels, providing a valuable tool to enhance safety and efficiency in airport maintenance operations. The system not only addressed some limitations of facial expression recognition and eye blinking analysis techniques, such as lighting conditions and occlusion, but also provided a more comprehensive assessment by considering the entire body context.

In summary, this project has shown the potential of AI-based pose detection and fitness assessment systems in improving airport maintenance safety and operations. By continuing to refine and develop the system, it can become an invaluable tool for airports worldwide, helping to prevent accidents, reduce risks, and ensure smooth and efficient airport operations.

12.2 Future work

The project encompassed development and implementation of the YoloV7 pose detection model to accurately detect and track the pose of human maintenance operators in a smart airport setting. The YoloV7 model can be customized and fine-tuned to address the unique challenges and requirements of airport maintenance operations, such as diverse lighting conditions, background noise, and multiple moving objects.

Using pose estimation alone may not be sufficient to classify drunk, fatigued, or unfit maintenance operators accurately. Combining pose estimation with other techniques, such as facial expression recognition, eye tracking, or physiological measures, can improve the overall performance and

reliability of the classification system. Data from wearable devices like smartwatches or fitness bands can be incorporated to provide additional insights into worker fatigue, heart rate, and overall well-being. Also, pose estimation can be combined with other sensor data, or worker biometric data, to improve the overall accuracy of the system in detecting unfit workers.

The model can be expanded to monitor workers in other industries, such as construction, manufacturing, or logistics, where worker safety is crucial. It can also be used to detect and classify other hazardous behaviours, such as improper lifting techniques or unsafe work practices.

We can take benefit of the advancements in pose estimation techniques and deep learning algorithms to develop more accurate and robust models. This could include the use of larger or more diverse training datasets, incorporating new architectures, or implementing techniques like transfer learning.

In cases where multiple cameras capture the same maintenance worker, the model can be evolved to perform multi-camera pose estimation to generate a consistent and accurate pose representation by combining information from different camera views. This may involve using techniques such as camera calibration, geometry, and 3D reconstruction to align and merge the pose information from different perspectives. However, for the project we have not implemented the same and it can be part of future work.

Future improvements to this project could also include incorporating multimodal data sources, optimizing the system for real-time performance, and enhancing data privacy and security measures. Additionally, the user interface and visualization tools could be refined to provide a more intuitive and actionable experience for airport staff, maintenance supervisors, and security personnel.

Bibliography

[1] H Røgind et al. "Postural sway in normal subjects aged 20-70 years". In: (). DOI: 10.1046/j.1475-097X.2003.00492.x. URL: https://onlinelibrary.wiley.com/doi/10.1046/j.1475-097X.2003.00492.x.

[2] Sean Maudsley-Barton Id et al. "A new process to measure postural sway using a Kinect depth camera during a Sensory Organisation Test". In: (2020). DOI: 10.1371/journal.pone.0227485. URL: https://doi.org/10.1371/journal.pone.0227485.

[3] Minxiang Ye et al. "A Depth Camera Motion Analysis Framework for Tele-rehabilitation: Motion Capture and Person-Centric Kinematics Analysis". In: *IEEE Journal on Selected Topics in Signal Processing* 10 (5 Aug. 2016), pp. 877–887. ISSN: 19324553. DOI: 10.1109/JSTSP.2016.2559446.

[4] Avigyan Sinha and Aneesh R P. "Drowsiness Detection System Using Deep Learning". In: *2021 Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII)* (2021). DOI: 10.1109/ICBSII51839.2021.9445132.

[5] Li-Wei Ko et al. "Driver Drowsiness Detection Based on Face Feature and PERCLOS You may also like Eyeblink recognition improves fatigue prediction from single-channel forehead EEG in a realistic sustained attention task Soft Computing Techniques for Driver Alertness The effects of unintentional drowsiness on the velocity of eyelid movements during spontaneous blinks Driver Drowsiness Detection Based on Face Feature and PERCLOS". In: *Journal of Physics* (2018), p. 12037. DOI: 10.1088/1742-6596/1090/1/012037.

