1 PROJECT

1.1 HUMAN FALL DETECTION SYSTEM

Seyit Ahmet Koçyiğit 21946366
Kenan Gökdeniz Acet 2200356034
Alper Mert 2200765029

1.2 PROJECT DEMO VIDEO

`https://www.youtube.com/watch?v=kI8upv6W7ko`

2 INTRODUCTION

Human fall detection is a critical area of research with applications in healthcare, surveillance, and assisted living. Falls are a significant concern for the elderly and individuals with mobility issues, often resulting in serious injuries or even fatalities if immediate assistance is not provided. Traditional fall detection systems, which rely on wearable sensors or manual monitoring, have limitations such as discomfort, inconvenience, and inability to provide real-time monitoring.

The aim of this project is to develop a computer vision-based system for human fall detection that can automatically detect falls in real-time. By utilizing computer vision techniques, we aim to create a non-invasive and efficient solution that can detect falls accurately, even in complex environments.

2.1 CHALLENGES

The challenges involved in human fall detection using computer vision include:

Variability in falls: Falls can occur in various ways and in different environments, making it challenging to develop a robust detection algorithm. Real-time processing: Achieving real-time performance is crucial for timely assistance. The system should be capable of processing video feeds in real-time to detect falls as they happen. Complex environments: The system should be able to detect falls accurately in diverse environments, including different lighting conditions, cluttered backgrounds, and occlusions.

2.2 MOTIVATION

The motivation behind this project is to provide an efficient and non-intrusive solution for fall detection, which can significantly improve the safety and well-being of the elderly and individuals with mobility issues. By using computer vision techniques, we aim to overcome the limitations of traditional wearable sensor-based systems and provide a more effective solution for fall detection.

2.3 RELATION TO THE THEME

This project is closely related to the theme of computer vision and its applications in healthcare and safety. By developing a computer vision-based fall detection system, we aim to leverage the advancements in computer vision technology to address a critical problem and improve the quality of life for individuals at risk of falls.

3 RELATED WORK

3.1 PROJECT 1

`https://github.com/uttej2001/Image-based-Human-Fall-Detection`

This project utilizes the YOLOv5 object detection model to detect human falls in images and videos. YOLOv5 is known for its speed and accuracy. The fall detection dataset is acquired from Kaggle and prepared for training and validation using a custom dataset definition. The model is trained for 20 epochs with transfer learning using pre-trained weights and training progress is monitored using TensorBoard. Evaluation includes logging training losses and performance metrics, and the trained model is used for fall detection in videos.

3.2 Project 2

`https://github.com/taufeeque9/HumanFallDetection`

The project developed a fall detection system using computer vision and deep learning techniques. It employs the OpenPifPaf library for pose estimation and an LSTM neural network model for fall detection classification. The code is structured to process video input in real-time, leveraging multiprocessing to parallelize keypoint extraction and the fall detection algorithm. Key components include functions for video capture, resizing, keypoint extraction, feature extraction, and fall detection. The system is configurable via command-line arguments and includes functionality for batch processing multiple video files.

3.3 Project 3

`https://github.com/KananMahammadli/Fall-Detection-using-CNN-Architecture`

This project implements a machine learning model, named fallnet, for detecting falls from images. Data is extracted from a zip file containing training and testing datasets, with labels stored in CSV files. The CNN architecture consists of 6 feature extraction blocks followed by a header, with the number of units increasing from 16 to 32 and then to 64 after every 2 blocks.

During training, the data is split into training and validation sets using stratified sampling. The model is trained for 6 epochs, with performance monitored using validation data to prevent overfitting. Evaluation includes visualizing the history of model accuracy and loss during training, saving the model, and predicting labels for test images.

4 Method

4.1 Data Extraction and Preprocessing

We begin by extracting shoulder, belly, and knee positions from video footage along the y-axis to capture vertical movement. This positional data is structured into a dataframe for each video. To incorporate temporal dynamics, lagged values of the data are computed, ranging from 1 to 4 frame differences. These lagged features provide insight into the trajectory and velocity of movement. Additionally, we label the dataset to predict the next frame given a set of previous frames, enhancing the model's understanding of temporal sequences leading to potential fall events.