[6] *YOLOv7 Pose vs MediaPipe in Human Pose Estimation*. URL: https://learnopencv.com/yolov7-pose-vs-mediapipe-in-human-pose-estimation/.

[7] Sarah Mroz et al. "Comparing the Quality of Human Pose Estimation with BlazePose or OpenPose". In: *BioSMART 2021 - Proceedings: 4th International Conference on Bio-Engineering for Smart Technologies* (2021). DOI: 10.1109/BIOSMART54244.2021.9677850.

[8] *Multi-Person Pose Estimation with Mediapipe — by Shawn Tng — Medium*. URL: https://shawntng.medium.com/multi-person-pose-estimation-with-mediapipe-52e6a60839dd.

[9] Zhe Cao et al. "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* (). URL: https://gineshidalgo.com.

[10] Debapriya Maji et al. "YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2022-June (Apr. 2022), pp. 2636–2645. ISSN: 21607516. DOI: 10.1109/CVPRW56347.2022.00297. URL: https://arxiv.org/abs/2204.06806v1.

[11] *Human Pose Estimation: Deep Learning Approach [2023 Guide]*. URL: https://www.v7labs.com/blog/human-pose-estimation-guide.

[12] Weiqiang Li, Jiatong Mu and Guizhong Liu. "Multiple Object Tracking with Motion and Appearance Cues". In: *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019* (Sept. 2019), pp. 161–169. DOI: 10.1109/ICCVW.2019.00025. URL: https://arxiv.org/abs/1909.00318v1.

[13] Jeremy Cohen. *Exactly how the Hungarian Algorithm works*. Feb. 2023. URL: https://www.thinkautonomous.ai/blog/hungarian-algorithm/.

[14] Jeremy Cohen. *Computer Vision for Multi-Object Tracking — Live Example*. Apr. 2019. URL: https://www.thinkautonomous.ai/blog/computer-vision-for-tracking/.

[15] Xi Chen, Xiao Wang and Jianhua Xuan Bradley. "Tracking Multiple Moving Objects Using Unscented Kalman Filtering Techniques". In: ().

[16] Alex Bewley et al. "SIMPLE ONLINE AND REALTIME TRACKING". In: (2017). URL: https://github.com/abewley/sort.

[17] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[18] Gábor Melis, Chris Dyer and Phil Blunsom. "On the state of the art of evaluation in neural language models". In: *arXiv preprint arXiv:1707.05589* (2017).

[19] Yu Luo et al. "Arbee: Towards automated recognition of bodily expression of emotion in the wild". In: *International Journal of Computer Vision* 128.1 (2019), pp. 1–25. DOI: 10.1007/s11263-019-01215-y.

[20] Vineet Mehta et al. "DIF : Dataset of perceived intoxicated faces for drunk person identification". In: *2019 International Conference on Multimodal Interaction* (2019). DOI: 10.1145/3340555.3353754.

[21] Tal Hassner, Yossi Itcher and Orit Kliper-Gross. "Violent flows: Real-time detection of violent crowd behavior". In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2012). DOI: 10.1109/cvprw.2012.6239348.

[22] Tanja Bänziger, Marcello Mortillaro and Klaus R. Scherer. "Introducing the Geneva Multimodal Expression Corpus for experimental research on emotion perception." In: *Emotion* 12.5 (2012), pp. 1161–1179. DOI: 10.1037/a0025827.

[23] Miriana Bianculli et al. "A dataset for automatic violence detection in videos". In: *Data in Brief* 33 (2020), p. 106587. DOI: 10.1016/j.dib.2020.106587.

[24] Pranoyr. *Pranoyr/CNN-LSTM: CNN LSTM architecture implemented in Pytorch for Video Classification*. URL: https://github.com/pranoyr/cnn-lstm#cnn-lstm.

[25] Francisco Ordóñez and Daniel Roggen. "Deep Convolutional and LSTM Recurrent Neural Networks for multimodal wearable activity recognition". In: *Sensors* 16.1 (2016), p. 115. DOI: 10.3390/s16010115.

Appendix A

Image

A.1 TensorBoard Chart

Several model with different hyper-parameter were trained and tested.

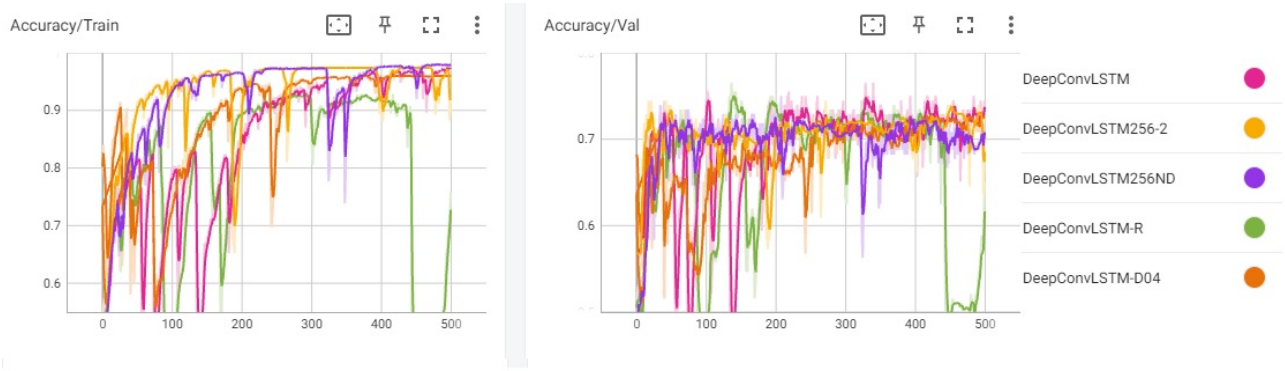

Figure A.1: Accuracy for different tested architecture

Figure A.2: Loss for different tested architecture

Appendix B

Code

All Python file can be found in https://github.com/DavidMihalcea880/GDP2023

B.1 LSTM model

```
from torch import nn
from torch.nn.modules import dropout
class MyModel(nn.Module):
    def __init__(self, input_size=51, hidden_size=256, num_classes=2):
        super(MyModel, self).__init__()
        self.lstm1 = nn.LSTM(input_size=input_size, hidden_size=hidden_size,
                             num_layers=2,dropout=0.4)
        self.fc1 = nn.Linear(hidden_size, 64)
        self.fc2 = nn.Linear(64, num_classes)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=0.1)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        out, (ht, ct) = self.lstm1(x)
        # take last time-step output
        out = out[-1, :, :]
        out = self.dropout(out)
        out = self.relu(self.fc1(out))
        out = self.softmax(self.fc2(out))
        return out
```

Listing B.1: LSTM Training Model

```
from torch import nn
from torch.nn.modules import dropout
class MyModel(nn.Module):
    def __init__(self, input_size=51, hidden_size=256, num_classes=2):
        super(MyModel, self).__init__()
        self.lstm1 = nn.LSTM(input_size=input_size, hidden_size=hidden_size,
                             num_layers=2,dropout=0.4)
        self.fc1 = nn.Linear(hidden_size, 64)
        self.fc2 = nn.Linear(64, num_classes)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=0.1)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        out, (ht, ct) = self.lstm1(x)
        # take last time-step output
        out = out[-1, :, :]
        out = self.dropout(out)
        out = self.relu(self.fc1(out))
```

```
        out = self.softmax(self.fc2(out))
        return out
```