4.1.1 Model Development

The core of our predictive system relies on LSTM neural networks, renowned for capturing temporal dependencies in sequential data. We train individual LSTM models for each video, treating each video's data as a separate time series.

4.2 Training Strategy

Given the sequential nature of the data, training the LSTM model on concatenated data from all videos is impractical. Instead, we adopt a video-specific training approach. Each video's data is treated as a single time series, and the LSTM model is trained to predict the next frame given a sequence of previous frames. This process constitutes one epoch of training. Subsequently, we fine-tune the model parameters using data from all remaining videos. This iterative process continues until convergence or satisfactory performance is achieved

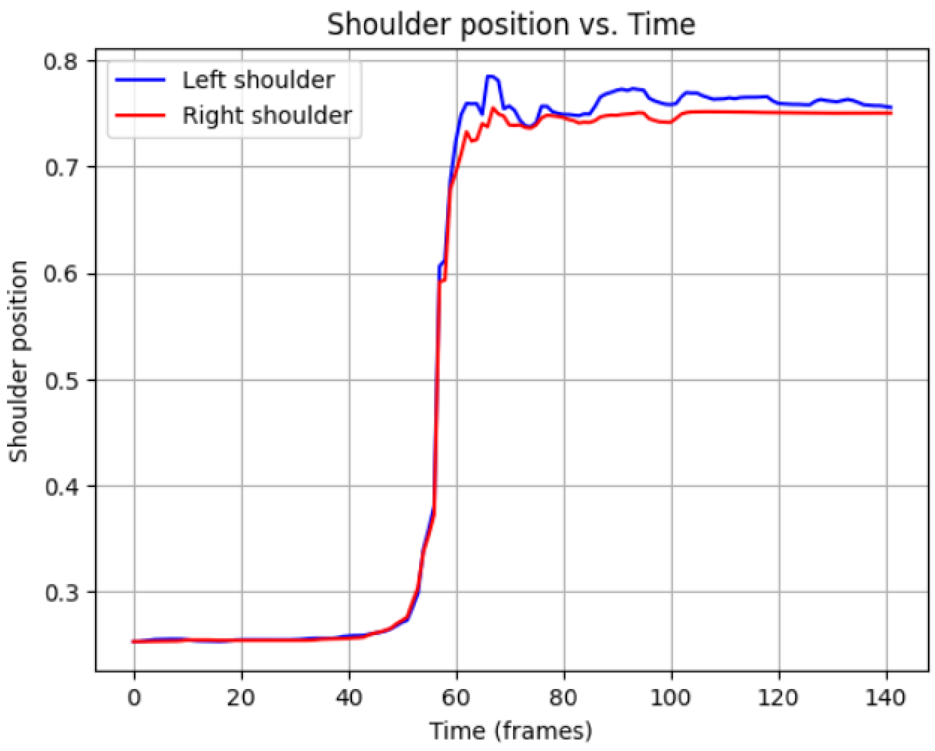


Figure 1: Shoulder y position vs time

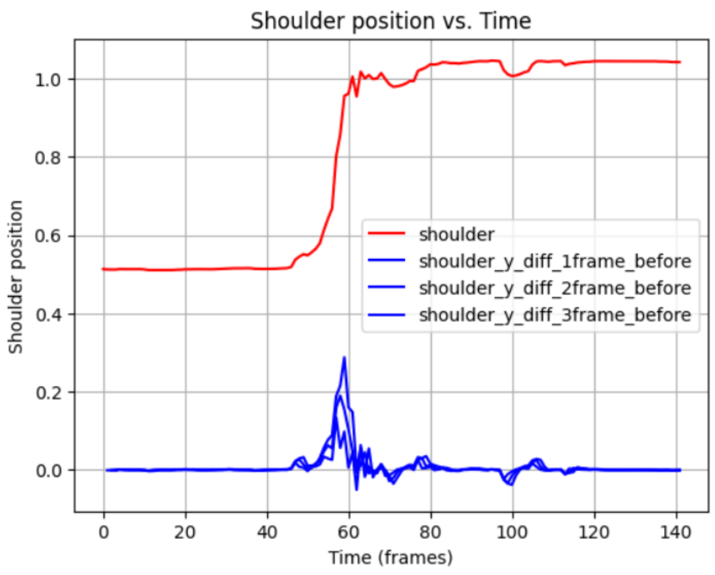


Figure 2: Shoulder y position and differences vs time

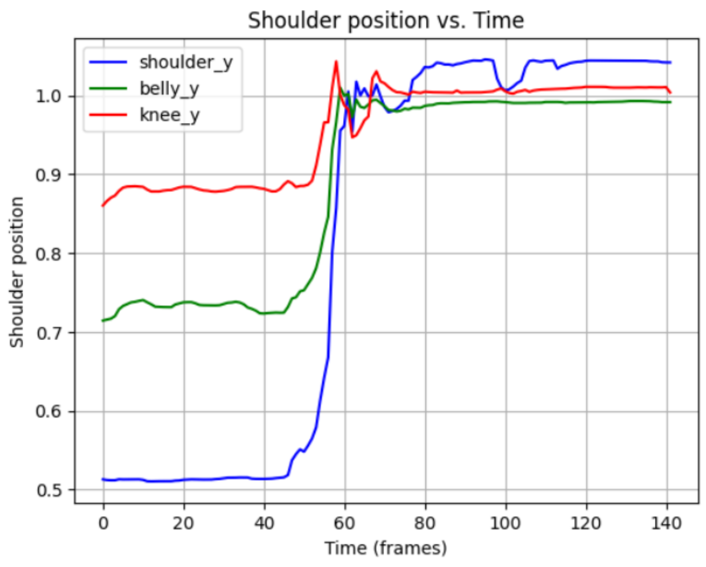


Figure 3: Extracted Features vs time