Listing B.2: LSTM Interference Model

B.2 Yolov7 Keypoint extraction

```
import cv2
import torch
import sys
from torchvision import transforms
from utils.datasets import letterbox
from utils.general import non_max_suppression_kpt
from utils.plots import output_to_keypoint, plot_skeleton_kpts
import numpy as np
from random import randint
from sort import Sort, KalmanBoxTracker
import json
import os

@torch.no_grad()
def load_model(poseweights= 'yolov7-w6-pose.pt'):
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    model = torch.load(poseweights, map_location=device)['model']
    # Put in inference mode
    model.float().eval()
    if torch.cuda.is_available():
        pass
      # half() turns predictions into float16 tensors
      # which significantly lowers inference time
      #model.half().to(device)
    return model


def Output2TrackerInput(output):

    temp = output.clone() if isinstance(output, torch.Tensor) else np.copy(
    output)
    temp = temp[:,2:]
    temp[:, 0] = (temp[:, 0] - temp[:, 2] / 2)
    temp[:, 1] = (temp[:, 1] - temp[:, 3] / 2)
    temp[:, 2] = (temp[:, 0] + temp[:, 2] / 2)
    temp[:, 3] = (temp[:, 1] + temp[:, 3] / 2)
    trackerInput = []
    for i in temp:
        trackerInput.append([*i.tolist()[0:5],*[0],*i[5:]]) # 0 is the class

    return np.array(trackerInput)

def sort():

    sort_max_age = 5
    sort_min_hits = 2
    sort_iou_thresh = 0.2
    sort_tracker = Sort(max_age=sort_max_age,
                        min_hits=sort_min_hits,
                        iou_threshold=sort_iou_thresh)
    #.........................
    return sort_tracker

```

```
def rand_color():
    #........Rand Color for every trk.......
    rand_color_list = []
    for i in range(0,5005):
        r = randint(0, 255)
        g = randint(0, 255)
        b = randint(0, 255)
        rand_color = (r, g, b)
        rand_color_list.append(rand_color)
    #.....................................
    return rand_color_list

@torch.no_grad()
def video_output(frame,model,sort_tracker):

    image = frame
    # Resize and pad image
    image = letterbox(image, 640, stride=64, auto=True)[0]
    # Apply transforms
    image = transforms.ToTensor()(image)
    image = image.cuda()
    # Turn image into batch
    image = image.unsqueeze(0)
    with torch.no_grad():
        output, _ = model(image)
        output = non_max_suppression_kpt(output,
                                         0.25, # Confidence Threshold
                                         0.65, # IoU Threshold
                                         nc=model.yaml['nc'], # Number of
   Classes
                                         nkpt=model.yaml['nkpt'], # Number of
    Keypoints
                                         kpt_label=True)
        nimg = image[0].permute(1, 2, 0) * 255
        nimg = nimg.cpu().numpy().astype(np.uint8)
        nimg = cv2.cvtColor(nimg, cv2.COLOR_RGB2BGR)

        output = output_to_keypoint(output)
        if len(output) > 0:
            trackerInput = Output2TrackerInput(output)
            tracked_dets = sort_tracker.update(trackerInput)

            return nimg,tracked_dets

    return nimg,None


def main(source=0):
    #Select GPU
    device = torch.device("cuda:0")
    #Load Model
    model = load_model()
    sort_tracker = sort()

    #o
    cap = cv2.VideoCapture(source)
    ret= {}
    while cap.isOpened:
        check, frame = cap.read()
        if check:
```

```
            frame = letterbox(frame, 640, stride=64, auto=True)[0]
            nimg,output=video_output(frame,model,sort_tracker)
            if output is not None:
                for i in output:
                        if i[-1] in ret:
                            ret[i[-1]].append(i[8:-1].tolist())
                        else:
                            ret[i[-1]] = [i[8:-1].tolist()]
            key = cv2.waitKey(1)
            if key == ord('c'):
                break
        else:
            break
    cap.release()
    cv2.destroyAllWindows()
    #ret to json
    outputloc = source.split('.')[0].split('/')[-1]
    # Create the output directory if it does not exist
    output_dir = f'output/{folder_name}'
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    with open(f'output/{folder_name}/{outputloc}.json', 'w') as fp:
        json.dump(ret, fp, indent=4)
    KalmanBoxTracker.reset()


if __name__ == '__main__':
    # Iterate over the files in the specified folder
    folder_name = sys.argv[1]
    for file in os.listdir(folder_name):
        if file.endswith(".mp4"):
            print("Processing:", file)
            main(f'{folder_name}/{file}')
```

Listing B.3: Yolov7 Keypoint extraction script

B.3 Classifier training

```
# -*- coding: utf-8 -*-
"""TrainLSTM.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1dIFrScfDT3x9b7kU-SII0OzJ0rCYi6BI
"""

# Commented out IPython magic to ensure Python compatibility.
import torch
from torchvision import transforms
from torch import nn, optim
from torch.utils.data import DataLoader, TensorDataset, random_split
import matplotlib.pyplot as plt
import cv2
import numpy as np
import time
import os
import numpy as np
```

```
import json
from torch.utils.data import Dataset
from torch.nn.utils.rnn import pad_sequence, pack_padded_sequence,
    pad_packed_sequence

# %matplotlib inline

class LSTMDataset(Dataset):
    def __init__(self, sequences, labels):
        self.sequences = sequences
        self.labels = labels

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, index):
        sequence = self.sequences[index]
        label = self.labels[index]
        return sequence, label

def collate_fn(batch):
    sequences = [item[0] for item in batch]
    lengths = [len(seq) for seq in sequences]
    padded_sequences = pad_sequence(sequences, batch_first=False)
    labels = torch.stack([item[1] for item in batch])
    labels = labels.view(-1)
    return padded_sequences, labels , lengths

from torch.nn.modules import dropout
class MyModel(nn.Module):
    def __init__(self, input_size, hidden_size=256, num_classes=2):
        super(MyModel, self).__init__()
        self.lstm1 = nn.LSTM(input_size=input_size, hidden_size=hidden_size,
                             num_layers=2,dropout=0.4)
        self.fc1 = nn.Linear(hidden_size, dense_n)
        self.fc2 = nn.Linear(dense_n, num_classes)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=0.1)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x, lengths):
        # Pack padded sequences
        packed_inputs = pack_padded_sequence(x, lengths, batch_first=False,
    enforce_sorted=False)
        out, (ht, ct) = self.lstm1(packed_inputs)
        out, lengths = pad_packed_sequence(out, batch_first=False)
        # take last time-step output
        out = out[lengths - 1, range(len(lengths)), :]
        out = self.dropout(out)
        out = self.relu(self.fc1(out))
        out = self.softmax(self.fc2(out))
        return out

from torch.nn.modules import dropout
class MyModel(nn.Module):
    def __init__(self, input_size, hidden_size=256, num_classes=2):
        super(MyModel, self).__init__()
        self.lstm1 = nn.LSTM(input_size=input_size, hidden_size=hidden_size,
                             num_layers=2,dropout=0.4)
        self.fc1 = nn.Linear(hidden_size, dense_n)
```

```
        self.fc2 = nn.Linear(dense_n, num_classes)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=0.1)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        out, (ht, ct) = self.lstm1(x)
        # take last time-step output
        out = out[- 1, :, :]
        out = self.dropout(out)
        out = self.relu(self.fc1(out))
        out = self.softmax(self.fc2(out))
        return out

!unzip /content/output.zip

x=[]
y=[]

output_dir = '/content/output/walking'
# Get a list of all .json files in the directory
json_files = [f for f in os.listdir(output_dir) if f.endswith('.json')]
for json_file in json_files:
    f_name =  os.path.join(output_dir, json_file)
    with open(f_name, 'r') as js:
          a=json.load(js)
          #get momentum and angle and confident score of keypoints of each person
          c = {}
          for i in a:
              confidence = []
              c[i] = []
              for frame in a[i]:
                  confidence.append(frame[2::3])
                  del frame[2::3]
              for j in range(1,len(a[i])):           #Calculate momentum and angle
                  c[i].append([*[np.sqrt((a[i][j][k]-a[i][j-1][k])**2  + (a[i][j][k+1] - a[i][j-1][k+1])**2) for k in range(0, len(a[i][j]),2)], *[np.arctan2(a[i][j][k+1] - a[i][j-1][k+1], a[i][j][k] - a[i][j-1][k]) for k in range(0, len(a[i][j]),2)], *[(confidence[j][k] + confidence[j-1][k])/2  for k in range(0, len(confidence[j]))]])
          f=[torch.tensor(np.array(v), device='cuda', dtype=torch.float) for v in c.values() if len(v)>4]