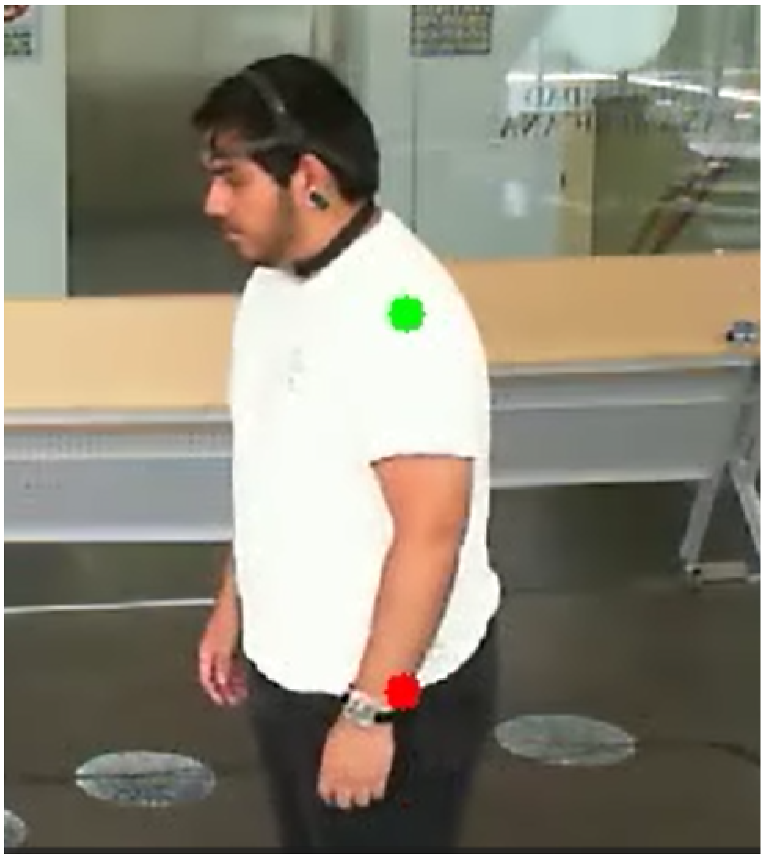


Figure 4: Interest points from side

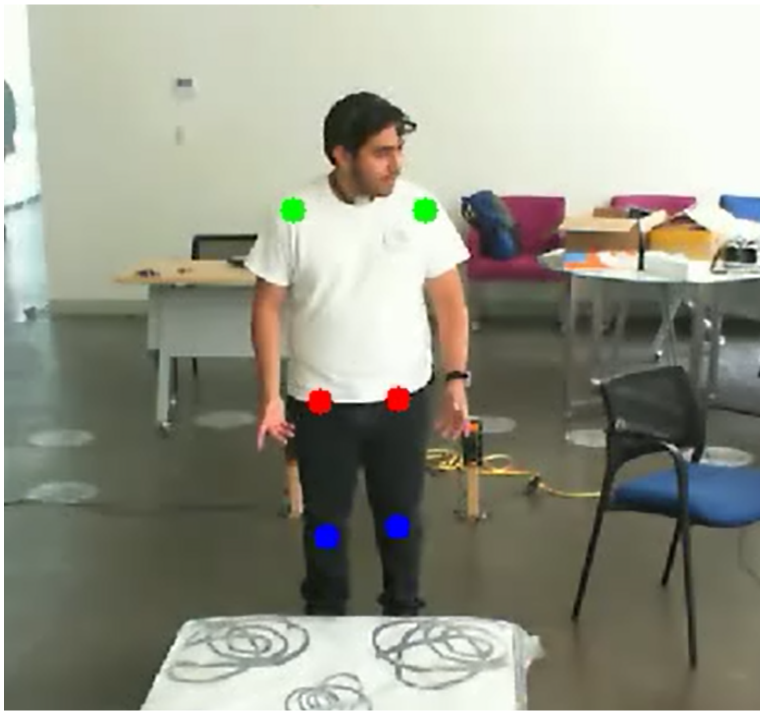


Figure 5: Interest points from front

4.3 Evaluation and Validation

To assess the model's predictive accuracy, standard evaluation metrics such as precision, recall, and F1-score are utilized. Cross-validation experiments are conducted to validate the model's robustness across different datasets. This methodological approach aims to develop an effective and robust predictive system for identifying fall events by leveraging LSTM neural networks, lagged features, and labeled data for temporal sequence prediction. Further experimentation and validation are necessary to optimize model performance and ensure real-world applicability.

5 Experimental Settings and Initial Results

5.1 Dateset

`https://sites.google.com/up.edu.mx/har-up/`

The dataset includes images. To create a sequential representation, these images will be combined to generate videos with a fixed frame rate of 18 fps. Then, feature extraction will be performed on these videos to capture shoulder, belly, and knee positions.