          l=[torch.tensor([1], device='cuda') for v in c.values() if len(v)>4]
    x = x + f
    y = y + l

dataset=LSTMDataset(x,y)

# Create PyTorch DataLoader objects for batch training
Split=round(len(x)*0.8)
train_dataset, val_dataset = random_split(dataset, [Split, len(x)-Split])
train_loader = DataLoader(train_dataset, batch_size=batch_size, collate_fn=collate_fn, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size, collate_fn=collate_fn, shuffle=False)

# Define hyperparameters
```

```
input_size = 51
hidden_size = 256
dense_n= 64
num_classes = 2
learning_rate = 0.0001
batch_size = 90
num_epochs = 500

# Initialize the model, loss function, and optimizer
lstm = MyModel(input_size).cuda()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(lstm.parameters(), lr=learning_rate)

from torch.utils.tensorboard import SummaryWriter
log_dir = f'logs/DeepConvLSTM-D04'
writer = SummaryWriter(log_dir)

# Train the model
for epoch in range(num_epochs):
    # Train the model on the training set
    lstm.train()
    train_loss = 0.0
    train_acc = 0.0
    for padded_sequences, labels , lengths in train_loader:
        optimizer.zero_grad()
        outputs = lstm(padded_sequences, lengths)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        train_loss += loss.item() * padded_sequences.size(0)
        y_pred = torch.argmax(outputs, dim=1)
        train_acc += torch.sum(y_pred == labels).item()

    train_loss /= len(train_loader.dataset)
    train_acc /= len(train_loader.dataset)

    # Evaluate the model on the validation set
    lstm.eval()
    val_loss = 0.0
    val_acc = 0.0
    with torch.no_grad():
      for padded_sequences, labels , lengths in val_loader:
          outputs = lstm(padded_sequences,lengths)
          loss = criterion(outputs, labels)
          val_loss += loss.item() * padded_sequences.size(0)
          y_pred = torch.argmax(outputs, dim=1)
          val_acc += torch.sum(y_pred == labels).item()

      val_loss /= len(val_loader.dataset)
      val_acc /= len(val_loader.dataset)

    # Log the training and validation loss and accuracy
    writer.add_scalar('Loss/Train', train_loss, epoch)
    writer.add_scalar('Accuracy/Train', train_acc, epoch)
    writer.add_scalar('Loss/Val', val_loss, epoch)
    writer.add_scalar('Accuracy/Val', val_acc, epoch)

# Close the SummaryWriter object
writer.close()
```

```
189
190 # Commented out IPython magic to ensure Python compatibility.
191 #%load_ext tensorboard
192 # %reload_ext tensorboard
193 # %tensorboard --logdir=logs
194
195 torch.save(lstm.state_dict(), 'lstmmodel_weights.pth')
196
197 from sklearn.metrics import confusion_matrix
198 import seaborn as sn
199 import pandas as pd
200
201 y_pred = []
202 y_true = []
203
204 # iterate over val data
205 for inputs, labels , lengths in val_loader:
206         lstm.eval()
207         output = lstm(inputs,lengths) # Feed Networ
208         output = (torch.max(torch.exp(output), 1)[1]).data.cpu().numpy()
209         y_pred.extend(output) # Save Prediction
210
211         labels = labels.data.cpu().numpy()
212         y_true.extend(labels) # Save Truth
213
214 # constant for classes
215 classes = ('normal','unfit')
216
217 # Build confusion matrix
218 cf_matrix = confusion_matrix(y_true, y_pred)
219 df_cm = pd.DataFrame(cf_matrix / np.sum(cf_matrix, axis=1)[:, None], index =
      [i for i in classes],
220                      columns = [i for i in classes])
221 plt.figure(figsize = (12,7))
222 sn.heatmap(df_cm, annot=True)
223
224 import torch.onnx
225 torch.onnx.export(lstm,               # model being run
226                   x,                         # model input (or a tuple for
    multiple inputs)
227                   "lstm.onnx",   # where to save the model (can be a file or
     file-like object)
228                   export_params=True,        # store the trained parameter
    weights inside the model file
229                   opset_version=10,          # the ONNX version to export
    the model to
230                   do_constant_folding=True,  # whether to execute constant
    folding for optimization
231                   input_names = ['input'],   # the model's input names
232                   output_names = ['output']) # the model's output names
```

Listing B.4: Classifier training script

B.4 Interference

```
1 import cv2
2 import torch
3 from torchvision import transforms
4 from utils.datasets import letterbox
5 from utils.general import non_max_suppression_kpt
```

```python
from utils.plots import output_to_keypoint, plot_skeleton_kpts
import matplotlib.pyplot as plt
import numpy as np
from lstm import MyModel
from random import randint
from sort import Sort, KalmanBoxTracker
import time
import json
import os


def draw_boxes(img, bbox, identities=None, names=None,offset=(0, 0)):
    for i, box in enumerate(bbox):
        x1, y1, x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[1]
        y2 += offset[1]
        id = int(identities[i]) if identities is not None else 0
        data = (int((box[0]+box[2])/2),(int((box[1]+box[3])/2)))
        label = str(id)
        (w, h), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
        cv2.rectangle(img, (x1, y1), (x2, y2), (255,0,20), 2)
        cv2.rectangle(img, (x1, y1 - 20), (x1 + w, y1), (255,144,30), -1)
        cv2.putText(img, label, (x1, y1 - 5),cv2.FONT_HERSHEY_SIMPLEX,
                    0.6, [255, 255, 255], 1)
        # cv2.circle(img, data, 6, color,-1)   #centroid of box

    return img



def loadLSTM(lstmweights= 'lstm.pt'):
    lstm = MyModel().cuda()
    lstm.load_state_dict(torch.load('lstmmodel_weights.pth'))
    lstm.eval()
    return lstm


@torch.no_grad()
def load_model(poseweights= 'yolov7-w6-pose.pt'):
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    model = torch.load(poseweights, map_location=device)['model']
    # Put in inference mode
    model.float().eval()
    return model

def Output2TrackerInput(output):

    temp = output.clone() if isinstance(output, torch.Tensor) else np.copy(
    output)
    temp = temp[:,2:]
    temp[:, 0] = (temp[:, 0] - temp[:, 2] / 2)
    temp[:, 1] = (temp[:, 1] - temp[:, 3] / 2)
    temp[:, 2] = (temp[:, 0] + temp[:, 2] / 2)
    temp[:, 3] = (temp[:, 1] + temp[:, 3] / 2)
    trackerInput = []
    for i in temp:
        trackerInput.append([*i.tolist()[0:5],*[0],*i[5:]]) # 0 is the class