5.2 Feature Extraction from Dataset

We begin by extracting shoulder, belly, and knee positions from video footage along the y-axis to capture vertical movement. This positional data is structured into a dataframe for each video. To incorporate temporal dynamics, lagged values of the data are computed, ranging from 1 to 4 frame differences. These lagged features provide insight into the trajectory and velocity of movement

5.3 Initial Results

Currently we don't have any initial results we are working on data labelling for our own stategy to train our model.

6 Experimental Results

6.1 Hyper Parameter Choice

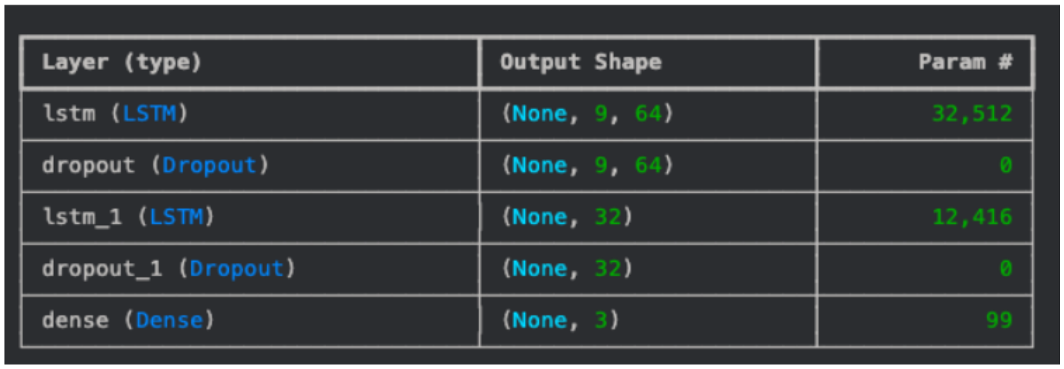
We used LTSM to train our model. For connecting the tracking points across consecutive frames we used 9 consecutive frames. We tried lower values but they didn't catch the gradient of the points as desired and the model under performed. We used higher values but they didn't give any additional accuracy for the model. So we choiced 9 for these reasons.