```

```
    return np.array(trackerInput)

def sort():

    sort_max_age = 5
    sort_min_hits = 2
    sort_iou_thresh = 0.2
    sort_tracker = Sort(max_age=sort_max_age,
                        min_hits=sort_min_hits,
                        iou_threshold=sort_iou_thresh)
    #......................
    return sort_tracker

def rand_color():
    #........Rand Color for every trk.......
    rand_color_list = []
    for i in range(0,5005):
        r = randint(0, 255)
        g = randint(0, 255)
        b = randint(0, 255)
        rand_color = (r, g, b)
        rand_color_list.append(rand_color)
    #.....................................
    return rand_color_list

@torch.no_grad()
def video_output(frame,model,sort_tracker):

    image = frame
    # Resize and pad image
    image = letterbox(image, 640, stride=64, auto=True)[0]
    # Apply transforms
    image = transforms.ToTensor()(image)
    image = image.cuda()
    # Turn image into batch
    image = image.unsqueeze(0)
    with torch.no_grad():
        output, _ = model(image)
        output = non_max_suppression_kpt(output,
                                         0.25, # Confidence Threshold
                                         0.65, # IoU Threshold
                                         nc=model.yaml['nc'], # Number of
    Classes
                                         nkpt=model.yaml['nkpt'], # Number of
     Keypoints
                                         kpt_label=True)
        nimg = image[0].permute(1, 2, 0) * 255
        nimg = nimg.cpu().numpy().astype(np.uint8)
        nimg = cv2.cvtColor(nimg, cv2.COLOR_RGB2BGR)
        output = output_to_keypoint(output)

        if len(output) > 0:
            trackerInput = Output2TrackerInput(output)
            tracked_dets = sort_tracker.update(trackerInput)
            for idx in range(output.shape[0]):
                plot_skeleton_kpts(nimg, output[idx, 7:].T, 3)
            draw_boxes(nimg, tracked_dets[:, :4], tracked_dets[:, -1])
            return nimg,tracked_dets

    return nimg,None
```

```
123
124 def main():
125     # Initialize time variables
126     last_wipe_time = time.time()
127     #Select GPU
128     device = torch.device("cuda:0")
129     #Load Model
130     model = load_model()
131     sort_tracker = sort()
132     #Load LSTM
133     lstm = loadLSTM()
134     #cap = cv2.VideoCapture("rtsp://safeoper:hnTe-$k2x-!sZq-mWo9@trc.dartec.
    cranfield.ac.uk/axis-media/media.amp")
135     cap = cv2.VideoCapture(0)
136     ret= {}
137     LabelHistory = {}
138     labels = ["fit","unfit"]
139     while cap.isOpened:
140         check, frame = cap.read()
141         if check:
142             frame = letterbox(frame, 640, stride=64, auto=True)[0]
143             nimg,output=video_output(frame,model,sort_tracker)
144             if output is not None:
145                 for i in output:
146                         if i[-1] in ret:
147                             if len(ret[i[-1]]) <60:
148                                 ret[i[-1]]= np.vstack((ret[i[-1]],[i[8:-1].
    tolist()]))
149                             else:
150                                 FramesOfFeatures = []
151                                 confidence = []
152                                 temp = ret[i[-1]].tolist()
153                                 for frame in temp:
154                                     confidence.append(frame[2::3])
155                                     del frame[2::3]
156                                 for j in range(1,len(temp)):
157                                     FramesOfFeatures.append([*[np.sqrt((temp
    [j][k]-temp[j-1][k])**2  + (temp[j][k+1] - temp[j-1][k+1])**2) for k in
    range(0, len(temp[j]),2)], *[np.arctan2(temp[j][k+1] - temp[j-1][k+1],
    temp[j][k] - temp[j-1][k]) for k in range(0, len(temp[j]),2)], *[(
    confidence[j][k] + confidence[j-1][k]) /2  for k in range(0, len(
    confidence[j]))]])
158
159                                 tensorOfFeatures = torch.tensor(
    FramesOfFeatures).float().cuda()
160                                 ret[i[-1]]=ret[i[-1]][1:]
161                                 #lstm
162                                 input = torch.unsqueeze(tensorOfFeatures, 1)
163                                 output = lstm(input)
164                                 label = torch.argmax(output, dim=1)
165                                 if label == 1:
166                                     LabelHistory[i[-1]].append(1)
167                                 if len(LabelHistory[i[-1]]) == 30:
168                                     timestamp = int(time.time())
169                                     cv2.putText(nimg,f'Person{i[-1]} is
    unfit',(50,600), cv2.FONT_HERSHEY_COMPLEX, 0.9, (102, 255, 255), 2)
170                                     filename = f"Img_out/unfit_{timestamp}.
    jpg"
171                                     cv2.imwrite(filename, nimg)
172                                     LabelHistory[i[-1]] = []
```

```
                        else:
                            LabelHistory[i[-1]] = []
                            ret[i[-1]] = np.empty((0,51))
                            ret[i[-1]]= np.vstack((ret[i[-1]],[i[8:-1].
    tolist()]))
                        if time.time() - last_wipe_time >= 30:
                        # Delete all files in Img_out folder
                            for file in os.listdir("Img_out"):
                                os.remove(os.path.join("Img_out", file))
                                last_wipe_time = time.time()
            cv2.imshow('frame',nimg)
            key = cv2.waitKey(1)
            if key == ord('c'):
                break
        else:
            break
    cap.release()
    cv2.destroyAllWindows()
    #ret to json
    KalmanBoxTracker.reset()


if __name__ == '__main__':
    main()
```

Listing B.5: Interference on webcam or Hilda script