We trained our model for 100 epochs with a batch size of 128. The training window size was set to 9. During testing, we initially used a test window size of 4 but found that it resulted in larger errors. To improve accuracy, we switched to using test window sizes of 1.

For data scaling, we experimented with both the MinMax Scaler and the Standard Scaler. The MinMax Scaler proved to be more performant, so we used it for scaling our data.

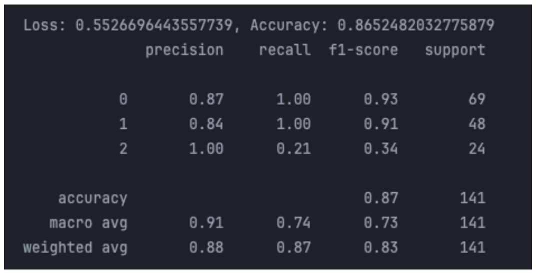
Our model architecture included two dropout layers, each with a dropout rate of 0.2, to help prevent overfitting.

This configuration helped us achieve better performance and reduce errors in our model.



Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 9, 64)	32,512
dropout (Dropout)	(None, 9, 64)	0
lstm_1 (LSTM)	(None, 32)	12,416
dropout_1 (Dropout)	(None, 32)	0
dense (Dense)	(None, 3)	99

6.2 Graphs



```
Loss: 0.5526696443557739, Accuracy: 0.8652482032775879
              precision    recall  f1-score   support

           0       0.87      1.00      0.93        69
           1       0.84      1.00      0.91        48
           2       1.00      0.21      0.34        24

    accuracy                           0.87       141
   macro avg       0.91      0.74      0.73       141
weighted avg       0.88      0.87      0.83       141
```

Figure 6: Accuracy

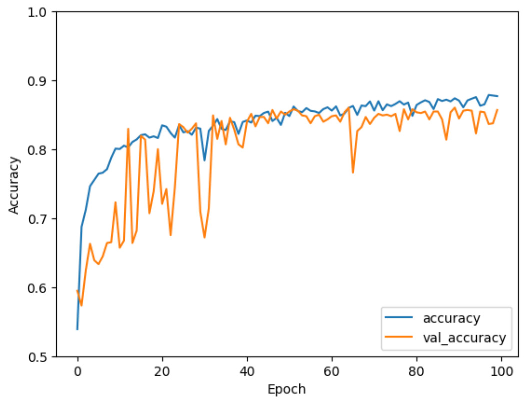
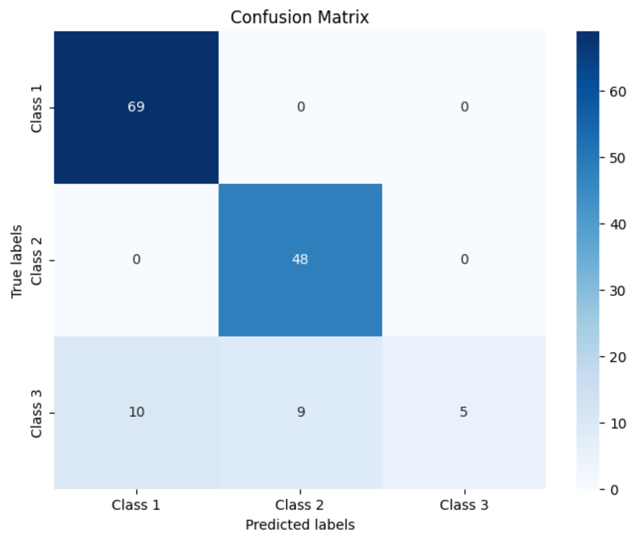


Figure 7: Val-Train Acc



Class 0: Fell
Class 1: idle
Class 2: Falling

6.3 Examples

```
https://www.youtube.com/watch?v=SYvlEdns06o
https://www.youtube.com/shorts/4zoBoHbjfZ4
https://www.youtube.com/watch?v=wQ2G4LuZqQg
```

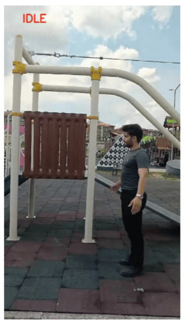


Figure 8: Idle

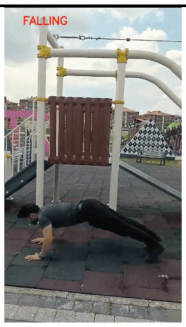


Figure 9: Falling

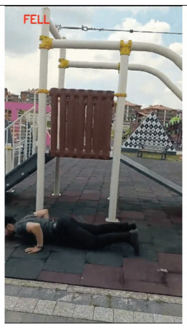


Figure 10: Fell

6.4 Code and Model

```
https://drive.google.com/drive/folders/1HO05-EnfbfUCroZNGIT0c7aNoz8Z2zYe?
usp=sharing
```

7 Discussions/Conclusions

7.1 Project is a success ?

We can say that our project is successful. We can catch the falling state of the person and we generated very small models that can be used on any hardware. We can program FPGAs to perform the same operation because we modeled our project as a dynamic system that follows a pattern.

7.2 Improve Result

To improve performance of the project we can use more tracking points and experiment with different LSTM setups. We can add more detail to the project by using CNN with point tracking.

7.3 Challenges

We have created a project from scratch by using only open-cv generated algorithms like finding special points of the human body. The data has images of frames. We combined them as a video and then labeled them with our own strategy. To label the data we extracted the special points of the human body like shoulders. After that we compared the different points positions across frames. Then we analyzed the data and created a strategy that programmatically labels the data. We added the difference of positions of points across frames. We found a threshold which lets us label data correctly. Once the sum of the difference of points is bigger than this threshold we can say that the person is falling.

7.4 Strong and weak parts

LSTM was used in our project effectively to detect human fall. With considering 9 consecutive frames to connect tracked important points, it is ensured that our model is able to capture the temporal dynamics of human movements correctly. This approach allowed the model to adapt subtle changes in motion and falling patterns, leading to robust fall detection. The selected important points of human body are easy to observe in 2 dimensions while falling. For example, while falling, shoulder's position changes are so rapid. With using OpenCV algorithms to extract key body points and comparing their positions across frames, established a methodical labeling strategy. This strategy, combined with the incorporation of point position differences and thresholding, ensured accurate annotation of fall events, laying a strong foundation for model training. For training and test parts, the data with clear falling actions sequences was used. In such cases where dataset includes sequences the falling is not clear or too fast, for example in a football match, or people with physical anomalies, our model may predict fall detection wrong.

7.5 Project outcome

Our project has 3 alternative actions for people: idle, falling, fell. While the input video is playing, our model takes each sequences frame by frame and categorized them with 3 alternate actions. If the person is standing, our project shows "IDLE" in that sequence, if the person is starting to fall, our project shows "FALLING" in that sequence, if the person is on the ground, our project shows "FELL" in that sequence. This system opens up possibilities for deploying our solution in various real life scenarios where quick response to fall incidents is crucial, such as eldercare facilities, sports events, or industrial settings. For example in eldercare facilities, timely detection of falls can lead to rapid medical assistance, reducing the risk of severe injuries. The project got high accuracy in detecting falls. This ensures that genuine fall incidents are promptly identified, enhancing safety measures. To sum up, our project not only achieves accurate fall detection but also provides a practical and scalable solution that can be applied in various settings to improve safety and response times. The outcomes of this project demonstrate its potential for significant impact in the field of human safety and monitoring.